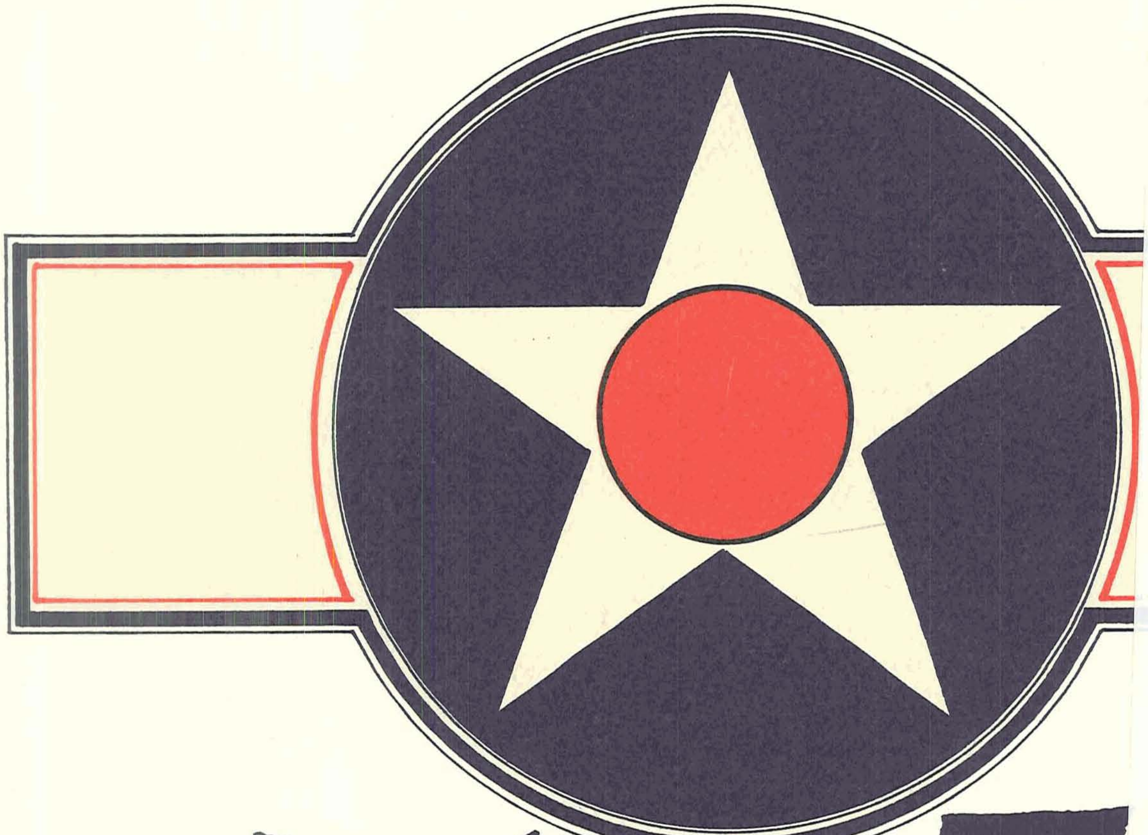
1993 NASA



5th

NASA SPACE ENGINEERING RESEARCH CENTER SYMPOSIUM ON VLSI DESIGN

5th NASA Symposium on VLSI Design

The University of New Mexico
Albuquerque, New Mexico

November 4-5, 1993

Cover by Peter Vincent

Welcome to the fifth annual NASA Symposium on VLSI Design, co-sponsored by the IEEE. Each year this symposium is organized by the NASA Space Engineering Research Center (SERC) at The University of New Mexico and is held in conjunction with a quarterly meeting of the NASA Data System Technology Working Group (DSTWG). One task of the DSTWG is to develop new electronic technologies that will meet next generation electronic data system needs. The symposium provides insights into developments in VLSI and digital systems which can be used to increase data systems performance.

The NASA SERC is proud to offer, at its fifth symposium on VLSI design, presentations by an outstanding set of individuals from national laboratories, the electronics industry and universities. These speakers share insights into next generation advances that will serve as a basis for future VLSI design.

In spite of the present economic conditions, interest in the conference has remained steady with 57 papers in 13 categories included in this years proceedings. National Laboratories are represented by Sandia National Laboratories, NASA Goddard Space Flight Center, USC Informatin Sciences Institute, Jet Propulsion Laboratory, Phillips Laboratory and the Johns Hopkins University Applied Physics Laboratory. Private industry is represented by Texas Instruments, IBM, Advanced Hardware Architectures, Systems & Process Engineering, Boeing Defense & Space Group, Honeywell Systems and Research Center, Corporation, LSI Logic Corporation, Alenia Spazio, Allied Signal Inc. and Hewlett Packard. Universities are represented by Concordia University, University of Utah, Prairie View A&M University, Christopher Newport University, State University of New York at Albany, California Institute of Technology, New Mexico State University , Nanyang Technological University, Cornell University, University of New York, Southern Illinois University, Colorado State University, University of Toledo , University of Peradeniya, University of Idaho, Brigham Young University, University of Colorado, University of Cincinatti, Grand Valley State University, San Diego University and The University of New Mexico. In addition we are happy to welcome a number of papers presented by international authors.

There are individuals whose assistance was critical to the success of this symposium. Foremost have been Barbara Martin who worked long hours to assemble the conference proceedings, Sterling Whitaker who organized the technical program and Christy Custer who organized the symposium in general. The efforts of these professionals were vital and are greatly appreciated. I hope you enjoy your stay in Albuquerque, New Mexico

Gary K. Maki

Session 1 – Featured Presentations I

Chairman: Gary Maki

Session 2 – Radiation Effects

Chairman: John Canaris

Session 3 – Architectures I

Chairman: Kelly Cameron

Session 4 – Featured Presentations II

Chairman: Nasir Ahmed

Session 5 – Mixed Signal

Chairman: **Don Wiseman**

Session 6 – Design Techniques I

Chairman: **Jody Gambles**

Session 7 – Fault Testing

Chairman: Tom Caudell

Session 8 – Architectures II

Chairman: Don Thelen

Session 9 – Synthesis I

Chairman: Phil Windley

Session 10 – Design Techniques II

Chairman: John Purviance

Session 11 – Synthesis II

Chairman: Sterling Whitaker

Session 12 – Signal Processing

Chairman: Joe Feeley

Session 13 – Included Papers

List of Authors

Session 1
Featured Presentations I

Chairman: Gary Maki

GaAs VLSI Design

Gary McMillian, Ph.D.
Vice President, Chief Scientist
Systems & Processes Engineering Corporation (SPEC)
1406 Smith Road, Austin, Texas 78721-3597
Phone: (512) 385-0082
Fax: (512) 385-3024
Internet: mcmillian@spec.com

Abstract - **Systems & Processes Engineering Corporation (SPEC) is developing libraries for design of GaAs VLSI circuits. VHDL based designs have been synthesized into a combination of standard cell and optimized layout datapath blocks using COMPASS Design AutomationUs CAE tools. The libraries are being developed for Vitesse SemiconductorUs 0.6 micron H-GaAs III process. These libraries and tools are being used by SPEC to develop a GaAs SPARC RISC processor.**

An Adaptive, Lossless Data Compression Algorithm and VLSI Implementations [1]

Jack Venbrux, Greg Zweigle †, Jody Gambles, Don Wiseman,
Warner H. Miller‡ and Pen-Shu Yeh‡

NASA Space Engineering Research Center for VLSI System Design
University of New Mexico, 2650 Yale SE, Suite 101
Albuquerque, NM 87106
† Advanced Hardware Architectures
PO Box 9669
Moscow ID 83843
‡ NASA/Goddard Space Flight Center
Code 738
Greenbelt, Maryland 20771

Abstract- **This paper first provides an overview of an adaptive, lossless, data compression algorithm originally devised by Rice in the early '70s. It then reports the development of a VLSI encoder/decoder chip set developed which implements this algorithm. A recent effort in making a space qualified version of the encoder is described along with several enhancements to the algorithm. The performance of the enhanced algorithm is compared with those from other currently available lossless compression techniques on multiple sets of test data. The results favor our implemented technique in many applications.**

1 Introduction

This paper compares the Rice algorithm [1, 2, 3, 4] with Lempel-Ziv-Welch (LZW) [5], adaptive Huffman coding, arithmetic coding [6], and two-pass Huffman based lossless JPEG coding [7]. VLSI implementations of the Rice algorithm are also described.

The Rice algorithm adapts to changes in image statistics within a single block of data (typically 16 samples). It is the most rapidly adapting algorithm tested. The Rice algorithm also has the useful property that its coding performance does not depend on past history. It keeps no log of statistics or memory of data strings. By not using a history, the Rice algorithm requires no external RAM and is amenable to applications in which small data packets are required. Should there ever be a missing packet of data, a Rice algorithm based Encoder/Decoder could continue functioning without a system reset. Algorithms that use history are not able to decode properly after a gap in the data stream.

The algorithm has been implemented in a VLSI chip set. It is the first Rice algorithm chip set and compresses data from 4 to 14 bits/sample. The encoder, called USE for Universal Source Encoder, can compress data in excess of 20 Msamples/s. The decoder, called USD for Universal Source Decoder, can decode data at half of that rate. The chip set has become a commercial product [8].

[1] This work was supported in part by NASA under Space Engineering Research Center grant NAGW-3292.

Input	FS codeword
0	1
1	01
2	001
3	0001
4	00001
5	000001
6	0000001

Table 1: Fundamental Sequence example.

A new encoding chip has been designed and is called USES (Universal Source Encoder for Space). It offers some coding improvements over the USE chip and adds features that make it easier to integrate into a system. The major coding improvement is the addition of two new very low entropy coding options that are used when data entropy is below 1 bit/sample. The system improvements include variable block sizes (8, 10, 16 samples), one more bit of quantization, ability to handle two's complement data, and on chip concatenation of coded data blocks. USES has been designed as a RAD-hard gate array. USES is baselined for LandSat 7 satellite project.

2 Rice Algorithm Overview

The Rice algorithm is a lossless compression method that is rapidly adaptive and efficient over a wide range of entropy conditions. The structure of the algorithm can best be understood through the following argument. Assume that a particular encoder has, for example, 13 different Huffman codes from which to choose. Each Huffman code is designed for a particular entropy range. The first codebook might be designed for an expected codeword length of 2 bits, while the next one might be designed for an expected codeword length of 3 bits, and so on in integer increments of one bit. By applying each of the 13 codebooks to the same data block of size J, and then choosing the codebook that produced the best compression, the coder would effectively cover a wide entropy range: approximately 1.5 bits through 14.5 bits. By deciding which coder was the winner on a block by block basis, the coder would be adaptive to changes in data entropy.

One of Robert F. Rice's accomplishments was develop a compression algorithm that requires *no codebooks* [1, 2, 3, 4], yet provides equivalent performance to the multiple Huffman codebook approach just described. The equivalence is true for sources that have a Laplacian distribution. Fortunately, for many types of sampled data, especially imagery, the prediction error can be well modeled as a Laplacian distribution. The Rice algorithm has been shown mathematically to provide equivalent performance to using multiple, parallel Huffman codes. Moreover, this result is supported by extensive simulations [9]. An intuitive feel for this equivalence can be obtained by understanding two components of the algorithm: a comma code that Rice calls the Fundamental Sequence (FS), and sample splitting.

The Fundamental Sequence is a comma code that has the property that a value m has

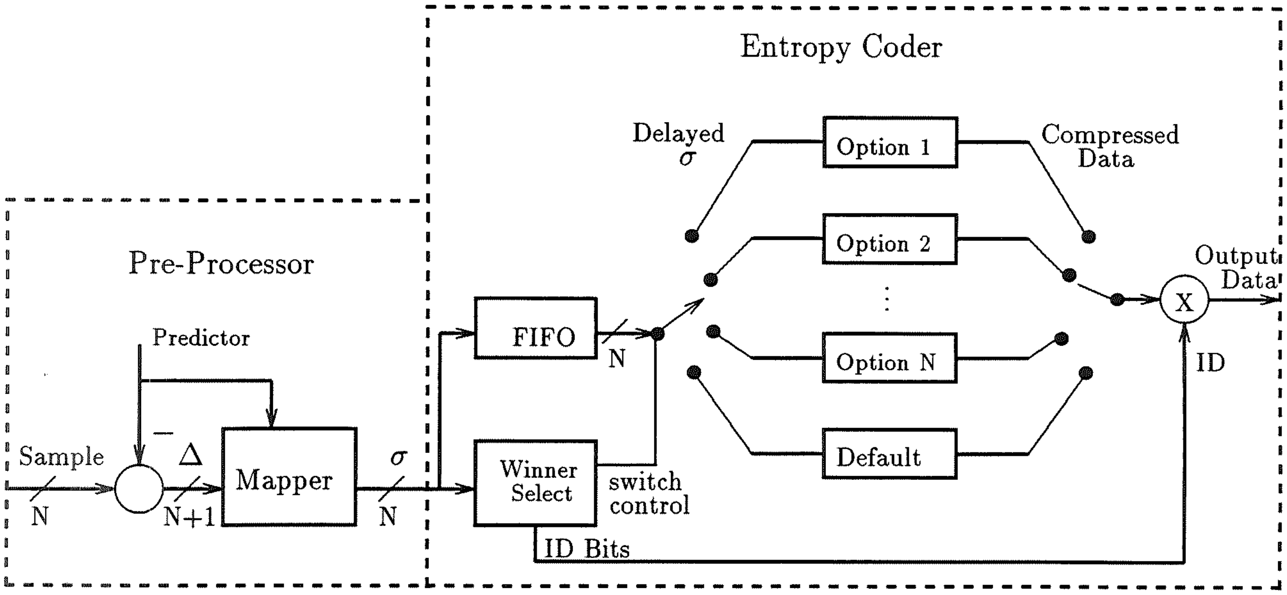


Figure 1: Block diagram of a typical Rice algorithm architecture.

a corresponding codeword that is made up of m zeros followed by a '1'. Table 1 shows an example of the FS sequence for given input values. Because each codeword is uniquely defined by simply knowing the input value, codebooks are not needed. A value of 6, for example, will be coded with 6 zeros followed by a '1'.

Sample splitting is a technique that assumes that the k least significant bits for every sample in a block are random, and therefore, can not be compressed. The Rice algorithm uses sample splitting in the following manner. First, it assumes the incoming N bit data can be compressed without splitting off any bits. The FS code is used on the incoming data words and the total bit count is recorded. Next, it assumes that the least significant bit of each sample is random and then uses the same FS code on the N-1 remaining bits. Again, the total bit count is recorded. Next, the 2 least significant bits (k=2) are assumed to be random, while the N-2 remaining most significant bits are FS coded. This process of sample splitting combined with FS coding the N-k most significant bits is continued for k= [0,1,2,..12]. The bit counts are compared and the winning sample split option is the one that codes the block of data in the fewest number of bits. Sample splitting defines 13 of the 15 code options used in the enhanced Rice encoder.

For low entropy data, direct coding of the data using k=0 (no split bits) produces the fewest number of coded bits. Higher entropy data is more efficiently coded by using sample splitting with k > 0. This reaffirms the assumption that as entropy increases, the randomness (or information content) in the k least-significant bits prohibits compression. Therefore, using successive split bits, combined with FS coding, is equivalent to using Huffman codebooks that are designed for successive integer steps in entropy. Due to the simple structure of the FS code, however, no actual codebook is necessary for each code option.

Fig. 1 shows the Rice algorithm architecture. The input section performs pre-processing on the data. The pre-processor consists of a subtractor followed by a mapping. Subtracting two N bit samples (the input sample and the predictor) results in an N+1 bit delta value.

The mapper maps the N+1 difference back to an N bit difference.

The entropy coder codes small blocks of samples (16 for example). The enhanced coder consists of a very low entropy coder, 13 coders that use sample splitting with k=[0,1,2,..12], and one default coder. The default coder passes data uncompressed. The very low entropy coder is a combination of two options, one, developed at Goddard Space Flight Center, efficiently codes in the range 0.5-1.0 bit/sample. The second low entropy coding option counts runs of zero entropy blocks.

Before processed data is sent out, ID bits are prepended to inform the decoder which of the code options have been used to compress the block of data. In viewing Fig. 1 it may appear unusual that a winner select block occurs *before* the code option blocks. The winner select block chooses which of the coders will code the block in the fewest number of bits. This is possible due to the simple structure of the FS code; the codeword size is always known before any coding takes place. A value m always has a codeword length of m+1. For a k split code option, the compressed bit count will always be (k*J)+FS bits, where J is the block size. The winner select does a trial coding run on the block of data by calculating how many bits it will take to encode with each of the code options. The winner is then selected. Because the code option blocks are so similar in structure, much hardware can be shared. Instead of having 15 coding blocks, each doing bit packing, two coding blocks were designed in the encoder that perform the necessary coding, resulting in a savings in chip area.

The method just described for determining the winner relies on exact counts– there are no approximations and the coding performance is optimal. Rice and associates at JPL have developed approximations ("decision regions") that simplify hardware or software requirements for the selection process [1].

Coding an image begins with saving a reference sample upon which the subsequent delta value is derived. The predictor could be externally supplied or default to the previous sample in a scanline. Differences between the predictor and the present sample are mapped into sigma values and sent to the entropy coder. While the winner select is calculating the winner, the sigma values are held in a FIFO. This FIFO is one of the few RAM elements required. The algorithm does not require any RAM to store codebooks or tables for statistics. Once the winner is selected, the sigma values are coded by the appropriate code option block. ID bits are prepended to the compressed data, and then output. The decoder performs the reverse of the encoding process.

3 Algorithm Comparisons

In this section the Rice algorithm will be compared with other lossless compression algorithms. The compression performance comparisons are reserved for the next section.

The algorithms to be compared with the Rice algorithm are:

- Lempel-Ziv Welch (LZW) [5]
- Adaptive Huffman Coding
- Adaptive Arithmetic Coding as described in [6].
- Lossless Huffman based JPEG using two data passes [7].

Each algorithm will be discussed in terms of the following parameters:

- Adaptation: ability to adapt to changing scene statistics.
- Memory requirements
- Error propagation

3.1 Adaptation

Different images may have widely varying scene statistics. Even within an image, scene statistics vary. In order to losslessly compress data, redundant information must be removed from the data. Frequently, this involves some sort of prediction method that is based on previous data. Without adaptation, predictors that work well in one section of an image will incur greater error in another section of the image. Within the five algorithms discussed in this paper, there are four different approaches to adaptation.

The two algorithms that are similar in terms of adaptation are adaptive Huffman coding and Arithmetic coding. Both techniques constantly update statistics on the incoming data. For the Huffman technique, the most likely occurring data element will be given the smallest codeword. In arithmetic coding the most likely occurring element is given an numeric value representative of the inverse of its probability. These two techniques require a history of past data in order to obtain meaningful source statistics.

Lossless JPEG coding adapts to image statistics on an image section by image section basis. Although it is possible to operate lossless Huffman based JPEG using a fixed table, the table will only be optimal for an image with a given set of statistics. The Huffman based JPEG coder results discussed in the next section are based on a two pass approach. In the first pass, optimal Huffman tables are created for the image. The second pass performs the coding using the tables. The compressed image data must therefore include the Huffman tables along with the compressed data.

LZW performs compression without prediction. Like the other algorithms just discussed it uses a history of past data. Instead of keeping statistics, LZW keeps past strings of data in memory. Compression occurs when the algorithm can express an incoming string as a pointer to an existing string.

The RICE algorithm is unique in this comparison in that it's prediction is not based on past history. It adapts on a block by block basis with blocks typically having only 16 samples. It does not keep a record of statistics nor does it create an optimal table. Because of the algorithm's novel approach to adaptation, the Rice algorithm has some advantages over the other algorithms in terms of reduced system memory requirements, reduced overhead due to not having to transmit code books, flexibility in the design of data packets (such as found in satellite communications), and the ability to contain channel errors within a single packet.

3.2 Memory requirements

Any algorithm that updates a set of statistics requires memory. Adaptive Huffman coding dynamically changes a Huffman code based on the statistics of image. Arithmetic coding, as implemented by Whitten, Neil, and Cleary [6], requires a table of counts to be updated.

For 8 bit data, the frequency table requires 256 entries. Higher quantizations require more memory.

Although LZW does not keep a table of statistics, it requires memory to store past data strings. The more history that is kept around, the more memory is required. UNIX compress uses 64k of memory to store past strings. A hardware system of the LZ algorithm [10, 11] that supports quantizations up to 15 bits used 520 kbytes of memory. Other commercially available chips that operate on 8 bit data, such as [12], require less memory.

To achieve maximum compression, lossless, Huffman based, JPEG, requires two passes: one to build optimal tables and the second pass to code the image. This means that the image as well as the Huffman table must be stored before coding.

The Rice algorithm requires no external memory for either encoding or decoding. The USE and USD chip set contained only 340 internal bytes of memory. USES, which was done in a gate array, requires less than 100 bytes of memory.

3.3 Packet size and Error Propagation

In satellite transmission, compressed data is likely to be transferred in packets rather than a continuous stream. If a packet full of compressed data has an error, or if a packet is lost, the affect on decompressing data can be severe. To reduce the impact of a possible error, it may be desirable to send small packets of data rather than large ones. Among the algorithms under consideration, only the Rice algorithm is amenable to small packet sizes. The reason for this is that it requires no history of either statistics or strings. To provide the best coding performance, the other algorithms must develop a meaningful set of statistics or strings of data. Although all of the algorithms adapt, none of the algorithms adapt as quickly as the Rice algorithm. For example, the satellite image suite, discussed in the next section, includes 6 small images. The Rice algorithm out performed all of the other algorithms in compressing these images.

If a packet of data were lost, all the algorithms that keep a history would have wrong statistics at the start of a new packet. Incorrect statistics would cause decoders to loose synchronization. This is true for adaptive Huffman coding, LZW, and arithmetic coding. JPEG coding, however, can be performed on an image section by section basis. As long as a JPEG packet were made up of complete image sections along with the corresponding Huffman tables, the JPEG decoder could resynchronize on the next packet. The Rice algorithm has no packet size restrictions (such as containing an entire image section), and errors in transmission will always be contained within a given packet. Because the Rice algorithm does not keep a history, the decoder automatically resynchronizes every packet.

4 Compression Comparisons

Six pieces of software were used to compress image data and provide basis for comparing the performance of the various algorithms. Two executables used are readily available on UNIX systems: UNIX compress (LZW algorithm) and UNIX pack (an adaptive Huffman coding algorithm). The arithmetic coding software used in this comparison is from a 1987 publication in Communications of the ACM [6]. The JPEG software is from Stanford, and implements the 2 pass Huffman-based lossless JPEG standard [7]. The USE software simulates the USE chip, and likewise USES software simulates the USES chip. Only JPEG, USE, and USES support quantizations greater than 8 bits.

As a basis for comparing compression performance, three image suites were selected that provide a variety of image types and sizes. The first suite is composed of all 26 images from the University of Southern California (USC) Miscellaneous Monochrome Image Database. Image sizes range from 256x256 to 1024x1024. The second suite of images include 9 satellite images. The first six are 128x128 images of earth. The additional 3 images are simulated imagery based on aircraft images. Each of the three 512x350 images are spatially registered but are from different spectral bands. Although not from a satellite, they represent the type of scientific data that will be produced by a thermatic mapper instrument on a future satellite. The only non-eight bit suite is composed of seven magnetic resonance images (MRIs). The images are 12 bits, each 512x512.

As mentioned in the previous section, JPEG, USE, and USES include a pre-processor that removes redundancy before actually doing the coding. Without a pre-processor, UNIX compress, UNIX pack, and arithmetic coding software do not, in general, perform as well on images as the algorithms that incorporate a pre-processor. This is illustrated in Fig. 2. In order to compare the entropy coding sections of algorithms that don't have a pre-processor with those that do, the USES pre-processor was applied to the images before running compress, pack, or the arithmetic coding software. Except for Fig. 2, all results reflect the use of a pre-processor on the images either internally: as in JPEG, USE, and USES, or externally: compress (PCOMPR), pack (PPACK), arithmetic coding (PARITH). The 'P' is prepended to the name of the algorithm to indicate the use of the pre-processor.

Fig. 3 is a plot of compression results on the USC images after using a pre-processor on the data. Images 1 through 14 are 256x256 images. In general, USES out performed the other algorithms on the smaller images. An exception to this is image number 13 (eye-chart) which is an extremely low entropy image. UNIX compress achieved the highest compression on image 13. Compress did not compress as well as the other algorithms on high entropy images such as the Baboon (image 17). On the USC image suite, USES had the same overall performance as the pre-processor with adaptive Huffman coding (UNIX pack). Using the pre-processor with arithmetic coding resulted in slightly better overall compression results than the other algorithms for the larger images (512x512 and 1024x1024).

The compression results on the satellite images are shown in Fig. 4. In this suite of predominately smaller images USES was the overall winner.

The third image suite is seven 12 bit MRI images. Only USE, USES, and JPEG software was written to be able to compress data greater than 8 bits per pixel. USE and USES both achieved the same compression results. USES achieves higher levels of compression than USE under three conditions: 1) when the data is 8 bits or less, 2) when there are pockets of low entropy, 3) when there is high entropy data in which the quantization is 13 or 14 bits per pixel. Within the MRI images, none of the three conditions were met so USE achieved the same compression as USES. Fig. 5 shows the compression result on the MRI test data.

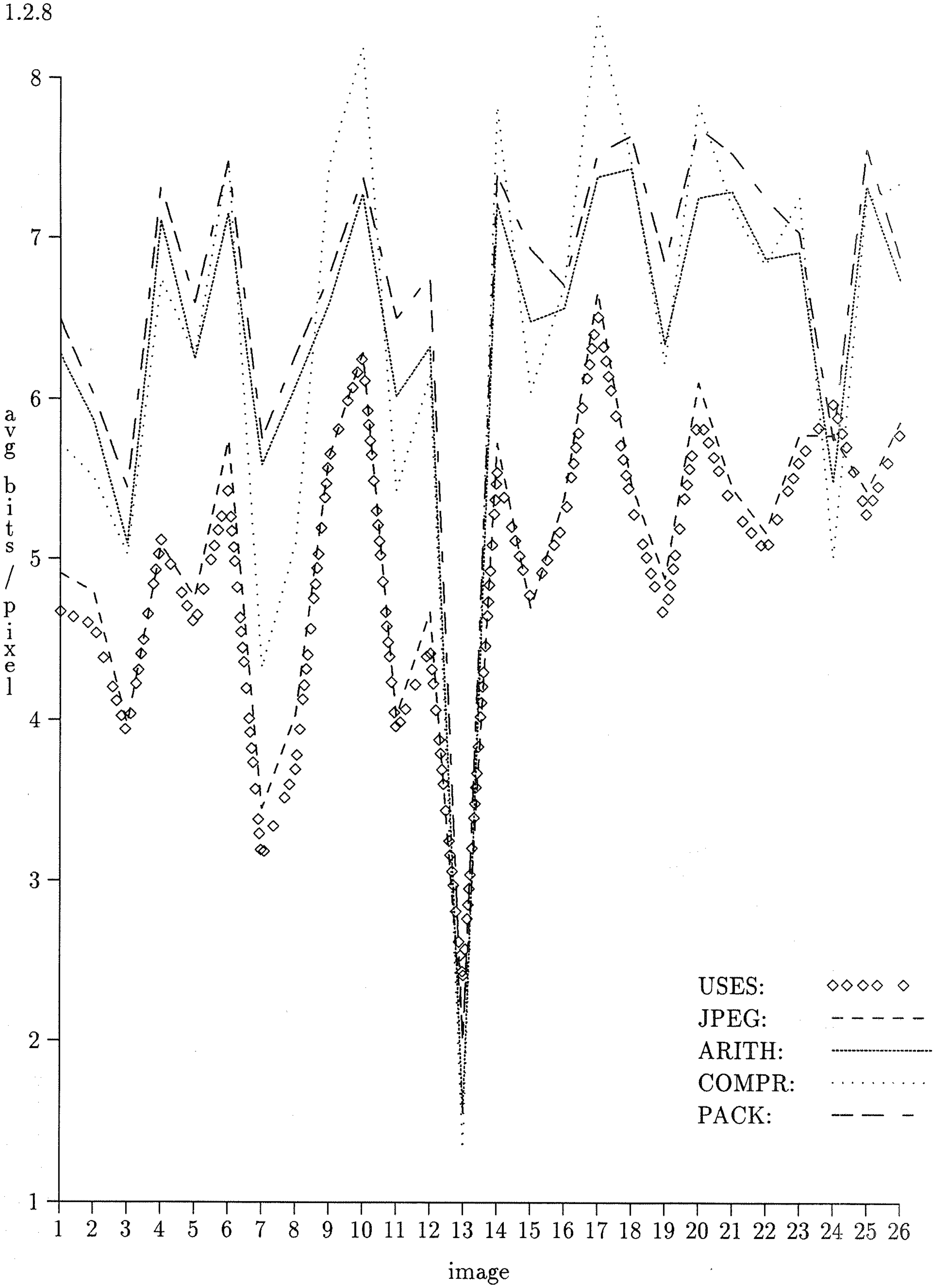


Figure 2: Average bits per pixel for all 26 images from USC miscellaneous monochrome image data base. All images are 8 bits but range from 256x256 to 1024x1024 in size. Results are without a pre-processor for arithmetic coding, LZW, and adaptive Huffman.

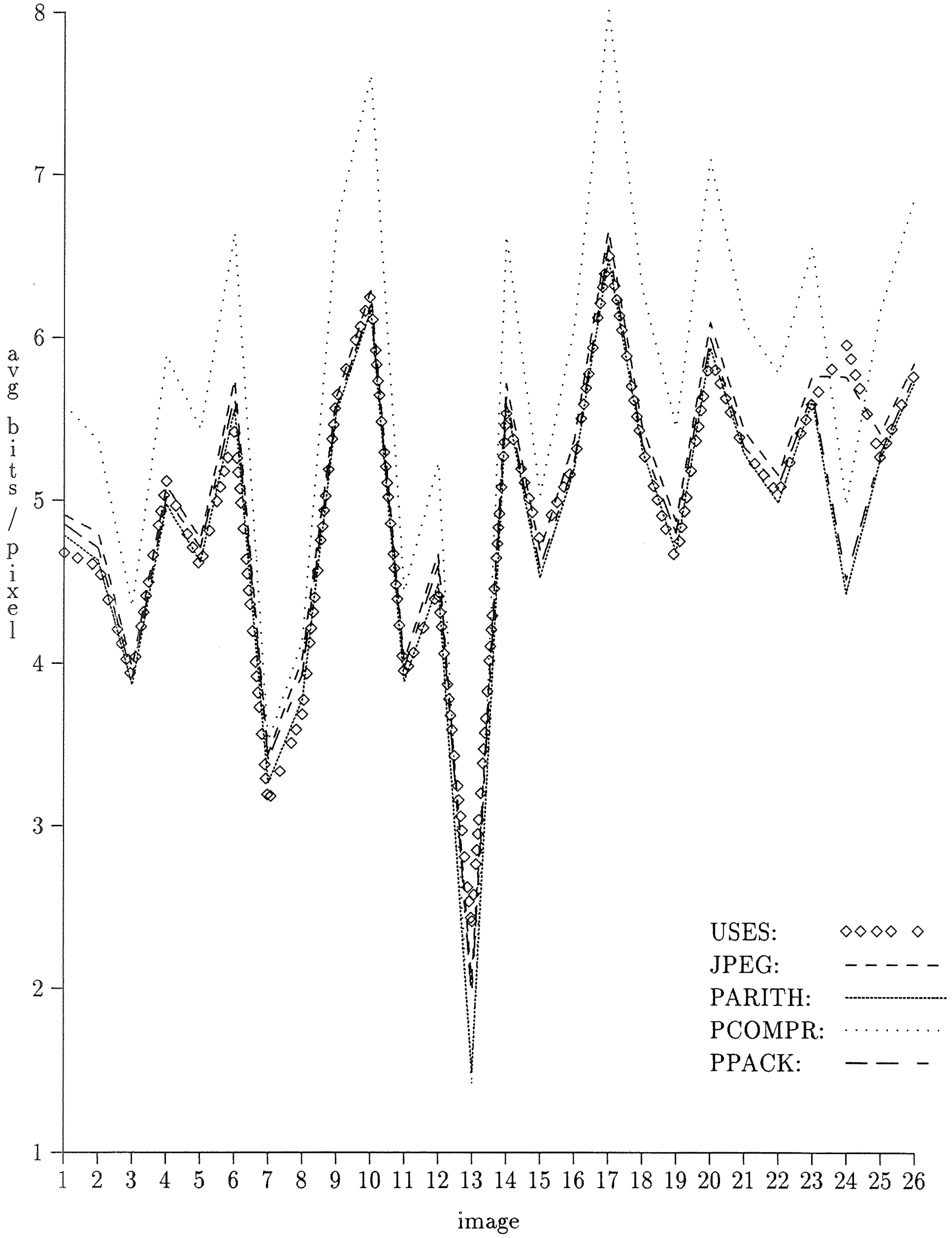


Figure 3: Average bits per pixel for all 26 images from USC miscellaneous monochrome image data base. All images are 8 bits but range from 256x256 to 1024x1024 in size.

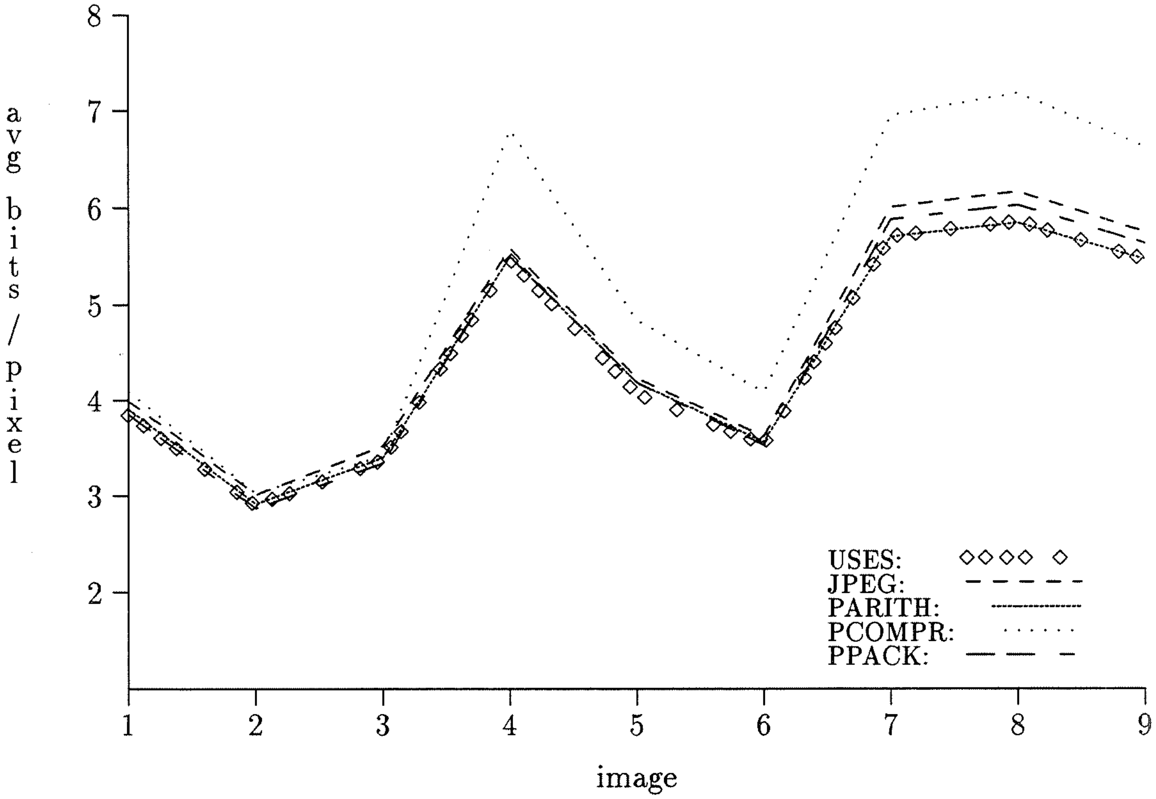


Figure 4: Eight bit satellite images of earth.

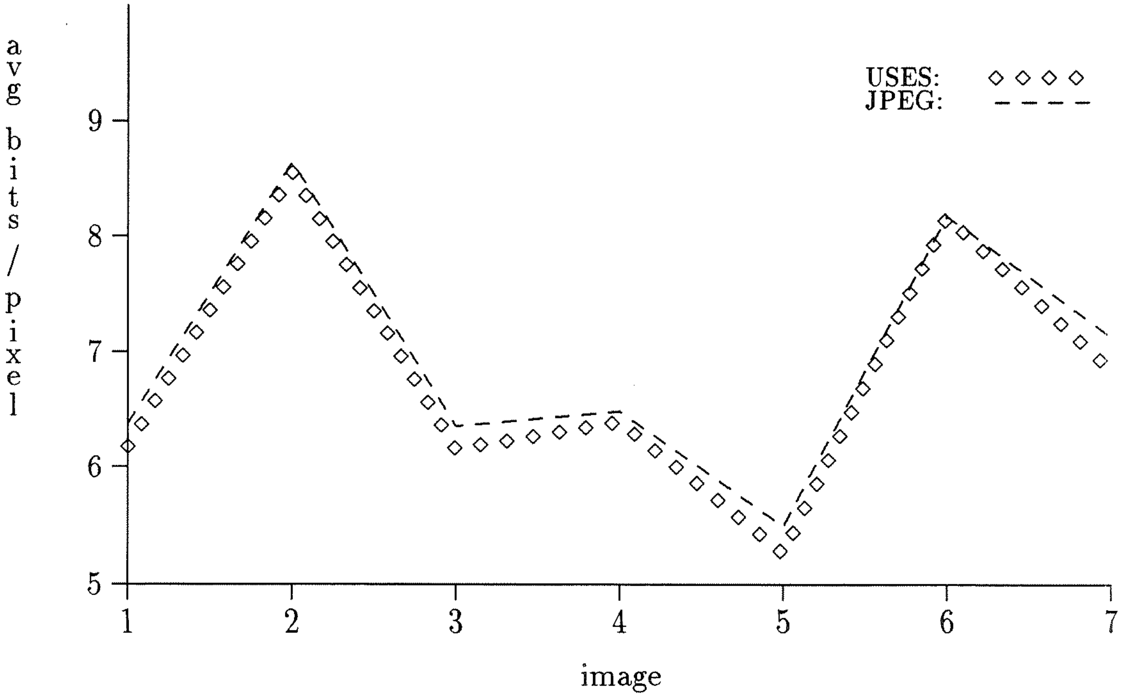


Figure 5: Comparing Huffman based JPEG and USES chip on 12 bit MRI data.

Table 2 shows a compression summary. For each image, the average bits per pixel was calculated after compression using each of the test algorithms. The data in the table is the average of those values. The lowest (winning) average is highlighted in bold face type.

Suite	p+LZW	p+aHuf	p+Arith	JPEG	USE	USES
satellite	5.23	4.54	4.51	4.66	4.54	**4.47**
USC monochrome	5.67	4.94	**4.85**	5.06	5.04	4.95
MRI 12bit	-	-	-	6.95	**6.79**	**6.79**

Table 2: An average of the average bits per pixel for each image suite. USES pre-processor was applied to images before coding LZW, adaptive Huffman, and Arithmetic coding to allow comparisons with algorithms having a built in pre-processor: JPEG, USE, and USES.

5 VLSI Implementations

The first chip set that employs the Rice algorithm was fabricated in 1991 and has since become a commercially available product [8]. A second generation Rice algorithm based encoder has been designed and is being fabricated in 1993 as a space qualified gate-array. The new encoder has been baselined for the LandSat 7 satellite project.

5.1 The First Chip Set

The first Rice chip set was fabricated in a 1 micron CMOS process. It uses a cell-based, structured-custom approach. In such an approach, engineers make specific effort to analyze data flow while developing the chip architecture. Layout technicians often abut the cells to minimize the amount of interconnect. Because of the structure of the Fundamental Sequence, no look-up tables were needed on either chip. The combined memory on both chips is only 340 bytes. The die size for each is 5 mm on a side, but neither chip fills up its core area.

5.1.1 Universal Source Encoder

The encoder's chip name is "USE", an acronym which stands for Universal Source Encoder. USE has a fully pipelined architecture; after 64 clock cycles, data is available for output. It can continuously accept new data and process every sample in one clock cycle. The USE architecture closely follows the Rice architecture diagram depicted in Fig. 1. A more detailed block diagram of the chip implementation is shown in Fig. 6.

The chip architecture and the generic Rice architecture differ in three ways. First, on the chip, differential pulse code modulation (DPCM) and mapper functions were implemented in one hardware block. Second, the winner select section was broken up into two sections for the chip: the bit counts section and the winner select section. The bit counts section is the largest single block on the chip. It calculates the number of bits that will be required to code the input block (16 samples) for each of the 12 coding options. The winner select

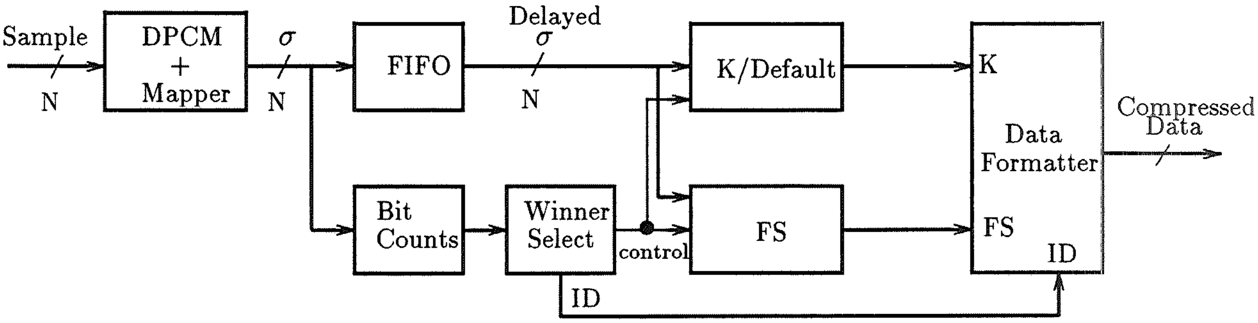


Figure 6: USE architecture.

block selects which of the options will produce the smallest number of coded bits and also perform calculations needed for the coding sections. The final difference between the VLSI implementation and the generic architecture is sharing of hardware among 12 coding options: only 2 hardware blocks are necessary (FS and K/Default). Control is distributed throughout the chip, but is mainly found in the input control section "InCntrl" and the two coding blocks.

The total amount of memory on the USE chip is only 130 bytes. The largest memory is the FIFO shown on the block diagram, 56 bytes, that holds the sigma values while the bit counts section calculates how many bits each of the options will take to encode the block. The K/Default section consists of a barrel shifter and control section that performs the packing of k split bits (or default data) into an output word size of 16 bits.

The data formatter concatenates the split bits with FS coded bits. It also prepends the 4 ID bits that specify which code option performed the coding. The output of the chip is a 16 bit parallel output with an asserted signal to flag the presence of data. The design reflects the requirements from NASA to output a contiguous block of compressed data that begins with a header word to specify the length of the block. An external packetizer should be used with the USE chip for concatenating the specified number of blocks to form a large data packet.

5.1.2 Universal Source Decoder

The decoder's chip name is "USD", for Universal Source Decoder. USD is a self timed circuit, requesting data from an external FIFO whenever it is ready for it. It requires no more than 2 clock cycles to decode a sample. The USD block diagram is shown in Fig. 7. Even though USD has about 3,000 fewer transistors than the USE chip, it is a more complex chip to design. The speed bottleneck is associated with the difficulty in decoding a serial bit stream. Even though the decoder input is 8-bit parallel data, it must decode the bit stream packed together by the encoder (and packetizer) one bit at a time. The start of the next block is not known until the present block is decoded. Therefore, it was difficult to design a decoder that would require only 1 clock cycle for decoding a sample.

The main decoding section includes 3 pipelines. If a block is coded by FS and k splits, then the FS data is routed to the FS pipeline for decoding and the k splits are routed to the k split pipeline. Likewise, if the block of data is default, the data would be routed to the default pipeline.

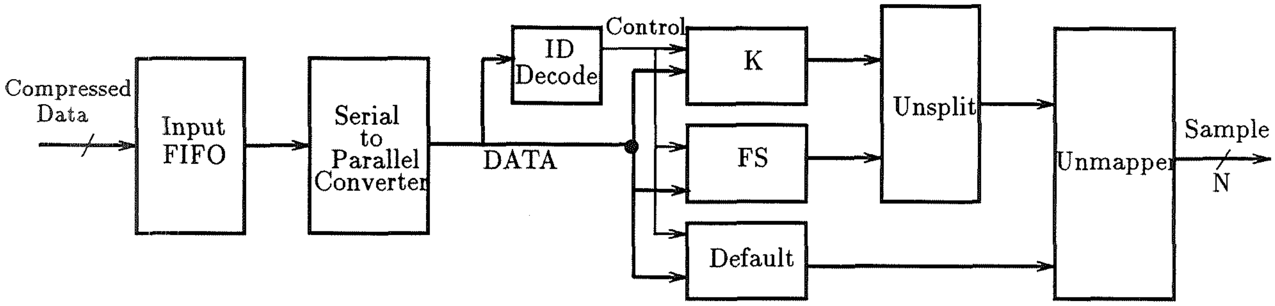


Figure 7: USD architecture.

Decoding a block of data includes the following.

- If a block begins with a reference sample, the reference must be removed from the bit stream.
- Decoding the ID bits in order to direct data to the pipeline that is to decode the data.
- Parsing the FS bits until the end is located. These bits are then processed by the FS section.
- Routing any k split bits to the k split pipeline.

The input to USD consists of a 66 byte FIFO. Data is stored in the FIFO until a custom serial to parallel converter is ready to parse new data. Within each of the three decoding pipelines is a small FIFO for storing data, bringing the total on board RAM to 210 bytes. The unsplitter concatenates the k split bits with the decoded FS data to form a sigma value. Default data as well as concatenated sigma data are then sent to the unmapper. It performs the reverse of the mapping function along with the reverse of DPCM. The decoded data is available at the output of the unmapper section.

The decoding pipeline is synchronized by a central control. Small localized control sections are associated with each of the pipelined stages: the input, the decoding, the unsplitter, and the unmapper. The decoder will handle data that is continuous or in packets. For satellite transmission, data is usually sent in packets. When an error occurs within a packet, the decoder is designed so that errors will not propagate to the next packet.

5.2 Universal Source Encoder for Space

The second generation Rice based encoder is called USES, an acronym that stands for Universal Source Encoder for Space. It will operate at 20 Msamples/s MIL spec and contains approximately 30,000 gates. Although it has only 88 I/O pins it will be packaged in a 196 pin flat-pack. (This is due to the limited number of packages from the gate-array vendor when using their double metal process.) The SEU immunity of the design, due to the foundry's cells and fabrication techniques, is expected to have an LET of close to 40 and be latchup resistant. The SEU and SEL results have been demonstrated on another gate array chip designed for NASA in the same process [13]. USES incorporates a number of new features not found in the USE chip. The new features include coding improvements as well as enhancements that make it easier to use in a system.

5.2.1 USES Coding Enhancements

The biggest coding enhancement found in USES that is not found in USE, is the addition of a very low entropy coder. The coder combines two options that come into play when the source entropy is less than 1 bit/sample. One option covers the range of 0.5 to 1.0 bits/sample. The other option run-length codes blocks of zero entropy data.

USES also has two new prediction modes not found in the USE chip. Simple two dimensional prediction is supported in USES. The prediction is the average of the previous sample with the data on the XP bus. Typically the value placed on the XP bus is the corresponding sample from the previous line. The compression gain found from using two-dimensional prediction may be up to 5%. The other prediction mode is a new technique developed at Goddard Space Flight Center for compressing multispectral data. The images must be spatially registered but may come from different sensors. Even though the sensors will have different gains and offsets, data from one spectral band can be used to help the prediction for another spectral band. Using this new technique, compression performance can be improved by 5 to 30%.

Other coding improvements incorporated into the USES design include the ability to compress reference data. Instead of sending a reference every line, USES can be put into a mode in which only 1 reference is sent at the beginning of a packet and all other references are compressed. USES also uses only 3 ID bits at the start of every block when the data is quantized to 8 bits or less. The USE chip always outputs 4 bits of ID.

5.2.2 USES System Improvements

The biggest improvement found in the USES design for system designers is data packetization. USE requires an external packetizer to concatenate blocks of data. The data packets referred to here, are not NASA CCSDS packets but rather concatenated bits of compressed data. Only at the end of a packet can there be any fill bits to fill out the packet to a 16 bit word. The data packet size may be set to a certain number of samples, or be as long as desired and ended in an asynchronous fashion whenever the EndPac pin is pulsed high.

Another enhancement to USES is that the design supports twos complement integers as well as positive integers. USES also supports block sizes of 8 and 10 samples in addition to the typical 16 samples per block. The USE design supported quantizations of 4 to 14 bits/sample while USES adds support up to 15 bits/sample. USES output data bus is tri-statable to simplify it's integration into systems that make use of shared buses.

Table 3 provides a comparison between USE and USES chips.

6 Summary

The Rice algorithm was compared with LZW, adaptive Huffman coding, and adaptive Arithmetic coding- all without a pre-processor. The Rice algorithm, with the added enhancements found in the USES chip, achieved the highest compression in 24 out of the 26 images in the USC miscellaneous monochrome images. Using the same pre-processor as is used in the USES chip, the enhanced Rice algorithm had a performance comparable to adaptive Huffman coding. USES, however, outperformed all the algorithms on small images- indicative

Metric	USE	USES
Foundry Type	Commercial	RAD Hard, Level S parts
Technology	1.0uM custom	1.2 uM Gate Array
Package Type	84 Pin PLCC	196 Pin flat-pack w/88 I/O
Pads	TTL compatible	CMOS, with tri-state outputs
Rapidly Adaptive	YES	YES
Quantization	4-14	4-15 bits/sample
Entropy Coverage	1.5-12.5	0-14.5 bits/sample
Prediction Modes	near. neighbor+external	also 2D, Multi-spectral
MSamples/Sec	approx. 25 at MIL spec	20 at MIL spec
Variable ID	fixed 4bit	3bit ID if 8bits or less
Input Data	positive only	positive or twos complement
Block Sizes	16	8, 10, 16 samples
Output Format	individual blks	concatenated packets
Entropy Coding	YES	YES
Packetizing Mode	NO	Yes. User selects packet size
Ref-to-Ref diff.	NO	Can Compress references
Tristatable Outputs	NO	YES

Table 3: Comparisons between USE and USES chips.

of the Rice algorithm's rapid adaptation to scene statistics. The enhanced algorithm also achieved greater compression than the two pass Huffman based JPEG. This was true even when compressing 12 bit MRI data.

LZW, as implemented in UNIX compress, achieved the best compression of all the algorithms on low entropy images such as the USC eye-chart. Conversely, compress demonstrated the lowest compression of all the algorithms on high entropy data such as the USC Baboon image. Without a pre-processor, arithmetic coding's compression results were not as good as JPEG or USE or USES. Pre-processing the images before arithmetic coding produced the best overall performance on the 26 USC images.

The Rice algorithm does not use a history of strings or statistics, it operates on a block by block basis. This characteristic of not operating on statistics of past data results in the implementations not requiring external memory and allowing the Rice algorithm to be used for in operating on small packets of data. If a data packet should be lost, the Rice-based encoder and decoder could continue to operate without needing to reset the system.

The first Rice algorithm based chip set was described as well as a second generation encoder which adds new low entropy coding options and on chip packetizing. The first chip set has become a commercial product and the new encoder has been baselined for LandSat 7 satellite project.

7 Acknowledgement

The authors wish to thank G. Maki, director of the Microelectronics Research Center, for helping to interface with the various groups interested in this project; and R.F. Rice for encouragement and suggestions over the past few years. L. Miles and E. Cameron provided some important software during the course of the USES project and deserve recognition for their contributions.

References

[1] R. F. Rice, Pen-Shu Yeh, and W. H. Miller, "Algorithms for a Very High Speed Universal Noiseless Coding Module", JPL Publication 91-1, Jet Propulsion Laboratory, JPL Publication Office, Mail Stop 111-130, Pasadena, California 91109, Feb. 15, 1991.

[2] R. F. Rice, "Some Practical Universal Noiseless Coding Techniques", JPL Publication 79-22, Jet Propulsion Laboratory, JPL Publication Office, Mail Stop 111-130, Pasadena, California 91109, March 15, 1979.

[3] R. F. Rice, "Practical Universal Noiseless Coding", 1979 SPIE Symposium Proceedings, Vol. 207, San Diego, California, August 1979.

[4] R. F. Rice and J. R. Plaunt, "Adaptive Variable Length Coding for Efficient Compression of Spacecraft Television Data," IEEE Trans. on Communication Technology, Vol. COM-19, Part I, pp. 889-897, Dec. 1971.

[5] Welch, T.A. "A Technique for High-Performance Data Compression", IEEE Computer vol 17, no. 6, pp.8-19, June 1984.

[6] Ian H. Witten, Radford M. Neal, and John G. Cleary, "Arithmetic coding for Data Compression", Communications of the ACM, June 1987, Vol. 30, Number 6, pp. 520-540.

[7] Joint Photographic Experts Group, an ISO (International Organization for Standardization) technical group. The specific document is X3L2.8, dated Aug. 1990. Available from The American National Standard Institute, X3 Secretariat: Computer and Business Equipment manufacturers Association, 311 First Street NW, Suite 500, Washington, DC 20001-2178.

[8] Chip No. AHA3370 and AHA3371, Advanced Hardware Architectures, NE 2365 Hoplins Court, Pullman, WA 99163.

[9] Pen-Shu Yeh, R. F. Rice, and W. H. Miller, "On the Optimality of Code Options for a Universal Noiseless Coder", JPL Publication 91-2, Jet Propulsion Laboratory, JPL Publication Office, Mail Stop 111-130, Pasadena, California 91109, Feb. 15, 1991. Revised in 1993.

[10] I. A. Shah, O. Akiwumi-Assani, and B. Johnson, "A Chip Set for Lossless Image Compression", IEEE J. Solid-State Circuits, vol 26, pp. 237-244, March 1991.

[11] I. A. Shah, O. Akiwumi-Assani, and B. Johnson, "A Chip Set for Lossless Image Compression", Proc. Custom Integrated Circuits Conf., Boston, MA, May 13-16, 1990, paper 17.7.

[12] Chip No. AHA3101, Advanced Hardware Architectures, NE 2365 Hoplins Court, Pullman, WA 99163.

[13] K. A. LaBel *et.al.* "Heavy Ion Results for Electronic Devices". In Workshop Record of 1993 IEEE Radiation Effects Data Workshop.

Avoiding and Tolerating Latency in Large-Scale Next-Generation Shared-Memory Multiprocessors [1]

David K. Probst
Department of Computer Science
Concordia University
1455 de Maisonneuve Blvd. West
Montreal, Quebec H3G 1M8
probst@macgyver.crim.ca and probst@vlsi.concordia.ca

Abstract - **A scalable solution to the memory-latency problem is necessary to prevent the large latencies of synchronization and memory operations inherent in large-scale shared-memory multiprocessors from reducing high performance. We distinguish latency avoidance and latency tolerance. Latency is *avoided* when data is brought to nearby locales for future reference. Latency is *tolerated* when references are overlapped with other computation. Latency-avoiding locales include: (i) processor registers, (ii) data caches used temporally, and (iii) nearby memory modules. Tolerating communication latency requires parallelism, allowing the overlap of communication and computation. Latency-tolerating techniques include: (i) vector pipelining, (ii) data caches used spatially, (iii) prefetching in various forms, and (iv) multithreading in various forms. Relaxing the consistency model permits increased use of avoidance and tolerance techniques. Each model is a mapping from the program text to sets of partial orders on program operations; it is a convention about which temporal precedences among program operations are necessary. Information about temporal locality and parallelism constrains the use of avoidance and tolerance techniques. Suitable architectural primitives and compiler technology are required to exploit the increased freedom to reorder and overlap operations in relaxed models.**

Keywords: large-scale multiprocessors, latency avoidance and tolerance, prefetching, multithreading, relaxed consistency, overlap of communication and computation, split-phase synchronization and memory operations.

1 Introduction

To meet the high-performance computing needs of national space and aerospace programs, computer architects and application designers must understand how parallel applications will scale to use large-scale next-generation shared-memory multiprocessors. One key requirement in solving "industrial strength" application problems with extremely large data sets are parallel machines with both (i) a scalable solution to the memory latency problem, and (ii)

[1] This research was supported in part by NSERC grants A3363 and MEF0040121, and by Micronet, A Network of Centres of Excellence.

scalable long-range communication bandwidth in the interconnection network. In large-scale machines, thousands of processor cycles (i.e., missed opportunities to issue floating-point instructions) may elapse while waiting for a high-latency synchronization or memory operation to complete. Although the speed of light will not increase, memory reference latency can be *avoided* by copying data to nearby locales, while both memory reference and synchronization operation latency can be *tolerated* by various techniques of overlapping communication and computation. One novel tolerance technique is the use of split-phase synchronization and memory operations. In this paper, we will survey a number of consistency models, and compare their support for avoidance and tolerance techniques.

A scalable solution to the memory-latency problem is necessary to prevent the large latencies of synchronization and memory operations—which are inherent in large-scale shared-memory multiprocessors—from reducing processor utilization. We distinguish latency avoidance and latency tolerance. Latency is *avoided* when data is brought to nearby locales for future reference. Latency is *tolerated* when references are overlapped with other computation. Latency-avoiding locales include: (i) processor registers, (ii) data caches used temporally, and (iii) nearby memory modules. Since latency avoidance requires temporal locality (i.e., repeated access of the same datum), the consistency model limits its use. Tolerating communication latency depends on the ability to overlap communication and computation. Latency-tolerating techniques include: (i) vector pipelining, (ii) data caches used spatially, (iii) prefetching in various forms, and (iv) multithreading in various forms. Since latency tolerance requires parallelism (i.e., the absence of ordering constraints), the consistency model limits its use. A consistency model is a mapping from the program text to sets of partial orders on program operations; it is a convention—agreed to by programmer and architect—as to which temporal precedences among program operations are necessary in any correct execution of the program [1,4,5,7,10]. Given other conventions about the proper use of synchronization in parallel programs, both temporal locality and parallelism are partly defined by the consistency model. Relaxing the consistency model permits increased use of avoidance and tolerance techniques. Suitable architectural primitives and compiler technology are required to exploit the increased freedom to reorder and overlap operations in relaxed models.

Both (i) the multiprocessor consistency model, and (ii) the set of explicit split-phase operations in the program text, define a partial order on program operations, specifying restrictions placed on completion order during program execution. There is parallelism when order is absent, and there is temporal locality for memory references to the same datum that fall between two successive synchronization points. Latency is avoided by copying data nearby (e.g., into a processor register). The scope of safe residence of a datum in a latency-avoidance locale is defined by temporal locality. For example, the value in the register must be written back to memory to restore consistency before the next synchronization point. Latency is tolerated by overlapping (apparent) communication delay in a synchronization or memory operation with other computation. The scope of safe overlap of program operations is defined by parallelism. Split-phase operations may specify parallelism either (i) explicitly (e.g., a nonblocking read is issued early because it is *placed* early, and the arrival of data is tested before its use), or (ii) implicitly (e.g., a programmer-designated synchronization operation is *initiated* early by the compiler independently of its placement, but sufficiently in advance of the use of its result). A consistency model also specifies other parallelism that is implicit in the program text.

Latency-avoidance techniques scale if temporal locality increases with data-set size, and this locality can be exploited by human or automatic means. Latency-tolerance techniques scale if parallelism increases with data-set size, and this parallelism can be exploited by human or automatic means. Coping with latency requires a combination of: (i) various forms of prefetching, (ii) various forms of multithreading, and (iii) use of split-phase synchronization and memory operations. In coarse-grained multithreading, we run multiple instruction streams on each processor, and switch to another stream whenever the current stream is experiencing a long delay [2]. In fine-grained multithreading, the "something else we do while waiting" comes from the *same* instruction stream. In both cases, the goal is not having the processor stall when a stream waits for one of its operations to complete.

A consistency model is a tradeoff between (i) reasonable programmer effort, an intuitive programming model, and an effective reasoning technique, and (ii) machine performance, including manageable complexity in compiler and runtime system, and cost-effective architectural primitives for fine-grained *intrathread* ordering. This tradeoff is surprisingly difficult. We still badly need: (i) articulation of a shared high-level programming model, (ii) proper definition of the interface between programmer and compiler, (iii) enhanced compiler technology to deal with data dependences that are highly irregular and/or evolving at runtime, and (iv) cost-effective architectural primitives that smoothly integrate the asynchronous receipt of acknowledgment messages with normal control flow. Support for latency avoidance and tolerance is an important factor.

2 Latency-Avoidance and Latency-Tolerance Techniques

Figure 1 shows an abstract block diagram for a multithreaded, scalable, shared-memory multiprocessor. It illustrates different kinds of latency-avoidance locales. If memory access is physically nonuniform, then it is reasonable to distribute data memory modules more or less uniformly throughout the network. In this way, data may be placed in (prefetched to) nearby memory modules whenever possible, thereby reducing memory latency [2].

Both (i) processor registers and (ii) caches used temporally are latency-avoidance locales that require temporal locality. In contrast, caches used spatially (i.e., hardware-controlled prefetching through long cache lines) is a latency-tolerance technique that requires spatial locality. Since spatial locality is not guaranteed in multiprocessor applications, this technique may work well only for applications with slowly migrating working sets. A theory of working-set hierarchy appears in a recent study of resource distribution in large-scale cache-coherent NUMA architectures [20]. Operation buffers (in the simplest case, write buffers) are used in latency-tolerance techniques; they allow memory references to be overlapped. For example, a write buffer allows reads to be serviced while there are pending writes.

Vector pipelining is a latency-tolerance technique. Vector instructions use hardware well because they overlap address generation, memory access and operand computations. When there is regularly structured parallelism in an application, an optimizing compiler may be able to vectorize sequential code for vector pipelining. The goal is to combine functional pipelines and memory-access pipelines so that multiple combining operations can be issued

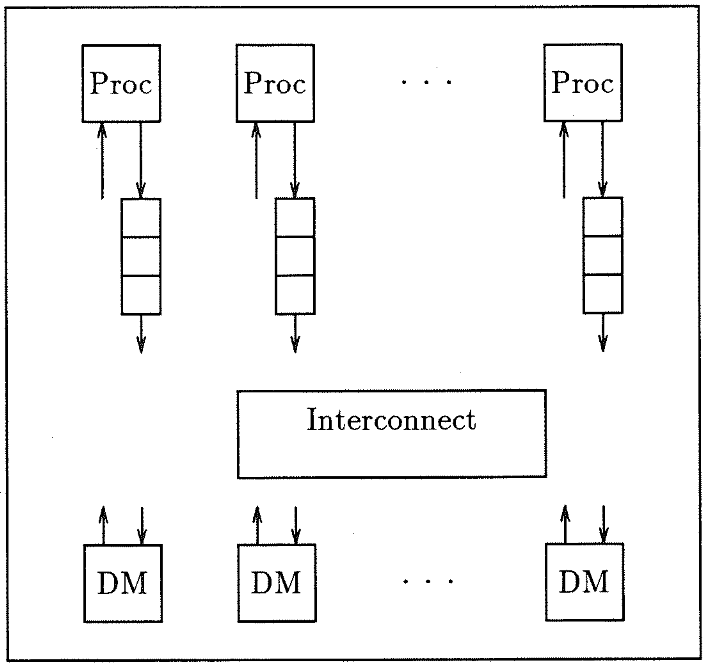


Figure 1: Abstract block diagram showing interconnection network, processor modules, data memory modules and operation buffers. Neither a "dancehall" nor a "boudoir" configuration is implied.

per clock cycle (more precisely, so that the operand-combining rate matches the instruction-issue rate). In vector machines, memory latencies are amortized over pipelined streams of data references. The set of vector registers replaces the set of data caches; the common objective is to improve locality and avoid reloading data. Since not all programs vectorize well, vector computing has an increasingly strong rival in scalar computing with instruction lookahead and good branch prediction. Vector machines also need a scalable mechanism to prevent the stream of values from being interrupted or rearranged by memory conflicts when a vector is moving between memory and a processor.

Multithreading is a family of latency-tolerance techniques with different granularities and different amounts of processor state (up to a maximum of one register set per thread). The common idea is rapidly switching control of the processor to a different thread whenever a high-latency operation is encountered. In *block multithreading*, the processor switches contexts only on synchronization faults and remote cache misses [14].

3 Relaxed Consistency Models

A multiprocessor consistency model specifies the set of allowable orderings of memory operations. More precisely, it specifies how the operations *appear* to execute, by defining the set of possible values that read operations may return. By analogy, in a uniprocessor, program operations appear to execute atomically (i.e., sequentially) in program order. In a uniprocessor implementation, memory operations may overlap, and may be issued and executed out of order as long as this appearance is maintained.

3.1 Sequential Consistency

In 1979, Lamport proposed the multiprocessor correctness condition sequential consistency (SC) [12]. We state a refinement—unconditional sequential consistency—of SC: even if arbitrary program operations *might* be synchronization operations, the result of any execution is the same as if the operations of all the instruction streams had been executed in some sequential order, and the operations of each instruction stream appear in this sequence in program order. The program behaves as if the memory accesses of all the threads were interleaved—respecting program order—and executed sequentially. The programmer *pretends* that program operations are atomic, i.e., happen in some definite order, and uses assertional reasoning, an effective invariant-based reasoning technique. Since atomicity is asserted at the granularity of single words of memory, there is no commitment to a high-level programming model (SC's greatest strength and weakness).

Relaxed consistency models, such as weak consistency and release consistency, were introduced because (unconditional) sequential consistency *appears* to place unacceptable restrictions on the allowable orderings of shared memory accesses within the same thread. Certainly, waiting for each stream operation to complete before issuing the next stream operation is too expensive [5,6]. Relaxed models require appropriate identification of synchronization operations. Significant overlap of memory accesses is allowed, and sequentially-consistent execution is guaranteed for certain restricted classes of parallel programs [9]. We focus on support for latency avoidance and latency tolerance. We analyze each consistency model as a partial order. Some partial orders are strictly stronger (i.e., have more order) than others. The weakest partial order provides the greatest scope for latency avoidance and latency tolerance because it admits the most parallelism and/or has the loosest synchronization skeleton.

3.2 Weak Consistency

Weak consistency (WC) views each sequential thread in a parallel program as an implicit partial order [5]. In WC, the programmer identifies synchronizing operations. We state implementation conditions for WC.

(*i*) No processor may issue a shared-variable access until all the synchronizing operations that precede the shared-variable access in program order have completed.

(*ii*) No processor may issue a synchronizing operation until all the shared-variable accesses that precede the synchronizing operation in program order have completed.

(*iii*) Synchronizing operations must be sequentially consistent with respect to each other.

In this model, synchronizing operation serve as memory *fences*. When a synchronizing operation completes, all previous operations have completed, and no future operation has been issued. Synchronizing operations define synchronization points at which consistency among processor views is guaranteed. Between synchronization points, freedom to reorder and overlap operations is allowed as long as uniprocessor control and data dependences are respected. This freedom defines the exploitable concurrency in WC.

3.3 Release Consistency

Release consistency (RC) weakens the partial order of WC [7]. In RC, the programmer distinguishes *acquire* and *release* operations. We state implementation conditions for RC.

(*i*) No processor may issue a shared-variable access until all the acquire operations that precede the shared-variable access in program order have completed.

(*ii*) No processor may issue a release operation until all the shared-variable accesses that precede the release operation in program order have completed.

(*iii*) Special accesses (synchronizing operations and nsynchs) must be sequentially (variant: processor) consistent with respect to each other.

RC partial order is strictly weaker than WC partial order because some precedence arcs between data and synchronization accesses have been deleted [7]. We say that WC is a proper *augment* of RC [15]. That is, RC has more exploitable concurrency than WC.

3.4 Protect Consistency

Protect consistency (PRC) weakens the partial order of RC [17]. In PRC, the programmer distinguishes: (i) P-type synchronizing operations (i.e., those that potentially block this thread), (ii) V-type synchronizing operations (i.e., those that potentially unblock some other thread), and (iii) PV-type synchronizing operations (i.e., those that potentially both block this thread and unblock some other thread). Within each thread, he identifies the necessary temporal precedences among synchronizing and ordinary program operations. Using a programming notation, he declares the necessary: (i) *synch* → *data* precedences, (ii) *data* → *synch* precedences, and (iii) *synch* → *synch* precedences. These precedences are not defined by program order. The compiler adds the necessary *data* → *data* precedences, completing a partial order for this thread. In addition, the programmer designates each P-type synchronizing operation as participating in either a static or a dynamic dependence. We state implementations conditions for PRC (cf. [4]).

(*i*) No processor may issue a shared-variable access until all the P-type operations protecting that memory access—that is, all the P-type operations that precede the memory access in thread partial order—have completed.

(*ii*) No processor may issue a V-type operation until all the shared-variable accesses protected by that V-type operation–that is, all the memory accesses that precede the V-type operation in thread partial order—have (at least locally) completed.

(*iii*) No processor may issue a PV-type operation until all antecedent shared-variable accesses have (at least locally) completed, and no processor may issue a descendant shared-variable access until the PV-type operation has completed, where "antecedent" and "descendant" are defined by thread partial order.

In WC and RC, the scope of synchronizing operations, i.e., which portion of the code is protected, is defined by relative position. In PRC, scope is defined by a programming notation—not relative position. Both WC and RC order are strictly stronger than intended order, i.e., the order required to support the program semantics. PRC order is strictly weaker than RC order because more precedence arcs have been deleted. For example, an ordinary access is delayed only for a designated *subset* of P-type operations that precede it in program order. RC is a proper augment of PRC. That is, PRC has more exploitable concurrency than RC.

Consider the (possibly unrealizable) performance-optimal consistency model—thread partial order (TPO)—in which the programmer specifies the necessary temporal precedences among program operations that must appear to be observed in *any* correct execution of the program [17]. Deletion of *any* precedence in the transitively-reduced partial order would admit at least one schedule of program operations that violates correctness. The correctness condition associated with TPO is nonatomic consistency (NC) [18].

Programmer specification of thread partial order is a difficult task, suitable only for critical program fragments [13]. Still, the programmer can specify the synchronization skeleton in each thread by declaring the *protects* relation between synchronizing and ordinary operations. This task is facilitated by an invariant-based view of necessary precedence (true dependence). When does operation b in thread t *depend on* operation a in thread s? Answer: when operation a makes invariant I true in s's view, and operation b's correctness requires that invariant I be true in t's view. This indicates how synchronization should be used. By convention, we have: (i) $P \rightarrow b$ only if P *awaits* I and $\{I\}$ b, and (ii) $a \rightarrow V$ only if V *notifies* I and a $\{I\}$. The solid arrows are elements of the protects relation, and the invariants inside curly brackets are guards and post-conditions, respectively.

Like SC, NC provides a legal explanatory account of all the values read; unlike SC, there is *no* presupposition of atomicity (i.e., absence of data races). We say that a global partial order on program operations—together with their arguments—is *legal* if it is possible to augment the partial order with a reads-from relation without introducing cycles, where there is no need to suppose that program writes have occurred in some definite order. As a multiprocessor correctness condition, *nonatomic consistency* (NC) states that:

NC: The result of any execution is the same as if the operations of all the partially-ordered instruction streams had been executed in some *legal* global partial order, and the operations of each instruction stream appear in this global partial order in (possibly augmented) thread partial order.

4 Instruction-Stream Multithreading

Instruction-stream multithreading using multiple hardware contexts is a standard technique for tolerating memory and synchronization latency. In Tera, each processor can execute multiple instruction streams simultaneously [2]. Up to 128 program counters may be active at one time. On every clock tick, the processor logic selects a ready stream, and allows it to issue its next instruction. Since instruction interpretation is completely pipelined by processor, network and memories, a new instruction from any stream can be issued safely during each cycle. Without overlap of instructions from the same stream, many streams per processor would be required to achieve peak performance. In Tera, explicit-dependence lookahead allows streams to issue multiple instructions in parallel. As multiprocessors scale in size, thread granularity will decrease to satisfy requirements for more parallelism, and frequency of synchronization will increase. Efficiency is critical. Two fundamental waiting mechanisms are *polling* (i.e., a thread repeatedly tests a synchronization condition until it becomes true), and *signaling* (i.e., a thread suspends itself, and is notified when the synchronization condition becomes true) [14]. In two-phase waiting, a thread polls up to some limit, and then suspends itself. At this point, a heavier-weight synchronization mechanism that avoids busy waiting is used. In a multithreaded processor, a polling thread can switch rapidly in round-robin fashion among processor-resident threads, overlapping its polling with useful computation by other threads [14]. No hardware context is loaded or unloaded during this switching.

5 Overlapping Instructions from the Same Thread

The technique of overlapping instructions from the same instruction stream tolerates latency without incurring the expense of context switching. The consistency model—no matter which one—defines a partial order for each thread that places restrictions on the completion order of memory accesses. This partial order is encoded during compilation, and decoded during program execution. A completely unrestricted partial order requires more encoding, and more decoding; this gives rise to practical limitations. Abstractly, each instruction i in a partially-ordered instruction stream has a finite number of *predecessor* instructions upon which it depends. Instruction i is delayed until all the instructions in its preset have completed. One implementation is as follows: Each read or write operation is issued by the processor in a nonblocking fashion, and contains a tag identifying instruction i—or at least a synchronization node controlling i's release—that causes the acknowledgment message to be returned to the proper place. The compiler knows the size of i's preset. When all acknowledgments have been received, a join operation releases instruction i. Within its time quantum, a thread runs until it suspends. At any time, a number of instructions will be concurrently enabled. Optimal dynamic scheduling selects the instruction that maximizes overlap, and minimizes time to completion. In practice, a set of scheduling heuristics must be used.

A given serial encoding of the partial order restricts the amount of parallelism in the thread that is visible to the processor. In Tera's explicit-dependence lookahead, a new instruction waits for the completion of all instructions with lookahead values referring to the new instruction [2]. These predecessor instructions live in a code window of size 8. Hence, at most eight instructions from a stream can be executing concurrently. In general, processor resource requirements increase (i) as allowable concurrency increases, and (ii) as the number of potentially waiting instructions increases.

If the processor architecture is not aggressive (i.e., not superscalar), then issue order—but not completion order—is largely determined by compiler scheduling. When processors block on read accesses, write latency can still be effectively tolerated as long as write buffer and data caches allow reads to be serviced while there are pending writes [8]. Tolerating read latency—once reads have been made nonblocking—is more difficult with conventional program design, since the computation that follows a read typically requires the return value. Unless there is sophisticated compiler scheduling during code generation, overlap is limited, since it is difficult for the processor to move past the use of the read to find independent computation (we do not consider superscalar processors).

The greatest scope for compiler scheduling is given by a partial order with the most parallelism and/or the loosest synchronization skeleton. To maximize overlap, the compiler should schedule read issues as early as possible, and write issues as late as possible. A well-specified *protects* relation minimizes delay of (i) reads by irrelevant P-type operations, and (ii) V-type operations by irrelevant writes. P-type operations that support dynamic dependences should also be issued early. More parallelism (i.e., less order) means that more operations from a thread (reads *and* writes) can be outstanding at one time.

5.1 Processors with Local Copies of their Views

Processors with caches and/or local memories can store part of their view nearby as a local copy. Processor i's *view* of location a is the value processor i may return on a read of a. Keeping copies of shared data nearby (i.e., caching) extends to any latency-avoidance locale. For example, there is register caching and nearby-memory caching. B.J. Smith defines prefetching as *any* operation issued out of program order that localizes a value [21]. Among locales, data caches are special in two respects: (i) they are invisible to the compiler, and (ii) they allow copies of concurrently writable data (cf. [4]). When there are copies of the view, local completion is enough to guard V-type operations. Augmented P-type operations on other processors can guarantee consistency [3]. If synchronizing variables are bound to the shared variables they protect, then synchronizing operations can be integrated with the remembering of updated values. Copies are sent from one processor to another only when there is a synchronization chain between them.

Figure 2 illustrates using local completion as a guard. Suppose that thread 1 has entered its critical section ahead of thread 2. At this point, there is a flow dependence from write(x) to read (x), and an antidependence from read(y) to write(y).

If there is no copy of the view, then both memory accesses in thread 1 must complete with respect to thread 2 before the V operation may be issued. Global acknowledgment is necessary. If there is a copy of the view, then the V operation may be issued as soon as both accesses have locally completed. When thread 2 issues its P operation, a further protocol delays completion until the new value of x has been sent from thread 1 to thread 2. The read of y has already locally completed in thread 1. This is conservative; due to dynamic scheduling of threads on processors, we do not know whether both threads happen to share the same copy of the view.

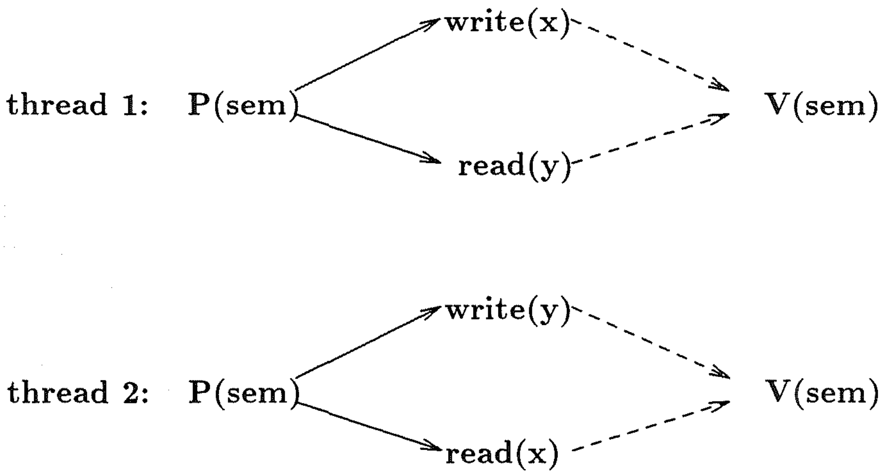


Figure 2: Thread partial orders with local completion.

5.2 Static versus Dynamic Dependence

Each synchronization chain $a\ \{I\} \rightarrow V \Longrightarrow P \rightarrow \{I\}\ b$ must be clearly marked as supporting either a *static* or a *dynamic* dependence. In a static dependence, we know *a priori* that invariant I will be false until operation a makes it true. In a dynamic dependence, which is best thought of as a generalized ownership protocol supporting (possibly exclusive) access, we do not know *a priori* that an "ownership conflict" exists. The resource may not currently be a source of conflict; it may be (i) not owned by any processor, or (ii) owned non-exclusively by one or more processors (if the requesting processor seeks non-exclusive ownership).

Figure 3 illustrates that designating synchronization chains as static or dynamic dependences can have performance implications. Suppose that the primitive hardware mechanisms for synchronization are *full/empty bits* and efficient traps [2,14]. We say that a synchronization attempt *succeeds* if it does not generate a trap, and *fails* otherwise. Since failures are signaled via traps, successful attempts are overhead free. Assume that synchronization attempts for static dependences and barrier synchronization fail with high probability (i.e., their synchronization latency far exceeds their communication latency). Also assume that synchronization attempts for dynamic dependences succeed with high probability (i.e., their synchronization latency equals their communication latency). Here, it makes sense to look for strategies in which communication latency can be completely hidden by being overlapped with other computation.

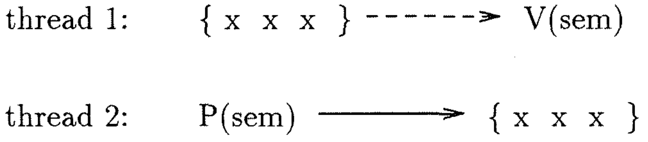


Figure 3: Either a static or a dynamic dependence.

Since Tera-style tagged variables with full/empty bits have been used to implement both critical sections and interthread static dependences, the performance consequences of making this distinction may not be immediately apparent. Synchronization can be on behalf of *cooperation* or *contention*. Cooperation is determinate in that we know *a priori* that the "produce" operation will precede the "consume" operation. For this reason, it is entirely reasonable to use tagged variables to implement static dependence; there is nothing cheaper. Contention is indeterminate in that *a priori* we know neither the outcome of contention (i.e., the grant order), nor whether contention currently exists. Having an ownership protocol run automatically at time of need may not be the cheapest solution. There may be performance benefits in running separate protocols for enforcing determinate and indeterminate precedence. In the latter case, we may choose to initiate communication directly in advance rather than have it happen indirectly at time of need. This allows us to *schedule* communication so that it overlaps the normal computation preceding entry to the critical section, giving us the latency tolerance of a split-phase transaction. By exposing communication to the compiler, we allow an ownership protocol to run in parallel with unrelated computation; in the best case of no contention, the effective latency of the "synchronizing load" has been reduced to *zero*.

6 Split-Phase Synchronization and Memory Operations

Split-phase synchronization and memory operations rely on the same principle: assuming the ability to overlap communication and computation, each communication must be initiated sufficiently in advance of the use of its result. With suitable primitive communication mechanisms, the responsibility for latency tolerance can be shifted to either compiler or programmer, depending on who has control over communication pipelining. Asynchronous communication is required in the sense that an instruction stream launches messages into the network and continues computing. The most significant distinction is the following: either (i) the programmer *places* the operation sufficiently early, and tests for its completion, or (ii) the compiler *initiates* the operation sufficiently early, and guarantees its completion. There is a further distinction as to whether completion is guaranteed by (i) an acknowledgment protocol, or (ii) delay assumptions. In this section, we discuss explicit split-phase memory operations whose completion is guaranteed by an acknowledgment protocol, and implicit split-phase synchronization operations whose completion is guaranteed by delay assumptions.

6.1 Split-Phase Synchronization Operations

Competitive multiprocessors *may* choose to leverage the huge, continuous engineering investments in hot sequential microprocessors. As a next step, they may also choose to use off-the-shelf network interfaces and switches developed by the ATM networks industry, which benefit from comparable investments [11]. The implications for next-generation parallel architectures in which the interconnect is essentially an ATM network are intriguing.

The significance of ATM interconnect is that reserved bandwidth virtual connections with guaranteed performance can be used as dedicated synchronization channels. In this setting, new ownership protocols—where correctness depends on real-time performance guarantees—are feasible. A thread can obtain a virtual connection with both a guaranteed bandwidth and a bounded latency for cell delivery. Cells traveling along reserved bandwidth virtual connections are called *constant bit rate* (CBR) traffic. Initially, a thread issues a request to the network to reserve a certain bandwidth and latency bound for a CBR virtual connection. Then, if the resources are available, the network reserves the required resources along a fixed path from source to destination. More generally, a thread may request a multidestination virtual connection for the purposes of reliable multicasting with guaranteed performance. The CBR performance guarantees are independent of load since they are implemented by a precomputed fixed schedule. In building the network, the buffer size is dependent only on the targeted virtual connection bandwidth. DRAM is already cheap enough to be cost effective in the gigabit range.

There is a well-known protocol due to Michael Fischer for real-time mutual exclusion. Starting at time 0, each competing process writes its name to variable x respecting an upper bound D for the write to complete; also, starting at time 0, each process reads some name from variable x respecting a lower bound d for the read to issue, where $D < d$. This guarantees that only one process can enter its critical section. This idea can be adapted to yield an efficient prologue for an ownership protocol. Suppose that process P wishes to enter its critical section. At time 0, P sends a reliable multicast with bounded latency to all potential rivals, where D is an upper bound on receiving an acknowledgment indicating interest in the critical section by a rival. P can both (i) conclude that no potential rival is interested, and (ii) enter the critical section without further protocol, as soon as d time units have elapsed, where $D < d$. Only if ownership conflict is detected is there any need to run a standard ownership protocol, quite possibly Fischer's protocol for real-time mutual exclusion.

Like any implementation of a synchronization primitive, this prologue—only briefly sketched above—must be formally verified. Techniques for specification and verification of real-time systems with partial orders may prove useful in this context [16,19].

6.2 Split-Phase Memory Operations

T. von Eicken and coworkers have proposed a simple coarse-grained programming model that provides explicit split-phase remote memory operations in the C programming language [22]. The Split-C model was designed to demonstrate the value of Active Messages, a primitive communication mechanism that smoothly integrates communication with computation. In Split-C, two split-phase operations are provided: PUT, which copies a local memory block into a remote memory, and GET, which retrieves a block of remote memory and makes a local copy. Both operations are nonblocking. Incrementing a separately specified completion flag on the processor that receives the data allows simple synchronization, e.g., by polling.

A simple matrix-multiply example $C = A$ x B is given in [22], where matrices are partitioned in blocks of columns across the processors. Each processor computes with its own columns of A and continues by getting the columns of the next processor. Communication and computation are overlapped by GETting the column for the next iteration while computing with the current column (the latency of the GET is balanced with the computation in the inner loop). Both remote memory access and its completion are explicit. Unrolling the loop gives: get(next); compute(current); await(next); swap(current, next);

etc. In this coarse-grained model, the programmer must explicitly overlap communication and computation. Since PUT and GET are nonblocking but general computation is not, the programmer of the matrix-multiply example has written a simple partially-ordered program in which each await(next) is a memory *fence* that separates one pair of concurrent {get(next), compute(current)} operations from the next.

In the Active Message model, each node has an ongoing computational task that is smoothly integrated with asynchronous message arrival. The scheduling policy allows large grains of computation to be enabled by the arrival of multiple messages. With this primitive communication mechanism, explicit split-phase (albeit coarse-grained) remote memory operations can be efficiently implemented. The upside is that the programmer can control the overlap of communication and computation; the downside is that the granularity is coarse, and the programmer effort is nontrivial.

7 Conclusion

High-performance computing in large-scale next-generation shared-memory multiprocessors depends on finding a scalable solution to the memory-latency problem, which includes both remote loads and synchronizing loads. Different consistency models offer different scopes for the use of latency-avoidance and latency-tolerance techniques because they make different assertions about the amount of temporal locality and parallelism that is available in a program. Temporal locality can be used to define regions where only one processor has a writable copy of a shared datum (latency avoidance). When copies are concurrently writable, a cache coherence protocol must be used. Parallelism maximizes the opportunities for instruction overlap (latency tolerance).

Switch-on-miss block multithreading is a powerful workhorse in multithreaded processors with multiple hardware contexts. Overlap of instructions from the same stream is allowed by (i) explicit split-phase memory operations, and (ii) implicit overlap of synchronizing and memory instructions as permitted by the consistency model. Protect consistency provides fine-grained parallelism and loose synchronization skeletons, increasing the number of operations that can be outstanding from a single thread. All of this should be combined with other forms of prefetching.

Dependence analysis works well in serial regions of parallel programs—as long as there is no interference from writes by competing threads—and is essential to all relaxed models, including PRC. The invariant-based view of programming facilitates specification of the protects relation. Fine-grained parallel programs are control intensive, so a lightweight *intrathread* order primitive is required for aggressive implementation.

We believe there is significant potential in ATM-like interconnect. The basic idea is to use virtual connections with guaranteed performance as synchronization channels. We have sketched a prologue—based on reliable multicast with bounded latency for cell delivery—to optimize an ownership protocol when synchronizing operations are labeled by their use. Significant performance advantages are obtained by overlapping (i) communication for "contention" synchronization, and (ii) unrelated computation, in the manner of a split-phase transaction.

References

[1] S. Adve and M. Hill, *Weak ordering – a new definition*, in Proc. 17th International Symposium on Computer Architecture, May 1990, pp. 2–14.

[2] R. Alverson et al., *The Tera Computer System*, in Proc. 1990 International Conference on Supercomputing, June 1990, pp. 1–6.

[3] B. Bershad and M. Zekauskas, *Midway: Shared-Memory Parallel Programming with Entry Consistency for Distributed-Memory Multiprocessors*, Department of Computer Science, Carnegie Mellon University, Report CMU-CS-91-170, April 1991.

[4] D. Callahan and B.J. Smith, *A Future-Based Parallel Language for a General-Purpose Highly-Parallel Computer*, in D. Gelernter et al., eds., Languages and Compilers for Parallel Computing, MIT Press, 1990, pp. 95–113.

[5] M. Dubois et al., *Memory Access Buffering in Multiprocessors*, in Proc. 13th International Symposium on Computer Architecture, June 1986, pp. 434–442.

[6] M. Dubois and C. Scheurich, *Memory Access Dependencies in Shared-Memory Multiprocessors*, IEEE Transactions on Software Engineering, **16:6**, June 1990, pp. 660–673.

[7] K. Gharachorloo et al., *Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors*, in Proc. 17th International Symposium on Computer Architecture, May 1990, pp. 15–26.

[8] K. Gharachorloo et al., *Performance Evaluation of Memory Consistency Models for Shared-Memory Multiprocessors*, in Proc. 4th International Conference on Architectural Support for Programming Languages and Operating Systems, April 1991, pp. 245–257.

[9] P. Gibbons et al., *Proving Sequential Consistency of High-Performance Shared Memories*, in Proc. 3rd ACM Symposium on Parallel Algorithms and Architectures, July 1991, pp. 292–303.

[10] P. Gibbons and M. Merritt, *Specifying Nonblocking Shared Memories*, in Proc. 4th ACM Symposium on Parallel Algorithms and Architectures, June 1992, pp. 306–315.

[11] H.T. Kung, *Some Emerging ATM Networks Will Make Distributed Computing Easier*, Keynote Address, 13th International Conference on Distributed Computing Systems, Pittsburgh, May 1993.

[12] L. Lamport, *How to Make a Multiprocessor Computer that Correctly Executes Multiprocess Programs*, IEEE Transactions on Computers, **28:9**, September 1979, pp. 690–691.

[13] L. Lamport, *How to Make a Correct Multiprocess Program Execute Correctly on a Multiprocessor*, DEC SRC Research Report 95, February 1993.

[14] B.-H. Lim and A. Agarwal, *Waiting Algorithms for Synchronization in Large-Scale Multiprocessors*, ACM Transactions on Computer Systems, **11:3**, August 1993, pp. 253–294.

[15] V.R. Pratt, *Modeling concurrency with partial orders*, Int. Journal of Parallel Prog., **15:1**, February 1986, pp. 33–71.

[16] D.K. Probst and H.F. Li, *Verifying Timed Behavior Automata with Nonbinary Delay Constraints*, in Proc. 4th Workshop on Computer-Aided Verification (CAV '92), G.v. Bochmann and D.K. Probst, eds., Lecture Notes in Computer Science 663, Springer-Verlag, 1993, pp. 123–136.

[17] D.K. Probst and H.F. Li, *A Simple Modern Correctness Condition for a Space-Based High-Performance Multiprocessor*, in Proc. 4th Nasa Symposium on VLSI Design, S. Whitaker, ed., Coeur d'Alene, Idaho, October 1992, pp. 6.4.1–6.4.10.

[18] D.K. Probst, *Efficient Implementation of Thread Partial Order for a Scalable Shared-Memory Multiprocessor*, in Third Workshop on Scalable Shared-Memory Multiprocessors, M. Dubois and S. Thakkar, orgs., in conjunction with 20th International Symposium on Computer Architecture, May 1993, San Diego, California.

[19] D.K. Probst and H.F. Li, *Verifying Timed Behavior Automata with Input/Output Critical Races*, in Proc. 5th Conference on Computer-Aided Verification (CAV '93), C. Courcoubetis, ed., Lecture Notes in Computer Science 697, Springer-Verlag, 1993, pp. 424–437.

[20] E. Rothberg et al., *Working Sets, Cache Sizes, and Node Granularity Issues for Large-Scale Multiprocessors*, in Proc. 20th International Symposium on Computer Architecture, May 1993, pp. 14–26.

[21] B.J. Smith, private communication, June 1993.

[22] T. von Eicken et al., *Active Messages: a Mechanism for Integrated Communication and Computation*, in Proc. 19th International Symposium on Computer Architecture, May 1992, pp. 256–266.

Session 2
Radiation Effects

Chairman: John Canaris

Rad-Hard Computer Elements for Space Applications

G. S. Krishnan
NASA Goddard Space Flight Center
Greenbelt MD

Carl D. Longerot and R. Keith Treece
Sandia National Laboratories
Albuquerque NM

Abstract - **Space Hardened CMOS computer elements emulating a commercial microcontroller and microprocessor family have been designed, fabricated, qualified, and delivered for a variety of space programs including NASA's multiple launch International Solar-Terrestrial Physics (ISTP) program, Mars Observer, and government and commercial communication satellites. Design techniques and radiation performance of the 1.25 micron feature size products are described.**

1 Introduction

Mars Observer, Magellan, Galileo, Polar Laboratory, Wind Laboratory, and other deep-space exploration vehicles place unusually stringent requirements on the integrated circuits (ICs) used in the electronic equipment aboard. Some of these requirements can only be met by radiation-hardened integrated circuits (RHICs), which can withstand the hostile radiation environments that exist in outer space, tolerate large total doses of radiation, and are immune to electrical upset when struck with high-energy ions or particles. This later feature, called single event upset (SEU) immunity, is especially important to ensure the integrity of the vital data collected during deep-space explorations.

Since the 1970's, Sandia has established cooperative relationships with various domestic semiconductor manufacturers, teaming with Intel, RCA, National Semiconductor Corporation, and Advanced Micro Devices to develop radiation-hardened integrated circuits that emulate several microprocessors, microcontrollers, and support chips. The semiconductor manufacturers, the customers, and Sandia have all benefited from this arrangement.

2 Component Types and Applications

Sandia has designed many large scale integrated circuits (LSICs) that have a high tolerance of space radiation environments including single event upset, high dose rate transients, total dose, and are latchup free. The devices have been fabricated, tested, and delivered for several satellite programs.

Some of the designs for earlier space missions are still in use; however, more complex device designs are being made available as the demand for more complex systems increase. The current Sandia designs include a 32-bit microprocessor (SA3300), timing control unit (TCU), floating point unit (FPU), interrupt control unit (ICU), glue chips (latch, I/O port, address decoder, and transceiver), and an 8-bit microcontroller (SA3865). The microprocessor, TCU, FPU, and ICU are rad-hard complementary metal oxide semiconductor (CMOS) emulations of National Semiconductor Corporation (NSC) devices (NS32C016, NS32C201, NS32381, & NS32202) and the microcontroller is a rad-hard emulation of the Intel 80C51.

The devices were successfully fabricated in the 1.25 micron AT&T CMOSV rad-hard process. For business reasons, AT&T chose to exit the rad-hard foundry business and United Technologies Microelectronics Center (UTMC) is now being qualified to fabricate our rad-hard products. Some of the devices have been delivered for space programs as 'custom screen' products which were fabricated, packaged, burned-in, electrically tested, and qualified as Class S like products per MIL-STD-883. Sandia also designs rad-hard custom and ASIC LSICs for weapon programs. LSI Logic has been qualified as a source for gate array ASICs and is presently providing parts for weapon systems. The rad-hard emulations of devices have been configured in some weapon systems to provide computer capability along with earlier rad-hard memory modules developed by Sandia in cooperation with commercial facilities.

Devices fabricated in the AT&T CMOSV process (SA3300, TCU, and ICU) have been delivered for applications in the NASA Goddard Space Flight Center's International Solar-Terrestrial Physics (ISTP) satellite program. The devices will be used in at least two of the programs missions: the WIND Laboratory (Figure 1) scheduled for launch in February 1994; and the POLAR Laboratory scheduled for launch in May 1994. The SA3300 was also delivered to the California Institute of Technology (Caltech) for use in the Mars Observer camera. This camera, developed by Caltech is expected to have the highest imaging resolution of any ever flown on a planetary mission. The Mars Observer spacecraft was launched September 25, 1992.

Figure 1: ISTP WIND Laboratory

The SA3865 microcontroller has been successfully fabricated in UTMC's 1.2 micron CMOS process and has been fully characterized by Sandia for rad-hard environments. The SA3865 is being qualified and delivered by Sandia to LORAL for use in a communication satellite application. The SA3300 is now being procured from UTMC for a Sandia weapon application and will be a full Class S product fabricated, packaged, screened, and qualified in the 1.2 micron CMOS process at UTMC. An ASIC containing an 8-bit microcontroller as an internal macro module, special interface and control logic, and timing and control features for use with the SA3300 is also being designed at Sandia. This ASIC will be fabricated in a LSI Logic rad-hard gate array technology and is scheduled for use in the same weapon application as the SA3300.

Sandia is presently investing resources in developing more complex rad-hard ASIC and multichip module (MCM) technologies with increased emphasis on higher density and high reliability requirements.

3 Methodology

Sandia uses its own network of more than 60 Mentor Apollo and Sun nodes and work stations with full LSIC design capability. Designers have all the necessary tools for both full custom and ASIC gate array LSIC designs including schematic capture, circuit and logic simulation, and layout along with the necessary design rule and connectivity verification capability. The work station network contains multiple libraries for design compatibility to a variety of commercial vendor processes and includes digital and analog capabilities.

Sandia also has a respectable electrical test capability with Sentry, Advantest, and HP IC testers for device prototype validation, characterization, and product screen and qualification. A radiation specialists group provides capability for full radiation environment testing at several facilities (Berkley, Brookhaven, and White Sands) while total dose (Cobalt 60) and some limited transient capability is maintained at Sandia. A Sandia owned Advanced MicroTechnology (AMT) computer controlled static and dynamic burn-in system is available for product screening and reliability studies. The system provides 128K by 104-bit wide dynamic test vector excitation and monitoring capability during dynamic burn-in at temperatures to 150 degrees C. These facilities provided the capability for screening and qualification of products for space applications.

A facility just outside the Sandia base gate has MIL-STD-883, Class B or Class S, Group B and Group D qualification capability where the Sandia burn-in equipment resides and which provides the convenience of integrating it with Sandia's electrical end point testing capability.

4 Effects of Radiation

Radiation-hardened microcircuits must be designed to address two types of ionizing radiation effects:

- *total dose damage,* which is cumulative and produced by x-ray and gamma ray sources
- *single event upsets,* which are produced by high energy ions created outside the atmosphere by solar flares and other cosmic phenomena.

The most harmful effects produced by x-rays and gamma rays occur in the oxide layers that are an essential part of all metal-oxide semiconductor devices. Fortunately, as designers make the device features smaller and smaller, the oxide layers can be thinner and, consequently, less susceptible to these harmful effects.

Unfortunately, devices become more susceptible to single event upsets as feature sizes become smaller. Dynamic devices that require electrical charges be replenished periodically are more susceptible to single event upsets than static devices, which can retain charge as long as they are suitably powered. Sandia's integrated-circuit designs are not only static, they also use polysilicon resistors to introduce propagation delays to mitigate the effects of the charge deposited in a device when it is hit with a high-energy ion.

A third type of radiation must also be guarded against which comes in the form of transient ionizing radiation, such as a short pulse of intense gamma rays that can lead to:

- *data loss* in binary elements,
- *latchup,* when parasitic bipolar transistors turn on due to high current in parasitic resistances,
- *snapback,* a high-current condition in n-channel transistors.

5 Design Techniques

To describe the process we employ to develop radiation-hardened microcircuits we will discuss the SA3300 microprocessor fabricated at AT&T. The CMOS technology provides high noise immunity, high frequency, low current, and static design features previously discussed as desirable for rad-hard circuit design.

Sandia with its expertise in designing radiation hardened CMOS integrated circuits joined with AT&T Bell Laboratories and their outstanding processing capabilities to develop an exceptionally radiation hard microelectronic technology (Figure 2). The basic configuration features two levels of metal and one level of polysilicon that provide a high-density conductor layout for interconnecting transistors and for power leads. An advanced configuration has a second level of polysilicon interconnection that provides high-value resistors for single event upset immunity.

The SA3300 is fabricated on an 86,000 square mil (0.33 by 0.26 in.) silicon chip. It contains 66,000 transistors plus other microelectronic components, operates at a clock speed of 15 MHz and is capable of processing more than one million instructions per second under worst-case radiation and other flight environments.

Sandia and AT&T Bell Laboratories have cooperatively developed radiation hardening techniques to ensure correct circuit operation, even in the presence of very high levels of ionizing radiation. These techniques depend on a synergism of circuit design, transistor structure, device layout, and process technology. We use conservative, fully static CMOS circuit techniques. Dynamic techniques (frequently used in commercial designs to reduce transistor count, increase density, and reduce chip size) depend on a charge being maintained and periodically refreshed in the device. This charge is stable in a normal environment but

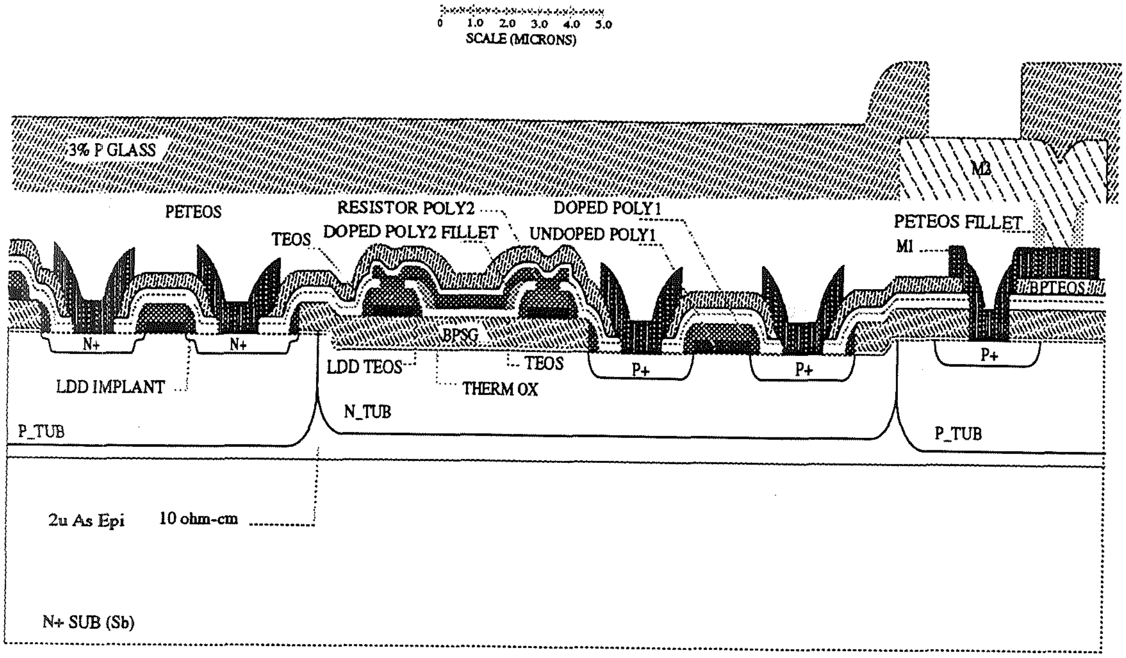


Figure 2: CMOSV Cross Section

can be lost during radiation. Dynamic techniques are, therefore, susceptible to radiation-induced upset and are unsuitable in hardened circuits.

As an additional measure, we use high-valued polysilicon resistors to delay propagation of a circuit disturbance caused by a cosmic-ray strike, a single event upset. This delay provides time for the transistor driving the struct node to restore the node to the correct state before the logic state of the binary storage circuit can be changed (Figure 3). The additional transistors and resistors required for these static-circuit and high-resistance hardening techniques make the chip area slightly larger than would be needed in a commercial microprocessor of comparable line-width technology.

Electronic charge generated by ionizing radiation, particularly for total dose, may be trapped in the gate oxide and can cause a shift in transistor threshold voltage. This radiation also generates photocurrents, in the case of short pulses of intense gamma rays, that can cause malfunction in unhardened commercial integrated circuits. Such intense pulses of radiation can generate currents causing binary data loss, latchup, or snapback. The heat generated by high currents caused by latchup may fuse bond wires or chip metalization to cause open-circuit failures (an epitaxial technology is used to prevent latchup). Snapback can render a circuit useless at lower currents by an SCR type action between the intentional CMOS devices and certain layout sensitive bipolar parasitic devices in conjunction with operating voltage (the phenomena was most frequently encountered with circuit voltages near 10 volts). Recovery from snapback requires a temporary removal of power from the circuit.

Radiation (x-rays or gamma rays) interacts with field and gate oxides to generate electron-hole pairs. The highly mobile electrons that do not recombine with holes are swept out of the oxide by the gate bias. However, holes, which are less mobile, are trapped and degrade device performance. We minimize this effect by using thin (less than 200 angstroms) gate

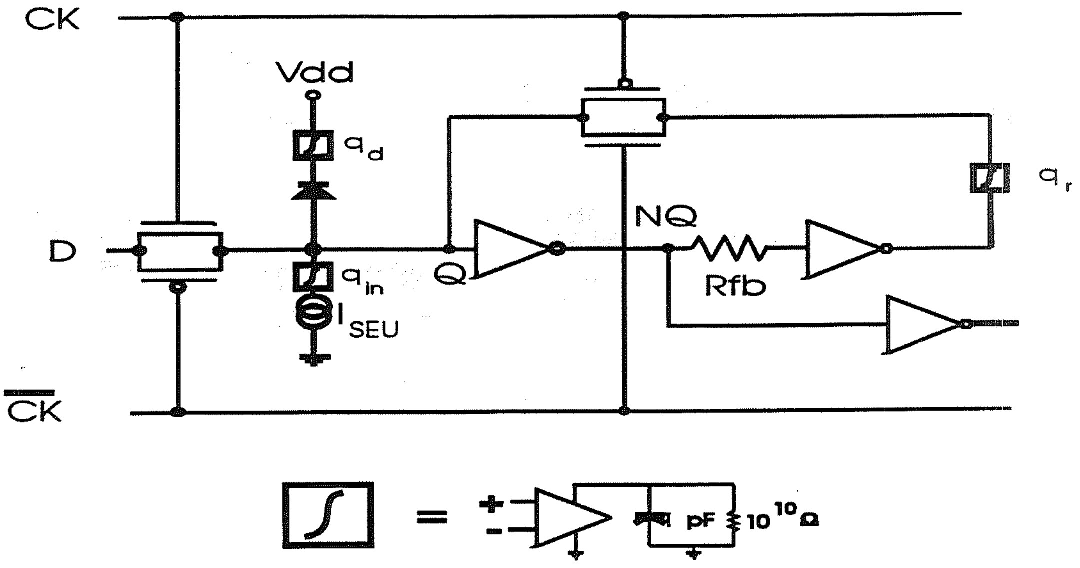


Figure 3: SEU Hardened D-Latch

oxides that limit the volume available for electron-hole pairs to form. Careful control of oxide growth, deposition conditions, and thermal cycles during manufacture also aid in minimizing threshold voltage shifts.

In addition to contributing to compactness of device design, the second level metal provides an additional low resistance interconnection path for power and ground routing. This low resistance reduces voltage drop caused by photocurrents generated during transient radiation which via rail collapse can cause circuit upsets including upsets in binary storage elements.

6 Device Performance

Data for the SA3300 will be used to demonstrate device performance obtainable using the above described design techniques and a radiation hard integrated circuit manufacturing process; in particular, for the AT&T CMOSV technology. We have, however, demonstrated quite similar results in performance for the SA3865 microcontroller designed by Sandia and manufactured in the UTMC 1.2 micron CMOS rad-hard process.

In Figure 4 the effect of feedback resistors on microprocessor performance is shown at 4.5 V as a function of temperature from -55 degrees to 125 C degrees C. For a low-valued feedback resistance of 3.6 kohms, the maximum frequency of operation at 4.5 V decreases almost linearly from a high of 34 MHz at -55 C to a low of 23 MHz at 125 C. This decrease is primarily due to a decrease in drive current caused by decreased carrier mobility over this temperature range. For a high value of 400 kohms for the feedback resistor, the frequency response increases monotonically from a low of 3MHz at -55 C to a high of 18 MHz at 125 C. The frequency performance here is dominated by feedback resistance. As feedback resistance decreases with increasing temperature the speed increases. For an intermediate value of feedback resistance of 160 kohms, performance increases with temperature then

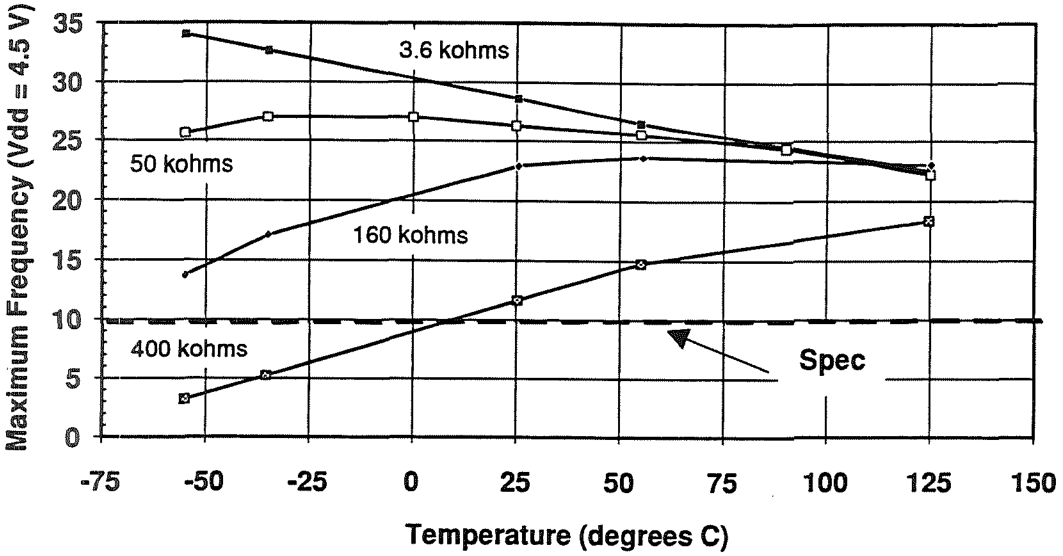


Figure 4: SA3300 Pre-Rad Performance

flattens above approximately 25 C. At lower temperatures, the performance increases with temperature, consistence with the resistance dominated performance of the 400 kohm curve. Above 25 C, the effect of decreasing resistance is compensated by decreasing current drive. The competing effect of decreasing feedback and decreasing transistor drive result in a nearly flat frequency response as a function of temperature for the 50 kohm parts. Parts with feedback resistors of 160 kohms or less exceed the performance specification of 10 MHz over the full temperature range. A 15 MHz requirement for the ISTP program was met with resistors in the 50 kohm range.

In Figure 5, we contrast the SEU upset response of the commercial NS32016 with that of the SA3300 at room temperature. The NS32016 is an NMOS microprocessor, and has a clear upset threshold of about 3 Mev-cm2/mg for the registers. This compares to a 23 Mev-cm2/mg threshold for the SA3300 using low-valued feedback resistors of 3.6 kohms. This illustrates the effectiveness of design hardening and technology improvements to increase the SEU tolerance of D-latch circuits. Further improvement in hardness is seen for the parts with 50 kohm and 160 kohm feedback resistors.

The expected error rate for the SA3300 and the NS32016 in Adams' 10% worst case environment at geosynchronous orbit are shown in Figure 6. Assuming for comparison that the NS32016 has the same number of sensitive latches as the SA3300, the 3 Mev-cm2/mg translates into roughly one error every six days. With a combination of circuit-design and technology changes the error rate decreases to about once a year. With both design and resistor hardening of 50 kohm, the error rate is decreased to once every 16 years. For upset thresholds above 100 Mev-cm2/mg (resistors between 50 and 160 kohm) the error rate drops to less than once every 100 years. Note from Figure 4 that this is accomplished while maintaining the part performance specification of greater than 10 MHz over the full temperature range.

Upset levels for pre and post neutron irradiated SA3300 devices are shown in Figure 7. All neutron irradiated devices exhibited upset levels in excess of 10E9 rad(Si)/sec, exceeding the design goal for this technology. These devices were exposed to a neutron fluence of 4E14

2.1.8

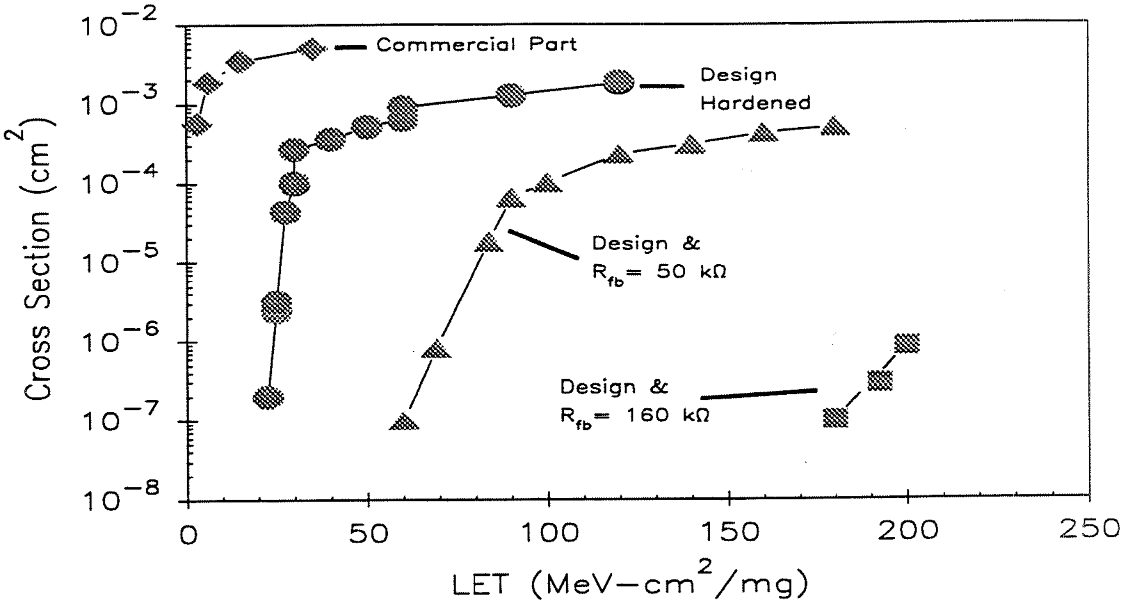


Figure 5: SA3300 SEU Tolerance

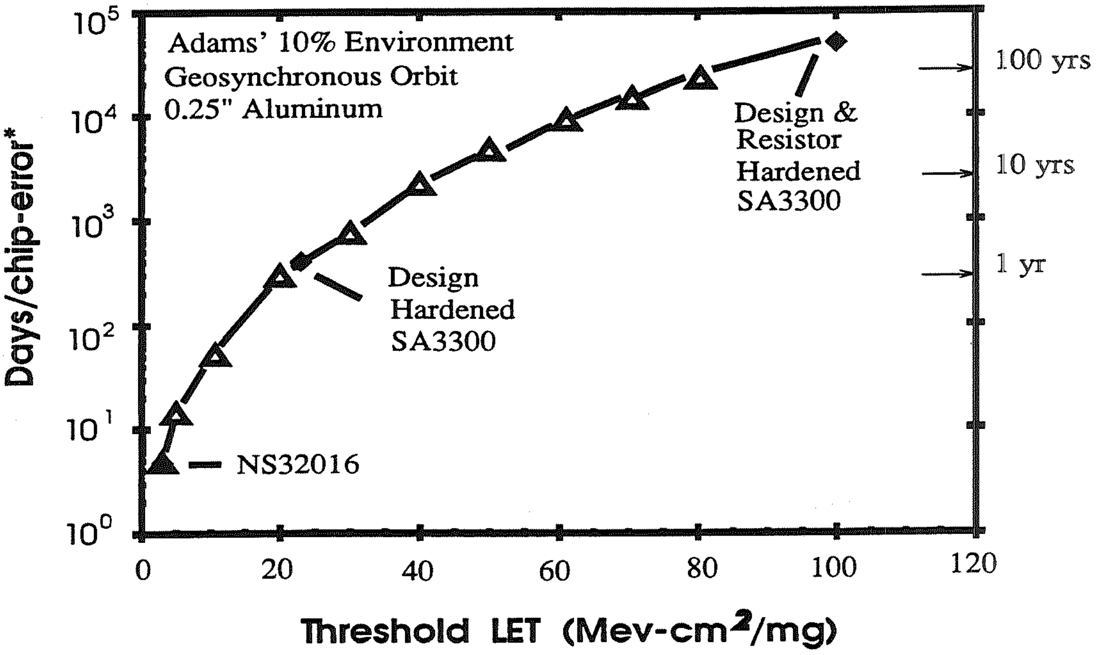


Figure 6: Microprocessor Error Rate

n/cm2 prior to packaging. The large increase in post-neutron upset level for the microprocessor can be attributed to damage in the n+ substrate region resulting in a reduction in lifetime and a corresponding reduction in collection depth and damage causing a reduction in vertical npn gain. The upset levels shown in the figure are for a 1 microsecond wide pulse (the upset level with a 30 ns pulse width for the pre and post 4E14 neutron-irradiated device was 1E9 and 3E9, respectively).

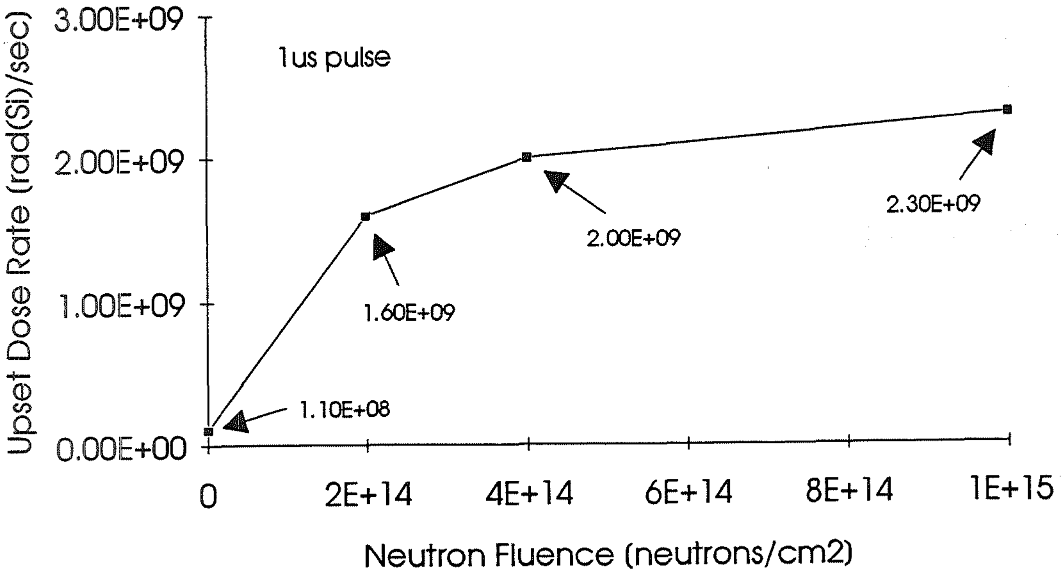


Figure 7: SA3300 Pre & Post Neutron Upset Levels

The percent degradation in maximum frequency (FMAX) and output drive current of the SA3300 microprocessor as a function of total dose is shown in Table 1. Total dose testing was performed on a small sample of devices using a Cobalt-60 gamma source at a rate of about 1E6 rad/hour. Degradations in speed and drive were virtually undetectable at 0.2 Mrad, and were less than 10continued to function well at 5 Mrad although the maximum frequency had degraded to about 18%.

SA3300	Post Radiation rad(Si)		
Parameter	$2.00E+05$	$1.00E+06$	$5.00E+06$
IOL	−2%	−5%	−15%
IOH	−3%	−4%	−11%
Max. Freq.	−2%	−6%	−18%

Table 1: Total Dose Degradation in % Change form Pre-Rad Value

7 Summary

In summary, the radiation tolerance of the NS32016 has been dramatically improved by a combination of design and feedback resistor hardening techniques. By using a combination of oversized transistors and thin epi, the SEU upset threshold increased from about 3 Mev-cm2/mg to 23 Mev-cm2/mg. With 50 kohm and 160 kohm feedback resistors, the upset threshold increased to about 60 Mev-cm2/mg and 180 Mev-cm2/mg, respectively, while maintaining the performance requirement. Transient upset of the SA3300 was on the order 1E8 and 1E9 rad(Si)/sec for 1us and 30ns pulse widths, respectively, with an improvement to 1E9 and 2E9, respectively with a neutron irradiation of 4E14 rad(Si)/sec. The total dose tolerance exceeded expectations with less than a 10% degradation in maximum operating frequency to more than a Mrad.

8 Acknowledgements

The authors gratefully acknowledge the assistance of R. W. Beegle, P. L. Candelaria, and M. A. Serna for their support in testing and D. W. Peterson, F. W. Sexton, J. N. Sweet, and M. D. Terry for assistance with some of the graphics.

References

[1] T. F. Wunsch, G. L. Hash, F. W. Hewlett, and R. K. Treece, "Transient Radiation Hardness of the CMOSV 1.25 micronTechnology," IEEE Trans Nucl. Sci., Vol. 38, No. 6, December, 1991.

[2] F. W. Sexton, W. T. Corbett, R. K. Treece, K. J. Hass, K. L. Hughes, C. L. Axness, G. L. Hash, M. R. Shanneyfelt, and T. F. Wunsch, "SEU Simulation and Testing of Resistor-Hardened D-Latches in the SA3300 Microprocessor," IEEE Trans Nucl. Sci., Vol. 38, No. 6, December, 1991.

Design of CMOS Logic Gates for TID Radiation[1]

John Okyere Attia and Maria L. Sasabo
Laboratory for Radiation Studies
Prairie View A&M University

Abstract - **The rise time, fall time and propagation delay of the logic gates were derived. The effects of total ionizing dose (TID) radiation on the fall and rise times of CMOS logic gates were obtained using C program calculations and PSPICE simulations. The variations of mobility and threshold voltage on MOSFET transistors when subjected to TID radiation were used to determine the dependence of switching times on TID. The results of this work indicate that by increasing the size of P-channel transistor with respect to the N-channel transistors of the CMOS gates, the propagation delay of CMOS logic gate can be made to decrease with, or be independent of an increase in TID radiation.**

1 Introduction

MOS devices when subjected to total ionizing dose (TID) radiation experience shifts in the device parameters due to increased oxide trapped charges and interface states [1]. The principal TID effects on MOS transistors are negative shift in the threshold voltage and degradation of channel mobility [2]. The threshold voltage and mobility affect the performance of electronic circuits. In this work, we investigate the effect of TID on the rise time, fall time and propagation delay of CMOS INVERTERS, NAND and NOR gates. In addition, the effect of transistor sizing on the propagation delay of CMOS logic gates under the influence of TID radiation will be determined.

2 Switching Time Expressions

Figure 1 shows a block diagram of a generic CMOS logic gate. The node capacitance at the output represents the equivalent load capacitance due to transistor gate capacitance, metal interconnect capacitance, and diffusion capacitance [3]. The capacitor will be charged during pull-up (or rise time transition) and it will be discharged during pull down (of fall time transition). The P-channel and N-channel network represent N-channel and P-channel transistors, either as singles or in series or parallel combinations. CMOS inverters, nand and nor gates can be obtained from the generic network shown in Figure 1 [3].

Switching time is a measure of the speed of operation of a logic gate. The switching times, considered in this work, are the rise time, fall time and propagation delay. The rise time is defined as the time required for logic gate output to rise to 90% of the maximum possible output, while the fall time is the time taken for the output to fall to 10% of the maximum possible output. The fall and rise time definitions are similar to those used by

[1] Partially supported by NASA JSC-NAG9-337

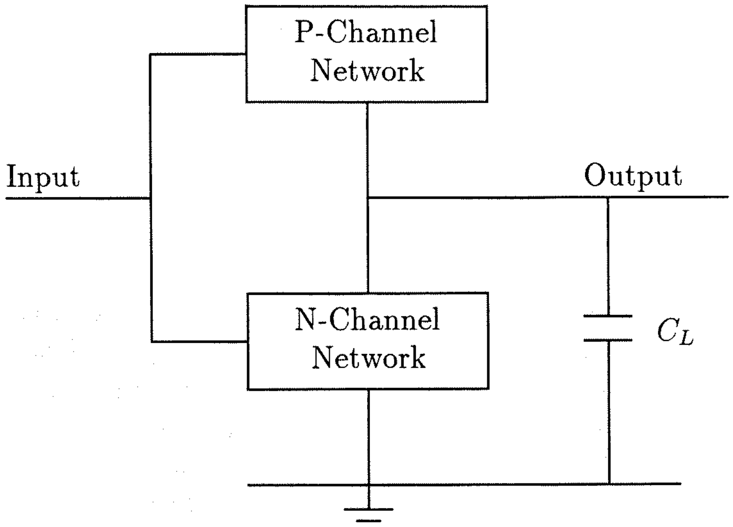


Figure 1: Block Diagram of Generic CMOS Logic Gate

Taub and Schilling [4]. The propagation delay is the arithmetic mean of the rise time and fall time.

To derive the CMOS inverter fall time, assume that the capacitor is initially charged to the maximum voltage, $V_c = V_{omax}$. When the inverter output falls from high to low, the NMOS transistor conducts. Initially, the transistor will be in saturation. After a time, due to the discharge of the capacitor, the transistor conducts in the non-saturation region. The time the NMOS transistor is in saturation can be shown to be [5]:

$$t_{sat} = \frac{2L_n C_L (V_{omax} - V_{gsn} + V_{tn})}{W_n \mu_n C_{ox} (V_{gsn} - V_{tn})} \tag{1}$$

where

$L_n =$ length of n-channel device.
$W_n =$ width of n-channel device.
$V_{gsn} =$ voltage between gate and source of n-device.
$\mu_n =$ mobility of n-channel device.
$C_{ox} =$ gate oxided device.

The symbols in the equation (1) are similar to those in the literature [3]. The time the transistor is in non-saturation region is given as:

$$t_{nonsat} = \frac{-0.5 C_L K_n^{-1}}{(V_{gsn} - V_{tn})} \left[ln \frac{(0.1(2(V_{gsn} - V_{tn}) - V_{omax})}{(2)(V_{gsn} - V_{tn}) - 0.1V_{omax})} \right] \tag{2}$$

where:

$$K_n = \frac{\mu_n C_{ox} W_n}{L_n} \tag{3}$$

The fall time is the sum of t_{sat} and t_{nonsat}. Adding equations (1) and (2), the fall time becomes:

$$t_f = \frac{(V_{omax} - (1 + 0.5lnY)(V_{gsn} - V_{tn}))}{0.5C_L^{-1}K_n(V_{gsn} - V_{tn})^2} \tag{4}$$

where

$$Y = \frac{0.1(2((V_{gsn} - V_{tn}) - V_{omax}}{2(V_{gsn} - V_{tn}) - 0.1V_{omax}} \tag{5}$$

Equation (1) is similar to that given in the literature [3, 4]. However, the expression for t_{nonsat} given in Equation (2) is different from that obtained by Taub and Schilling [4]. The latter obtained the following expression for t_{nonsat}:

$$t_{nonsat} = \frac{-C_L ln[0.1]}{K_n(V_{gsn} - V_{tn})^2} \tag{6}$$

It was assumed by Taub and Schilling that:

$$\frac{(V_{gsn} - V_{tn}) - V_{omax}}{(V_{gsn} - V_{tn}) - 0.1V_{omax}} = 1.0 \tag{7}$$

which is incorrect.

To derive the rise time equation, we assume that the load capacitance is originally discharged. The PMOS is then in the saturation region, and as the output voltage increases, the device enters into the non-saturation region. The time taken for the PMOS device to be in the saturation region is given as:

$$t_{sat} = \frac{2C_L(V_{omax} - (V_{sgp} - |V_{tp}|)}{K_p(V_{sgp} - |V_{tp}|)^2} \tag{8}$$

where:

$L_p =$ length of p-channel device.
$W_p =$ width of p-channel device.
$\mu_p =$ mobility of PMOS device.
$V_{sgp} =$ voltage between the source and gate of p-channel device.

and

$$K_p = \frac{\mu_p C_{ox} W_p}{L_p} \tag{9}$$

The time the PMOS is in the non-saturation region is given as:

$$t_{nonsat} = \frac{C_L lnX}{K_p(V_{sgp} - |V_{tp}|)} \tag{10}$$

The rise time is obtained by adding the equations (8) and (10):

$$t_r = \frac{C_L(V_{omax} - (1 - 0.5lnX)(V_{sgp} - |V_{tp}|))}{K_p(V_{sgp} - |V_{tp}|)^2} \tag{11}$$

where:

$$X = \frac{2.1(V_{sgp} - |V_{tp}|)V_{omax} - 2(V_{sgp} - |V_{tp}|)^2 - 0.1V_{omax}^2}{0.1V_{omax}(3(V_{sgp} - |V_{tp}|) - V_{omax})} \tag{12}$$

The propagation delay, t_{pd}, is the average of the rise time and fall time:

$$t_{pd} = \frac{(t_f + t_r}{2} \tag{13}$$

To obtain the switching times of the NAND and NOR gates, it should be noted that for the NAND or NOR gates, P-channel and N-channel networks contain transistors which are either in series or in parallel. If the transistors are in series, the switching time is approximately the sum of the switching time of the individual transistors [6]. However, if the transistors are connected in parallel, then the switching time will be reduced if all the transistors are conducting [6]. Equations (3) and (9) were modified to obtain the switching times of NAND and NOR gates.

3 C-Program Calculations and Results

To investigate the effect of TID on CMOS inverters, NAND and NOR gates, the switching time equations were employed. C programs were written to calculate the fall time, rise time and propagation delays of the logic gates. For different values of TID radiation, the corresponding values of threshold voltages and mobilities for the P-channel and N-channel transistors were obtained from available data [1,2]. This is shown in Table 1.

Dose	V_{tn}	V_t	μ_n	μ_p	KP_n	KP_p
1.e4	1.42	-1.04	408	200	3.20e-4	1.60e-4
4.e4	1.37	-1.08	401	199	3.14e-4	1.59e-4
7.e4	1.24	-1.13	397	194	3.11e-4	1.55e-4
1.e5	1.19	-1.19	394	188	3.09e-4	1.50e-4
2.e5	1.02	-1.22	384	180	3.01e-4	1.44e-4
3.e5	0.92	-1.28	380	176	2.98e-4	1.41e-4
4.e5	0.82	-1.30	368	172	2.89e-4	1.38e-4
6.e5	0.68	-1.36	360	164	2.82e-4	1.31e-4
7.e5	0.60	-1.38	356	162	2.79e-4	1.30e-4
9.e5	0.49	-1.40	353	158	2.77e-4	1.20e-4

Table 1: Total ionizing dose versus mobility and threshold voltage and calculated KP_n, and KP_p.

The mobility and the threshold voltage at various TID radiation were used to compute the rise and fall times of the CMOS gates. The minimum dimensions for the PMOS and NMOS devices are:

$t_{ox} = 500$ Angstrom $\quad C_{ox} = 8.0E - 04pF/um^2$

$L_n = L_p = 2um \quad W_n = W_p = 5um \quad C_L = 0.72pF$

Equations (3) and (9) were used to obtain the fall and rise times of CMOS inverter. Modified versions of the above mentioned equations were used to calculate the switching times of 3-input CMOS NAND and NOR gates.

Figures 2, 3 and 4 show the switching times for inverter, 3-input NAND and 3-input NOR gates, respectively. It can be seen from the figure that, the rise time increases with TID while the fall time decreases with TID.

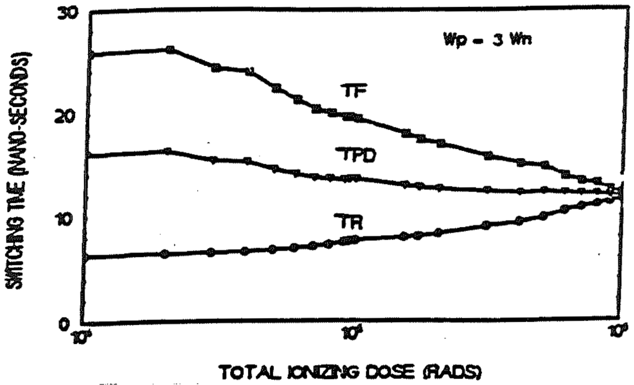


Figure 2: Fall time(TF), rise time(TR) and propogation delay(TPD) versus TID of CMOS inverter for $W_p = 3W_n$

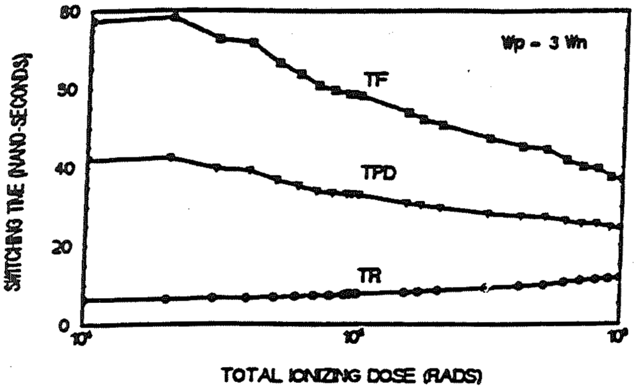


Figure 3: Fall time(TF), rise time(TR) and propogation delay(TPD) versus TID of CMOS NAND gate for $W_p = 3W_n$

The widths of the P-channel transistors were changed to obtain the effects of transistor sizing on the switching time. Figures 5, 6, and 7 show the propagation delay versus TID for inverter, 3-input NAND gate, and 3-input NOR gate, respectively. It is interesting to note that, by increasing the size of the P-channel transistor with respect to that of the N-channel transistor, the propagation delay of CMOS logic gates can be made to decrease. Furthermore, from figures 2 and 3, it can be seen that the propagation delay decreases with an increase in TID for INVERTERS and NAND gates. However, from Figure 4, there is a slight increase in propagation delay with respect to TID for NOR gates.

2.2.6

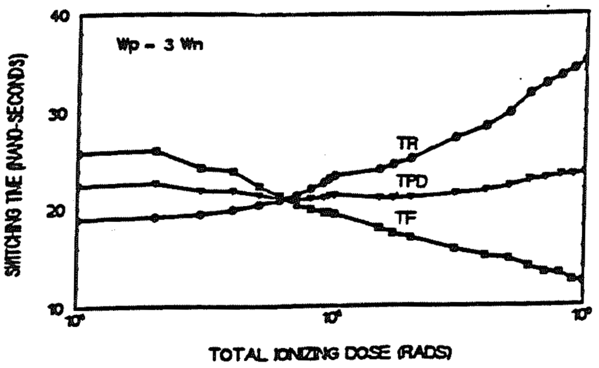


Figure 4: Fall time(TF), rise time(TR) and propogation delay(TPD) versus TID of CMOS NOR gate for $W_p = 3W_n$

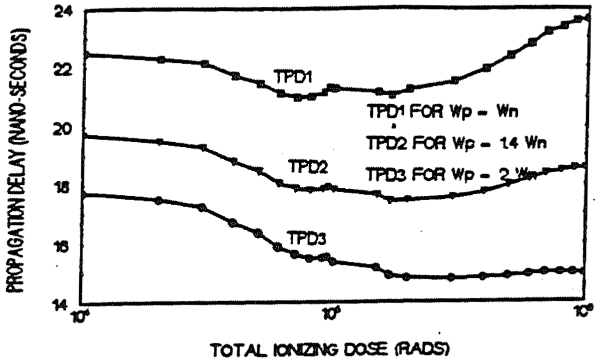


Figure 5: Propogation delay versus TID of CMOS inverter for various transistor sizes

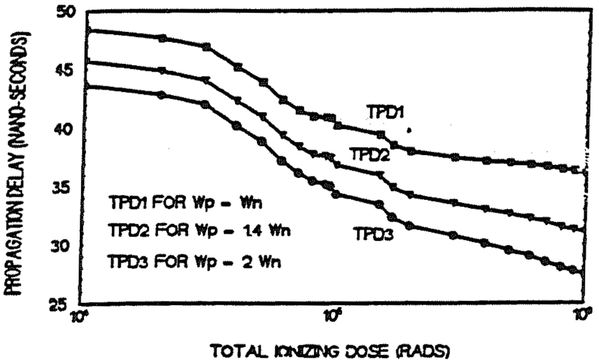


Figure 6: Propogation delay versus TID of CMOS NAND gate for various transistor sizes

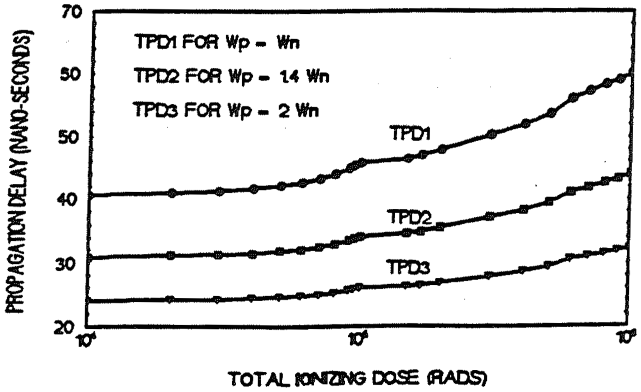


Figure 7: Propogation delay versus TID of CMOS NOR gate for various transistor sizes

4 PSPICE Simulations and Results

PSPICE is one version of SPICE, which is a standard program for simulation of integrated circuits. SPICE level 3 model was used to obtain more accuracy. The effect of total ionizing dose on switching time can be simulated through the threshold voltage VTO and KP parameter. The KP parameter is related to mobility as KP $= \mu C_{ox}$ [6]. The values of total ionizing dose versus mobility and threshold voltage used for PSPICE simulation are shown in Table 1.

The width of P-channel with respect to that of N-channel was varied in turn to obtain the effect of total ionizing dose on switching time of CMOS inverter, NAND and NOR gates. The rise time and fall time were obtained from PSPICE .PROBE data, while the propagation delay was calculated by averaging the rise time and fall time.

Figures 8, 9 and 10 show the graphs of the propagation delay versus TID for CMOS INVERTER, NAND and NOR gates, respectively. For the NAND gate, only one P-channel transistor was assumed to be on during simulation. This represents the maximum propagation delay. The minimum propagation delay is obtainable when all the PMOS transistors are on. For all the three CMOS logic gates, the fall time decreased with an increase in TID. On the other hand, the rise time increased for an increase in TID. These resultas are similar to those obtained from the C program calculations. In addition, it can be seen from the figures that, as the width of the P-channel transistors is increased with respect to that of the N-channel transistors. the propagation delay decreased.

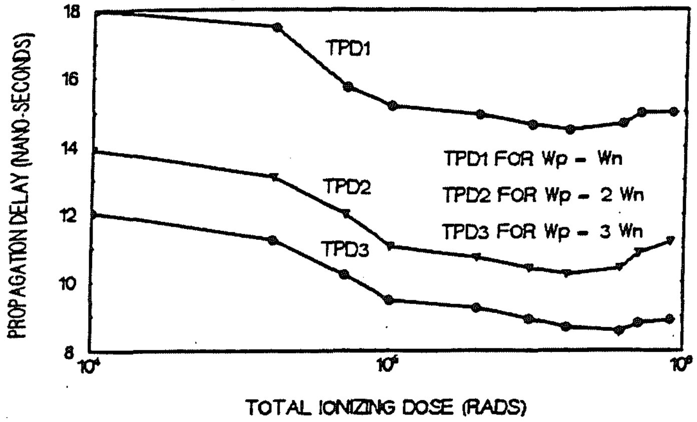


Figure 8: Propogation delay versus TID of CMOS inverter for various transistor sizes (PSPICE results)

5 Conclusions

From the C program calculations and the SPICE simulations, the following conclusions can be made. The fall time for the logic gates decreases with an increase in TID. The rise time for the gates increased with an increase in TID. In addition, when the width of the P-channel transistors are increased with respect to the N-channel transistors, the propagation delay decreased. Furthermore, by making the width of the P-channel transistors equal to three times the width of the N-channel transistors, the propagation delay of CMOS inverters,

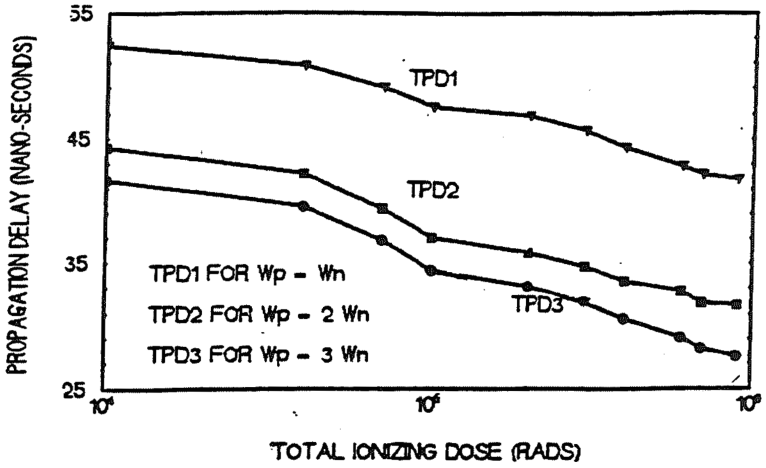


Figure 9: Propogation delay versus TID of CMOS NAND for various transistor sizes (PSPICE results)

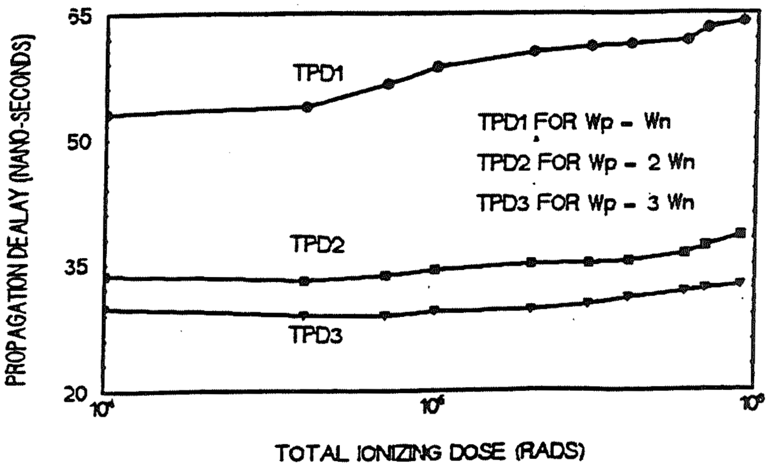


Figure 10: Propogation delay versus TID of CMOS NOR for various transistor sizes (PSPICE results)

3-input NAND gate and 3-input NOR gates can be made to decrease with or be almost invariant with an increase in TID radiation. The results of this work indicate that it is possible to design of CMOS logic gates whose switching times are almost independent of TID radiation.

References

[1] Winokur,P.S., Schwank,J.R., WcWhorter,P.J, Dressendorfer P.V. and Turpin,D.C. "Correlating the Radiation Response of MOS Capacitors and Transistors," IEEE Trans. on Nucl. Sci., Vol. NS-31, pp. 1453 - 1460, 1984.

[2] Dressendorfer,P.V, Soden,J.M, Harrington J.J. and Nordstrom, T.V, "The Effects of Test Conditions on MOS Radiation-Hardened results," IEEE Trans. on Nucl. Sci. Vol. NS-28, pp. 4281 -4287, 1981.

[3] Uyemura,J.P *Circuit Design for CMOS VLSI,* Kluwer Academic Publishers, 1992.

[4] Taub, H. and D. Schilling, D, *Digital Integrated Electronics,* McGraw Hill, 1977.

[5] Sasabo, Maria, "Effects of Total Ionizing Dose on Switching Times of CMOS Logic Gates", M.S. Thesis, Prairie View A&M University, Prairie View, Texas, August 1992.

[6] Geiger, R.L., Allen, P.E. and Strader, N.R., em VLSI Design Techniques for Analog and Digital Circuits, McGraw Hill, 1990.

Evaluation of Radiation Effects in Re-Oxidized Nitrided Oxide Devices by Hot Carrier Stressing at 77 K

N.C. Das, and V. Nathan
Phillips Laboratory / VTRP
Kirtland Air Force Base, New Mexico 87117-5776

Abstract - **An attempt has been made to understand the mechanisms of radiation induced degradation by using different types of hot carrier (HC) stressing experiments. The degradation of re-oxidized nitrided oxide gate dielectric due to irradiation and hot carriers are reported. The effect of electron and hole traps of the virgin device on radiation induced threshold voltage shift is discussed.**

1 Introduction

The reliability of MOS devices in space environments depends on their hardness to irradiation of high energy particles. Hence the design of VLSI circuit depends on the design parameters of the device fabricated with radiation hardened technology. In recent years many studies have been performed to use re-oxidized nitrided oxide as a substitute for thermally grown silicon dioxide for gate insulator [1,2]. Thermally nitrided SiO_2 has many advantages over thermal oxides: resistance to the penetration of impurities and to the interface state generation by irradiation and high field stressing [3]. Nitridation increases the number of electron traps [4] and decreases the hole trap density [5] in the oxide. Various mechanisms have been proposed for increased radiation hardness of the re-oxidized nitrided oxide (ONO) film. Ramesh et al [6] proposed that there are more compensating electron traps and Dunn et al [5] proposed that there are fewer hole traps in ONO film than in the SiO_2 film. Yount et al [7] from an ESR study concluded that the hole traps in ONO film are different from the hole traps (E' center) in thermal oxide. Though most of the studies on radiation effects in ONO film have been done at room temperature, a few attempts have been made to study the radiation effects at cryogenic temperatures. An experiment at low temperature will be useful in understanding the exact mechanism of radiation hardness of ONO film, since at low temperature the dominant degradation mechanism due to irradiation is charge trapping in the oxide, and interface state generation is negligible.

Various experiments have been done to correlate hot carrier (HC) and ionizing radiation effects in silicon dioxide devices. It has been reported [8], that the devices fabricated with a radiation hard process also have less hot carrier degradation. Palkuti et al [9], reported that there exists a direct relationship between the interface trap generation by hot carrier stressing and that by ionizing radiation in silicon dioxide devices. Our literature search found that no attempt has been made to correlate the HC effect and radiation effect in ONO devices at cryogenic temperatures. The small geometry MOS devices used in space need to be not only radiation hardened but also should have minimum degradation due to high field stressing. We performed detailed experiments on ionizing radiation and HC stressing in ONO devices fabricated with different processing conditions. The objective is to determine the mechanism of radiation induced degradation from the results of HC stressing experiments. Different types of stressing biases were used for hot carrier injection. The effect of both electron and hole traps of virgin devices on its radiation hardness was determined.

2 Experimental Details

Experiments were carried out on n-channel transistors with nitrided oxide (NO) and ONO gate dielectrics. The device identifications and the corresponding gate insulator processing conditions are given in Table 1. The nitridation and re-oxidation were carried out by rapid thermal processings. The gate oxide thickness, channel length, and channel width of the devices were 150 $\AA$, $1.25 \mu m$, and $11.0 \mu m$ respectively. The devices were fabricated at TRW microelectronic research center, CA. Except gate dielectric all other processing conditions were same for these devices. The devices were bonded in 24 pin ceramic package and mounted in a cryostat. The irradiation was performed in a Co-60 gamma cell at a dose rate of ≈ 4.1 *krad* (Si)/min. Two types of HC stressing were done. In type A, maximum substrate current biasing condition was used, in which it is believed that maximum interface state generation occurs for particular drain voltage [10]. In type B stressing, maximum gate current stressing ($V_g = V_d = 5.5\ V$) was used, in which maximum electron injection and trapping occur for particular drain bias [11]. Threshold voltage (V_t) was calculated from maximum slope of I_d versus V_g curve in the linear operating region of the device for $V_d = 0.1\ V$. The interface state density (D_{it}) was measured by charge pumping (CP) technique [12], in which a triangular pulse of 5.0 V amplitude and 100 kHz frequency was applied to the gate. The base of the gate pulse was varied from $-5.0\ V$ to $0.0\ V$. The source and drain were reverse biased at $0.5\ V$.

Table 1 Gate oxide processing conditions for different devices

Device Identification	$t_{ox}(\AA)$	Nitridation temp.(°C)/time(sec.)	Re-oxidation temp.(°C)/time(sec.)	Description
D1	150	950/45	1150/60	Light ONO
D2	150	1050/45	1000/60	Moderate ONO
D3	150	1150/45	No re-oxidation	Heavy NO

3 Results and Discussions

The V_t shifts for different total doses of irradiation at 77 k are given in Fig. 1. We observed minimum V_t shift for heavily nitrided (D3) device whereas maximum shift for lightly nitrided device. For devices D2 and D3 a near saturation in V_t shift is observed for high total doses.

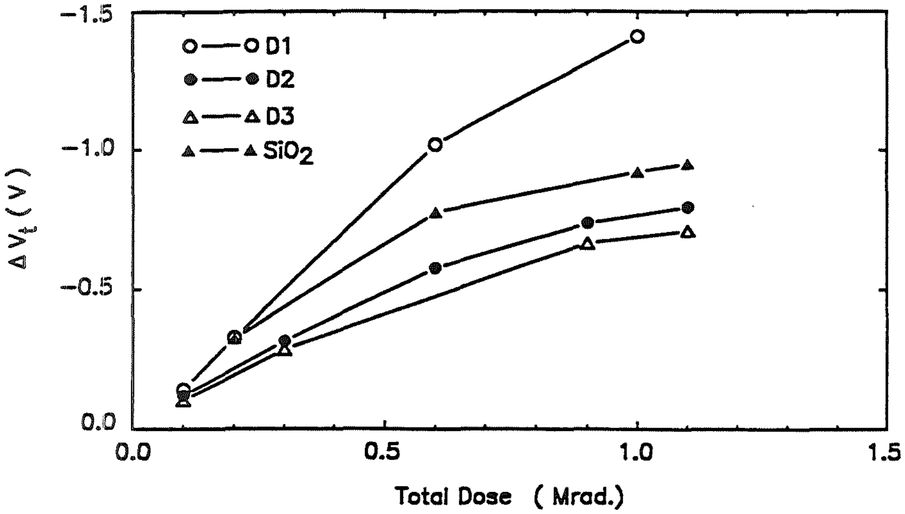


Figure 1: The threshold voltage shift versus total dose for three types of devices. Both irradiation and measurement were done at 77K.

Also shown in this figure is the V_t shift of the silicon dioxide device. It is interesting to note that though the V_t shifts of heavy and moderate ONO devices are less than that of silicon dioxide device, the V_t shift of lightly nitrided device is more than that of SiO_2 device. This may be due to the fact that during the high temperature ($1150°C$) nitridation condition of the lightly nitrided device, a large number of bulk traps are created in the oxide. Our CP measurements show (not shown here) negligible increase of D_{it} due to irradiation for all the devices. This agrees with similar experimental observation by Saks et al. [13] at 80 K. In Fig. 2, the increase of D_{it} by type A stressing is given. A large increase of interface state density is observed due to type A stressing, in contrast to negligible increase by high energy irradiation.

This means that the mechanism of interface state creation by HC is different from that of ionizing radiation. A maximum increase of D_{it} was observed for D1 device and minimum increase for D3 device. The increase of V_t shift due to type A stressing for different devices is shown in Fig. 3.

It is to be noted that the V_t shift is positive unlike that due to high energy irradiation, in which case a negative V_t shift is observed. The positive V_t shift is due to increase of interface states as well as that of electron traps in the oxide. For devices D1 and D2, the V_t shift is mainly due to increase of interface states. For device D3, though the increase of D_{it} is less than that of D1 and D2, the electron trapping is the dominant mechanism for the enhanced

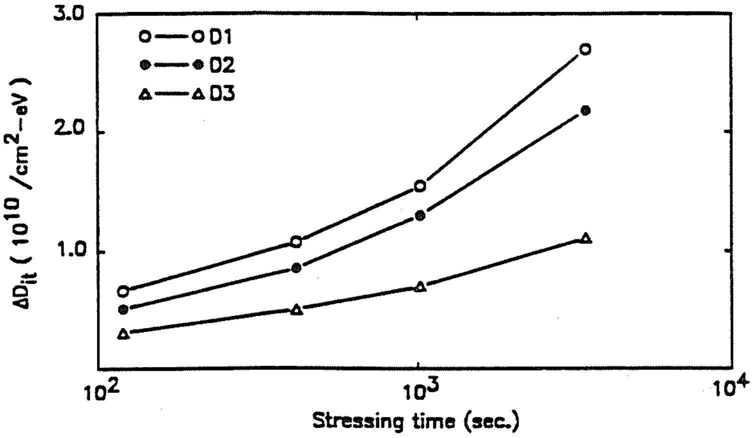


Figure 2: Increase in interface state density (D_{it}) versus stressing time for type A stressing at 77K. The stressing biases for different devices were chosen such that peak substrate currents for all the devices were 62 μA. Stressing bias for D1, $V_d = 4.1$, $V_g = 2.8\ V$, for D2, $V_d = 4.5$, $V_g = 2.6\ V$, and for D3, $V_d = 4.5$, $V_g = 3.0\ V$.

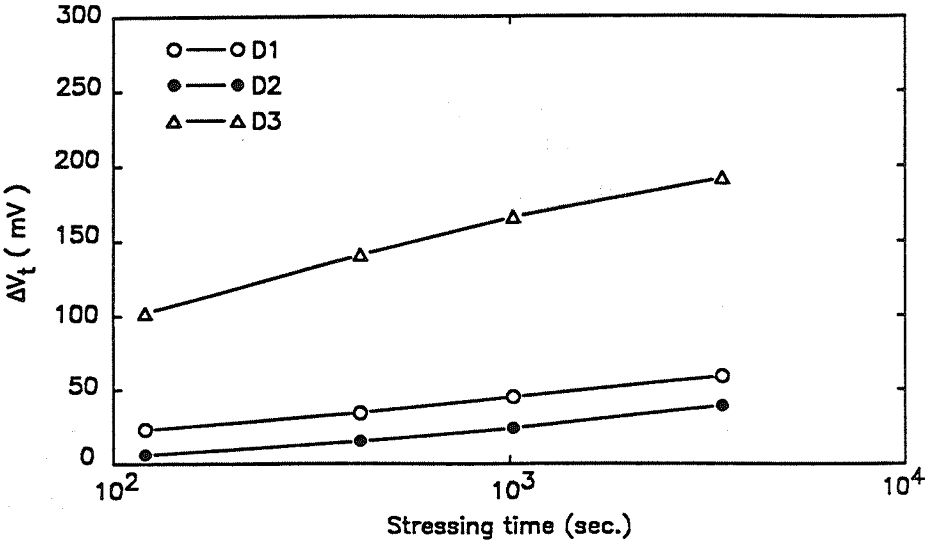


Figure 3: Threshold voltage shift versus stressing time for type A stressing.

V_t shift.

For type B stressing, the positive V_t shift, due to electron trapping, is maximum for D3 device (Fig. 4).

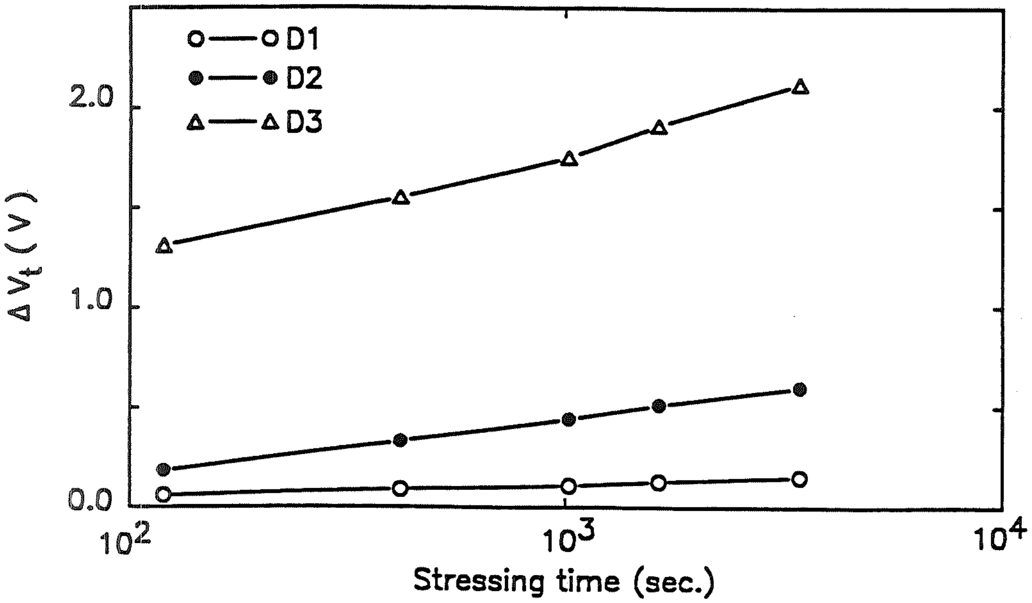


Figure 4: Increase of V_t versus stressing time for type B stressing at 77K. All the devices were stressed with $V_d = V_g = 5.5\ V$ biasing condition.

This type of stressing results in maximum electron trapping at the drain gate boundary. Hence the V_t shift is proportional to the electron trap density of different devices. The V_t shift is very large for D3 device even at very low stressing time. The increase of D_{it} for type B stressing is minor (not shown here) for all the three types of insulators. It may be mentioned here that though the radiation induced negative V_t shift for D3 device is minimum, the hot electron induced positive V_t shift is very high and thus the device may not be suitable from the reliability point of view.

Severi et al [14] attributed the increase of D_{it} due to high field stressing to the hole trap density in the oxide. This model would explain the one to one correspondence in the rank ordering of the V_t shift in Fig.1, and the D_{it} increase in Fig. 2, for each device. Note however that D2 is close to D3 in Fig. 1, and close to D1 in Fig. 2, and hence the correspondence is not of direct proportionality for D2 device. The voltage shift under type B stressing in Fig. 4 tells us that the electron trap densities for these three devices are quite different. Device D1 has little electron trapping, so we observe a rapid accumulation of hole traps due to irradiation (Fig. 1). In devices D2 and D3 there is appreciable trapping of electrons. When these devices are irradiated, electrons are trapped near the gate and holes are trapped near the silicon interface. This space charge causes a field collapse [15] region at the center of the gate insulator film. The field collapse region is similar for D2 and D3; hence their $\triangle V_t$ curves are close (Fig. 1). For moderate ONO devices, the radiation induced V_t shift is due to both less hole traps and increased electron traps.

4 Conclusion

In conclusion, HC stressing is useful to investigate the trap structure of the gate insulating film in silicon filed effect transistors. The irradiation induced V_t shift in moderate ONO device is due to both reduced hole trap and increased electron trap density. Moderate nitrided ONO device is the most suitable device from both radiation hardness and hot carrier injection points of view. The correlation results indicate that HC stressing experiments could be used as more economical tests for statistical process control.

Acknowledgments

This work was supported in part by an Air Force Materiel Command/ National Research Council Research Associateship. The authors thank J. Cable of TRW and J. Pickel of S-CUBED for providing the devices used in the experiment. The help of D.R. DeButy in the experimental measurements is gratefully acknowledged.

References

[1] A.B. Joshi, G.O. Lo, D.K. Shih, and D.L. Kwong, "Effect of Rapid Thermal Reoxidation on the Electrical Properties of Rapid thermally Nitrided Thin-Gate Oxides," IEEE Trans. Electron Devices, Vol. 39, 883, April, 1992.

[2] J.S. Cable, and J.C.S. Woo, "High-Field Mobility Effects in Reoxidized Nitrided Oxide (ONO) Transistors," IEEE Trans. Electron Devices, Vol. 39, 607, March, 1992.

[3] T. Hori, H. Iwasaki and K. Tsuji, "Charge trapping properties of ultrathin nitrided oxides prepared by rapid thermal annealing", IEEE Trans. Electron Dev., Vol. 35, 904, 1988.

[4] S.K. Lai, D.W. Dong and A. Hartstein, "Effects of ammonia anneal on electron trapping in silicon dioxide", J. of Electrochem. Soc., Vol. 129, 2042, 1982.

[5] G.J. Dunn, R. Jayaraman, W. Yank and C.G. Sodini, "Radiation effects in low pressure re-oxidized nitrided oxide gate dielectric", Appl. Phys. Lett., Vol. 52, 1713, 1988.

[6] K. Ramesh, A. Agarwal, A.N. Chandorkar, and J. Vasi, "Role of electron traps in the radiation hardness of thermally nitrided silicon dioxide", IEEE Electron Dev. Lett., EDL-12, 658, 1991.

[7] J.T. Yount, P.M. Lenaham, G.J. Dunn, "Electron spin resonance study of radiation induced point defects in nitrided oxide", Presented in 1992 NSERC conference.

[8] J.D. McBrayer, D.M. Fleetwood, R.A. Pastorek, and R.V. Jones, "Correlation of hot carrier and radiation effects in MOS transistors", IEEE Trans. Nucl. Sci., NS-32, 3935, 1985.

[9] L. Palkuti, R.D. Ormond, C.M. Hu, and J. Chung, "Correlation between channel hot electron degradation and radiation induced interface trapping in MOS devices," NS-36, 2140, 1989.

[10] C.Hu, S. Tam, F.C. Hsu, P.K. Ko, T.Y. Chan, and K.W. Terrill, "Hot electron induced MOSFET degradation-Model, monitor and improvement", IEEE Trans. Elect. Dev., ED-32, 375, 1985.

[11] N.C. Das, V. Nathan, S. Dacus, and J. Cable, "Combined effects of hot carrier stressing and ionizing radiation in SiO_2, NO, ONO MOSFETs", IEEE Electron Dev. Lett., EDL-14, 40, 1993.

[12] G. Groeseneken, H.E. Maes, N. Beltran, and R.F. De Keersmaecker, "A Reliable Approach to Charge Pumping Measuements in MOS Transistors," IEEE Trans. Elect. Dev., Vol. 31, 42, Jan. 1984.

[13] N.S. Saks and M.G. Ancona, "Generation of interface states by ionizing radiation at 80 K measured by charge pumping and subthreshold technique", NS-34, 1348, 1987.

[14] M. Severi, L. Dori, M. Impronta, and S. Guerri, "Process dependence of hole trapping in thin nitrided SiO_2 film," IEEE Trans. Electron Dev., ED-36, 2447, 1989.

[15] J.D. McBrayer, R.A. Pastorek, and R.V. Jones, "A model describing hot carrier and radiation effects in MOS transistors," IEEE Trans. Nucl. Sci., NS-34, 1647, 1987.

Comparison of Single Event Upset Rates for Microelectronic Memory Devices During Interplanetary Solar Particle Events

P. L. McKerracher, J. D. Kinnison, and R. H. Maurer
Johns Hopkins University Applied Physics Laboratory
Johns Hopkins Road
Laurel, MD 20723
(301) 953-5000 ext 4474; fax (301) 953-6696
Mckerrac@spacemail.jhuapl.edu

Abstract - **Variability in the methods and models used for single event upset calculations in microelectronic memory devices can lead to a range of possible upset rates. Using heavy ion and proton data for selected DRAM and SRAM memories, we have calculated an array of upset rates in order to compare the Adams worst case interplanetary solar flare model to a model proposed by scientists at the Jet Propulsion Laboratory. In addition, methods of upset rate calculation are compared: the Cosmic Ray Effects on Microelectronics CREME code and a Monte Carlo algorithm developed at the Applied Physics Laboratory. The results show that use of a more realistic, although still conservative, model of the space environment can have significant cost saving benefits.**

1 Introduction

The accuracy of single event upset calculations is limited by uncertainties in the environmental models available to the radiation engineer. Although the near earth models of the radiation belts are fairly well-defined, the high level of variability in the interplanetary charged particle environment during solar flares makes upset rate calculations for this region particularly challenging. Even Adams cautioned that his traditional worst case composite flare spectra as presented in 1984 imposed extreme demands on spacecraft design. It is crucial that the particle spectra used in error rate calculations be conservative, but not overly so. If these spectra are unduly conservative, the error rate estimates will be overly pessimistic, and some devices which may be usable in orbit will be disqualified. A comprehensive analysis by Feynman and others of solar events from 1956 to 1985 and a study of large events during the most recent solar maximum (1989-1991) has initiated significant revisions in the modeling of the solar flare environment. The revised model presents a more realistic, although still conservative model of solar particle events.

2 Adams Model Review

The original energy versus Linear Energy Transfer (LET) spectra for solar flare particle events was presented by Adams and Gelman in 1984. They formed a composite worst case proton differential energy spectrum by combining the August 4, 1972 and February 23, 1956 events. The former benchmark event had the highest fluxes at proton energies up to 100 MeV, while the 1956 had the hardest spectrum (high fluxes at high energy) ever recorded. They assumed that this composite flare was also heavy ion rich, used the heavy ion to proton ratios of some of their earlier work (1981) and derived integral LET spectra for all bombarding particles for this worst case composite event. In a sense, since the 1972 and 1956 integral proton spectra formed an upper bound for the spectra of all other flares, the original Adams model, in retrospect, appears to have been a worst-worst case model which consequently would yield very pessimistic Single Event Upset (SEU) predictions.

Energetic particle abundance data from ISEE 3 reported by Reames, Caine and von Rosenvinge (1990) contradicted the Adams model assumption that the solar particle events with the highest proton fluxes were also heavy ion rich. Indeed, Reames et. al. showed that there was an inverse correlation between proton intensity and the iron/carbon abundance ratio. Events with high Proton intensities are very low in iron rich events; iron poor events are often large proton events. In addition, the proton/helium (H^4/He) ratio for large proton events is often an order of magnitude larger than the solar coronal value of this ratio.

Uncertainties in the 1984 Adams model, combined with a wealth of new available particle data from interplanetary space, motivated scientist at the Jet Propulsion Laboratory (JPL) to complete a comprehensive review of existing data in order to generate more realistic solar flare models for interplanetary space. J. Feynman initiated this work by compiling data on proton fluences during solar flares.

3 Feynman Model

Until quite recently, the high-energy solar proton fluence model used most often in total dose calculations was that developed by King. But this model was designed specifically to make predictions about the twenty-first solar cycle during the period 1977-83 on the basis of data from the twentieth solar cycle. Feynman et al. have now shown that the assumptions made by King for the twenty-first cycle invalidate the use of his model for any other solar cycle, particularly since the twentieth cycle contained only one large event (the famous August 1972 event). The distribution of proton fluences greater than 10 MeV was such that a single mathematical distribution could not be used to characterize both the twenty-four ordinary events and the one anomalously large event of the twentieth cycle. Feynman and colleagues have reviewed all the energetic solar proton data from 1956 to 1985, which represented three complete solar cycles. They used observations from riometers, rockets, and balloons for 1956 through 1962 , and interplanetary spacecraft data from the Interplanetary Monitoring Platform (IMP 1 through 3 and 5 through 8), in addition to data from the Orbiting Geophysical Observatory (OGO 1) for 1963 through 1985. Events were defined as having proton fluence larger than 107 particles/cm2 for energies >10 MeV, or proton fluence larger than 105 particles/cm2 for energies >30 MeV. The events totaled 120 above the first energy and 140 above the second.

Significantly, the distinction between ordinary and anomalously large events disappears when all three solar cycles are combined; that is, the August 1972 event is no longer lies outside the proton fluence distributions. The 1960 event actually has the highest proton fluence, and two 1989 events in cycle twenty-two, those of August and September, have >10

MeV proton fluences within factors of 4 and 2 respectively, of the 1972 event. The 1989 events were not included in the original modeling by Feynman et al., but helped validate their conclusions.

An epochal analysis of the solar cycle variation of the annual flux was also carried out. The results show that the solar cycle can be divided into four inactive years of low annual fluence and seven active years with annual fluences greater than 5×10^7 particles/cm2 (>10 MeV). The active period extends from two years before sunspot maximum to four years after maximum. The times of sunspot maximum determined by Feynman et al. are 1957.9, 1968.9, 1979.9, and 1990.9 for the nineteenth, twentieth, twenty-first, and twenty-second cycles, respectively. Active years proved to have fluences greater than 5×10^7 particles/cm2 (>10 MeV), even if no major proton event occurred during that particular year.

When we evaluate the distribution of proton event fluences, we find that minimum or inactive years have such low fluence that only the seven active years need to be modeled. This result enables Feynman et al. to achieve a better statistical fit with a log normal distribution to events, with proton fluences larger than the model fluence. No division between ordinary and anomalous events exists, and events from several solar cycles can be used to obtain an adequate sample of event fluences.

Using the log normal distribution for large events, Feynman et al. performed a Monte Carlo calculation to determine the probability that a given spacecraft mission will experience a solar proton fluence above a given level. These probabilities were calculated for mission lengths of one to seven active years. The >10 MeV fluence expected with the new model is about twice that expected by the earlier models; the >30 MeV fluence is about the same. For example, for a mission lasting one solar active year, the chance of seeing >10 MeV fluence exceeding 7.7×10^{10} particles/cm2 is 2%, while for seven active years, the probability increases to 40%.

The effect of these new results on spacecraft hardware is seen primarily on the least shielded subsystem. Primarily, the increased >10 MeV proton fluence will cause greater proton-displacement-damage degradation of solar cells. Since the >30 MeV fluence is about the same as that of the older models, the total dose and single-event effects predicted owing to this more penetrating component of the solar proton spectrum will be quite similar to predictions of the older models.

4 Re-evaluation of Heavy Ion Models

The results from the ISEE 3 data had led workers to believe that the heavy ion component of a solar flare event should be tied to the alpha particle or helium spectrum for such an event and not to the proton spectrum. Spitale, Feynman and Wang (1992) then conducted a survey of solar alpha particle data using results from the IMP series of spacecraft, using the methodology that Feynman et al. (1990) had used for solar protons. The progress of this solar alpha particle investigation yielded excellent statistical distributions for hourly, daily and event fluences for alphas of energies greater than 1, 4, 10 and 30 MeV respectively. However, the magnitude of the fluences were in disagreement by a constant value, different for each of the distributions, from a smaller, but earlier work by Goswani et al. (1988) on solar alpha particles.

2.4.4

The JPL research team found a solution to the abundance or fluence magnitude problem by using heavy ion (Z>2) data. Experience gained modeling heavy ion data from the Galileo spacecraft for the October 1989 solar events (Garrard and Stone 1991) was used to model solar flare heavy ions, in general. In particular, the heavy ion energy spectrum was not a power law. Average heavy ion fluence ratios derivable from solar photospheric abundances in the atomic number range $6 \leq A \leq 30$ were available from Breneman and Stone (1985). Additional abundance data for $2 \leq Z \leq 5$ and $Z>30$ were available from Anders and Grevese to give a complete abundance model for the range $2 \leq Z \leq 90$. The ratio of the heavy ion fluence to the alpha particle fluence is approximately 3% (2.85-3.35%). A major part of this investigation was to use the results of Brenemans thesis (1985) on the effects of propagating ions of various charge to mass ratios and first ionization potentials through the solar photosphere and corona. Other data on First Ionization Potentials for helium from Reames et al. (1991) was combined with Brenemans work. The significant result of these studies was that the process of fractionalization of heavy ions taking place in the sun reduces the abundance of heavy ions with large Linear Energy Transfer values in interplanetary space.

The team of Croley, Cherng and Spitale combined these results into the JPL Solar Energetic Particle Event Environment Model. A normalization factor, J_0, was introduced to rectify the helium fluence to the heavy ion composition data. J_0 was correlated with the helium ion fluence statistical distributions of Spitale et al (1992). Therefore, by knowing the value of J_0 for a given upset rate calculation, one can determine the upset rate in a milder or more severe solar particle environment just by a simple scaling which is dependent on the level of certainty or amount of risk one is willing to take for a given mission.

Finally, the JPL model contains a exponential time dependence with a decay time of 20 hours. An entire solar particle event can be modeled as on flare with an exponential time decay.

In summary the new JPL model contains:

1. A statistically based model of solar event proton and helium fluence and flux

2. A composition model which ties all heavy ions to the helium flux

3. A characterization of the flare in the time and space domain

4. An energy dependence that is not a power law.

Importantly, the analyst or designer can now calculate an upset rate for a chosen design case event. That is, he/she can design to a given level of certainty for not exceeding a given particle flux simply by choosing the relevant value of J_0.

4.1 Heinrich Flux Spectra

By choosing J_0, a given heavy ion flux/fluence spectrum in interplanetary space is chosen. This spectrum must then be propagated through spacecraft shielding to the location of the integrated circuit. The result is a Heinrich spectrum of heavy ion fluence or flux versus Linear Energy Transfer at a given shield depth. This is the spectrum needed to help calculate single event effect rates when combined with the upset cross section of the device. When the Heinrich spectra for the integral heavy ion flux at 1 AU (Astronomical Unit) from the sun are compared, the JPL model yields values that are three to four orders of magnitude lower than those of the Adams worst case environmental model. In order to evaluate the usefulness of this new model, we calculated upset rates using the new JPL Heinrich spectra and compare them to calculations based on the original Adams model.

5 Upset Rate Calculations

As mentioned above, an estimate of the particle spectra to which a device is exposed is only part of the information needed to calculate an upset rate for the device; a measure of the sensitivity of the integrated circuit to the presence of charged particles is also required. For a monoenergetic beam of particles, the number of errors is directly proportional to the number of particles which pass through the device. When the number of particles is expressed as a fluence (i.e., in units of particles per unit area), the constant of proportionality is called the upset cross-section. The cross-section depends on the beam LET and angle of incidence. This dependence is measured by exposing a device to many different beams over a wide range of LET and incident angle.

The cross-section describes in a macroscopic sense the microscopic behavior of an integrated circuit in the presence of charged particles. Two values that are useful in upset rate modeling are the threshold LET, the minimum LET for which errors occur, and the maximum cross-section, sometimes called the asymptotic value. These numbers are used to calculate physical parameters for input into some upset rate models.

6 Pickel-Blanford Model

Single event upsets due to heavy ions were first observed in memories, and these devices are still among the most sensitive to the effects of charged particles. They are also the simplest to model. An ideal memory is made of an array of cells, each of which responds in exactly the same manner to the presence of a heavy ion. In this case, the cross-section is a step function with a value equal to the asymptotic cross-section above the LET threshold. Microscopically, the picture is equally simple. Each cell contains a sensitive region in which a charged particle with LET above the threshold is able to upset the cell. An ion which passes outside the sensitive region is not able to cause an upset. The area of this sensitive region is given by the asymptotic cross-section. In addition, the threshold LET determines the amount of charge which must be deposited in order to upset the cell; this is called the critical charge.In this picture, the upset rate is given by

$$N = 22.5\pi\sigma_A Q_c \int_{L_T}^{L_{max}} \frac{1}{L^2} D[p(L)]F(L)dL$$

where N is the upset rate, σ_A is the asymptotic cross-section, Q_C is the critical charge, L is LET, and $F(L)$ is the expected flux spectrum. The functional $D[p(L)]$ is the differential distribution of path lengths through the sensitive region over which a particle of a given L will produce more than the critical charge. The path length distribution is quite complex for any regions other than simple geometric shapes. In order to evaluate this integral, the sensitive region is assumed to be a rectangular parallelepiped. Even when simplifying assumptions about the shape of the sensitive region are used, the integral given above usually can not be evaluated analytically, but must be approximated numerically. The most common computer code for this task is CREME.

The Pickel-Blanford model is widely used for upset rate estimation, but has several deficiencies. First, the representation of a device as an array of identical elements is usually not true, especially if the model is applied to devices like microprocessors. Second, cells do not contain a single regularly shaped sensitive volume. Finally, the determination of critical charge and the dimensions of the sensitive region from the cross-section data requires information about the architecture of a device that a user may nor have; wrong guesses of the thickness of the sensitive region, for instance, can lead to large errors in the calculated upset rate. In spite of the deficiencies of the Pickel-Blanford model, it is the basis for nearly all work in single event upset rate calculation, and several refinements of the methodology have been discovered which correct some of the problems listed above. The reader should consult the IEEE Transactions on Nuclear Science for further details concerning refinements to the Pic kel-Blanford model.

7 Bendel Proton Upset Models

Protons found in the Van Allen belts or in solar events have an LET much less than $1MeV - cm^2/mg$. This is not sufficient to cause upset in most devices. However, a proton traveling through a device can interact with a silicon nucleus, producing low energy, high LET products. These reaction products can ionize enough charge to cause upset.

The Bendel 1-parameter model is based on the observation that in a monoenergetic proton beam, the upset rate as a function of proton energy follows a relationship that looks like the proton reaction cross-section in silicon. Bendel used the threshold energy and the asymptotic cross-section to describe the relationship between the upset rate and the proton energy. To simplify the model, Bendel developed a relationship between the asymptotic cross-section and the threshold energy. The equations used in this model are

$$N = \left(\frac{24}{A}\right)^{14} \left\{1 - e^{(0.18Y)^{1/2}}\right\}^4$$

$$Y = \left(\frac{18}{A}\right)^{1/2} (E - A)$$

where N is the upset rate in units of 10^{-12} upsets per proton/$cm2$ per bit, E is the proton energy, and A is the threshold energy. The parameter A is determined by measuring the upset rate at one or two proton energies and numerically calculating the value of A which gives the best fit to the equations for Y and N. The upset rate in any environment is given by the integral over energy of N times the proton fluence spectrum for that environment.

The 1-parameter model works well for older microelectronics. As the device feature size becomes smaller, deviations from the 1-parameter fit to the cross-section are observed. A 2-parameter model is now used to describe the proton upset cross-section for newer devices.

The equations for the 2-parameter model are identical to the 1-parameter equations except that the constant in the numerator of the first term in the equation for $N(24)$ is replaced by a parameter called B. The values of A and B for a given device are determined by measuring the upset rate for a monoenergetic proton beam over a range of proton energies, and then numerically calculating the values of A and B which give the best fit to the model equations. Again, the upset rate in a given environment is found by integrating N times the proton spectrum as a function of proton energy.

8 Probabilistic Model

The probabilistic model is based on a different interpretation of the upset cross-section. In this interpretation, the cross-section is simply an unnormalized probability that a particle of a given LET and angle of incidence will cause an upset. From this, we can define a Monte Carlo algorithm to calculate upset rates from cross-section data and an estimated fluence. This algorithm is as follows:

1. Generate a incident particle with some LET and random direction.

2. From this LET and direction, calculate the cross-section and divide by the total area of the device.

3. Generate a random number from the uniform distribution in the interval from 0 to 1. If this number is less than the normalized cross-section, count it as an upset.

4. Repeat for enough particles to match the input fluence spectrum.

A primary advantage of this model is that information about the device architecture is not needed to complete the calculation; the device is treated as a "black box" completely characterized by the cross-section data. However, the accuracy of the calculation depends on the quality of the random number generator used. Cross-sections can be quite small, and so the random number generator must produce numbers with resolution better than the smallest value of the cross-section in the data. If this is not the case, significant bias may enter the calculation, and the results may be incorrect by several orders of magnitude.

9 Applications

Using the CREME suite of software codes, we generated the Adams model LET spectra assuming 60 mils of aluminum shielding. An APL developed code written in C converted the JPL supplied J0 = 1 model to a J0 = 3 Table in the correct format to interface with the CREME UPSET code. We obtained sample SEU cross-section data from the work of Harboe-Sorensen et al. for DRAMS, Hitachi SRAM data from the work of LaBel et al. and Field Programmable Gate Array (FPGA) data from Kinnison (1991). In order to calculate the critical charge for upset we considered the LET upset threshold to be that at which the cross-section is 0.4 times the saturation level. This method accounts for the fact that the cross-section data is not a perfect step function. The critical charge in pC is then estimated as the depth of the critical volume times the LET threshold times 0.01. The x and y lateral dimensions of the critical volume are obtained from using the square root of the saturation cross-section, sigma, and the depth is based on val ues obtained by earlier works, when manufacturing process information was not available. Dimensions are necessary because the CREME UPSET routine assumes a parallelepiped geometry for the critical volume susceptible to upset.

Because the memory devices all had onset LET values below 4 MeV-cm2/mg it was necessary to perform calculations for indirect ionization upset induced by protons. As the onset of upset for the A1020 is about 20 MeV-cm2/mg, it is not susceptible to proton induced upset. Proton upset rates were obtained using proton data where available and the CREME BENDEL routine. Where no proton data was available, an estimate was made assuming the Bendel A parameter to be the 0.1 LET threshold value plus 15.

Table 1 presents the results for the nine memory devices and one FPGA included in the study. The upset rates are in upsets/bit-sec and upsets/bit-day in parentheses. Upset calculations were made for several different cases. Heavy ion upset rates were calculated for the Adams model under four different conditions for Interplanetary space. Cases 1. and 4. do not appear in the table:

1. Galactic Cosmic Rays only (quiet time environment, M=3) 2. The 90% worst case interplanetary solar flare environment (CREME M=8) 3. Peak Aug. 4, 1972, flare flux and worst-case composition (CREME M=10) 4. Peak composite worst case flare flux and worst case composition solar flare environment (CREME M=12)

In Table 2 these rates are compared with each other and with the heavy ion upset rate calculated using the JPL J0=3, 99% probability environmental model. The worst case solar flare upset predictions in the M=10 environment are typically at least 100,000 times greater than the quiet time rates. Essentially no difference was found in the upset rates of the M=10 and M=12 cases, therefore only the M=10 rates are listed in the table. The M=10 rates were 46 to 54 times greater than the M=8 rates. Clearly, this is where the difficulty arises. Using the lower upset rates in system design can have important consequences, particularly in the design of Error Detection and Correction codes used to counteract SEU effects in memory devices. But how closely does the Adams 90% curve emulate actual solar flare environment? Comparing the JPL J0=3 rates with the Adams M=8 rates answers our question. The JPL J0=3 rates are 2 to 14 times greater; the M=8/J=3 ratios range from 0.07 to 0.53. Correspondingly, the M=10/J=3 ratios r anged from 3.3 to 29. This leads us to believe that the Adams 90% curve is sometimes a more reasonable environment to assume and gives us closer to 99% confidence that our spacecraft will not experience a flare epoch with greater fluxes.

For the soft commercial memories sampled it is important to note that in most cases the heavy ion induced rates dominate over the proton induced SEU rates. The solar flare proton M=10 to heavy ion J0=3 ratios varied from 0.002 to 8. In the extreme cases the steepness of the Heavy Ion data predicts the results. The NEC DRAM is seen to have the steepest heavy ion curve from reviewing the LET threshold value of 4 MeV-cm2/mg in Table 1; the heavy ion effects at the lower LET values therefore contribute enough in this case that the Heavy Ion rate dominates the proton induced rates by 500 times. On the other hand, the Samsung DRAM has the most gently sloping cross-section profile. Consequently, it has the largest estimated critical charge and the lowest heavy ion upset rate. Here the proton rate is estimated to be twice the heavy ion rate, therebye dominating the total rate, which is the sum of the heavy ion and proton rates.

The Actel FPGA however, is not susceptible to proton upset. For such devices it is important then to select the correct heavy ion SEU rate. The M=10 Adams worst case rate here is 16 times higher than that calculated using the JPL model, because of the reduced high LET fluence due to solar fractionalization included by the JPL researchers.

10 Conclusions

Much of our work is intended for use in designing the Advanced Composition Explorer (ACE) spacecraft which will orbit the Earth/Sun L1 libration point. The spacecraft instruments are primarily designed to study solar events, so that ACE systems must continue to operate during the peak of an event. Therefore, SEU susceptibility is of primary concern. Application of newer, more realistic solar event models potentially allows more frequent use of new, higher density digital devices in ACE and similar satellite systems. For devices that are susceptible to proton upset, the new heavy ion model is especially relevant if the proton induced upset rate does not dominate the total upset rate for the part. It is therefore important to have good proton upset data. It would also be useful to re-evaluate the proton environment used in CREME. In devices which are not susceptible to proton upset, the JPL heavy ion model will yield significantly less pessimistic upset rates.

We intend to expand on this work by calculating upset rates based on the Monte Carlo based Probabilistic Model mentioned. We wish to use this method to overcome the limitations and uncertainties of the CREME calculations, which depend upon estimates of the critical volume dimensions and the assumption that the critical volume is a parallelepiped geometry. Normally much of this information is not available to the end user of the manufactured integrated circuit.

Acknowledgements: A special thanks to G. Murphy, M. Ratliff, and D. Croley of the Jet Propulsion Laboratory in Pasadena, California for suppling information on the models and for assistance via email!

References

[1] R. D. Belian, G. R. Gisler, T. Cayton & R. Christensen, High-Z Energetic Particles at Geosynchronous Orbit During the Great Solar Proton Event Series of October 1989, Journal of Geophys. Res., Vol. 97, No. A11, pgs. 16,897-16,906, Nov. 1, 1992.

[2] L. J. Lanzerotti, C. G. McMillan, R. E. Gold, S. E. Hawkins III, S. J. Tappin & R. Forsyth, Solar Particle Composition: Measurements in the March 1991 Event at 2.5 AU, , S. J. Tappin, & R. Forsyth, Geophys. Res. Letters, Vol. 19, No. 12, pgs. 1251-1254, June 19, 1992.

[3] D. V. Reames, H. V. Cane & T. T. von Rosenvinge, Energetic Particle Abundances in Solar Electron Events, The Astrophys. J., 357:259-270, 1990 July 1.

[4] J. H. Adams, Jr. & A. Gelman, The Effects of Solar Flares on Single Event Upset Rates, IEEE TRANS. on NS, Vol. NS-31, No. 6, Dec. 1984.

[5] D. F. Smart & M. A. Shea, Solar Proton Events During the Past Three Solar Cycles, Spacecraft, Vol. 26, No. 6, pg. 403-415, Nov.-Dec. 1989.

[6] H. H. Breneman, Solar Photospheric and Coronal Abundances From Solar Energetic Particle Measurements, Thesis, Cal. Inst. of Tech. 1985, SRL 85-3.

[7] D. R. Croley to G. B. Murphy, Spectral Shape of Solar Flare Heavy Ions, JPL IOM 5215-92-075, July 21, 1992.

[8] T. L. Garrard & E. C. Stone, Heavy Ions in the October 1989 Solar Flares Observed on the Galileo Spacecraft, Cal. Inst. of Tech. Pasadena CA 91125, SH 5.1.5.

[9] D. R. Croley to G. B. Murphy, Solar Flare Heavy Ion Model, JPL IOM 5215-92-042, June 22, 1992.

[10] E. Normand and W. J. Stapor, Variation in Proton-Induced Upset Rates From Large Solar Flares Using an Improved SEU Model, IEEE Trans. on NS, Vol. 37, No. 6, pgs. 1947-1952, Dec. 1990.

[11] W. J. Stapor, J. P. Meyers, J. B. Langworthy & E. L. Peterson, Two Parameter Bendel Model Calculations for Predicting Proton Induced Upset, IEEE Trans. on NS, Vol. 37, No. 6, pgs. 1966-1973, Dec. 1990.

[12] D. R. Croley to G. B. Murphy, Single Event Upset (SEU) Rate Per Fluence, TDRS-1 ACS CPE RAM (Fairchild 93l422), JPL IOM 5217-92-21, Jan. 20, 1992.

[13] L. C. Simonsen, J. E. Nealy, H. H. Sauer & L. W. Townsend, Solar Flare Protection for Manned Lunar Missions: Analysis of the October 1989 Proton Flare Event, 21st International Conference on Environmental Systems, July 1991.SAE Technical Paper Series # 911351.

[14] G. D. Reeves, T. E. Cayton, S. P. Gary, & R. D. Belian, The Great Solar Energetic Particle Events of 1989 Observed From Geosynchronous Orbit, J. of Geophys. Res. Vol. 97, No. A5, Pgs 6219-6226, May 1, 1992.

[15] J. Feynman, T. P. Armstrong, L. Dao-Gibner & S. Silverman, New Interplanetary Proton Fluence Model, J. Spacecraft, Vol. 27, No. 4, pgs. 403-410, Jul-Aug. 1990.

[16] A. B. Campbell, SEU Flight Data From the CRRES MEP, IEEE Trans. on NS, Vol, 38, No. 6, pgs. 1647-1651, Dec. 1991.

[17] D. L. Chenette & W. F. Dietrich, The Solar Flare Heavy Ion Environment for Single-Event Upsets: A Summary of Observation Over the Last Solar Cycle, 1973-1983, IEEE Trans. on NS, Vol. NS-31, No. 6, pgs. 1217-1222, Dec. 1984.

[18] D. R. Croley & M. Cherng to G. B. Murphy, Procedure for Specifying the Heavy Ion Environment at 1 AU, JPL IOM 5215-92-072, July 21, 1992.

[19] M. Ratliff to L. Edmonds, 1992 Model of Solar Flare Heavy Ions: Selected Spectra, JPL IOM 5215-92-143, Sept. 28, 1992.

[20] D. C. Marvin & D. J. Gorney, Solar Proton Events of 1989: Effects on Spacecraft Solar Arrays, J. Spacecraft, Vol. 28, No. 6, Nov.-Dec. 1991.

[21] R. Koga, et.al, On the Suitability of Non-Hardened High Density SRAMs For Space Applications, Aerospace Corporation, El Segundo, CA 90245.

[22] J. Feynman, T. P. Armstrong, L. Dao-Gibner & S. Silverman New Interplanetary Proton Fluence Model, J. Spacecraft, Vol. 27, No. 4, pgs. 403-410, Jul-Aug. 1990.

Device	LET th MeV cm/ mg	sigma cm -2/ bit	h,l,w microns = $V\sigma x$	Qc pC hxLET x 1E-2	M=8 90% solar flare adamsm8 int	M=10 worst case solar flare adamm10. int	HI JPL J=3 60 mils Al	protons M=10 60 mils
Hit HM628512 512x8 SRAM	2.8	3E-7	0.5, 5.5, 5.5	0.014	2.52E-7 (2.18E-2)	1.16E-5 (1.00E0)	3.51E-6 (3.03E-1)	no data
Fuj 4Mx 1D MB814100-1PSZ	13	7E-7	0.5, 8.4, 8.4	0.065	6.54E-8 (5.65E-3)	3.54E-6 (3.06E-1)	2.72E-7 (2.34E-2)	1.41E-8
Hit 4MX1 D HM514100ZP8	10	1.5E-7	0.5 3.9, 3.9	0.050	1.92E-8 (1.66E-3)	1.05E-6 (9.14E-2)	6.50E-8 (5.62E-3)	Hi est. 5.22E-7
Micron 4MX1 D MT4C1004C	12	3E-7	0.5, 5.5, 5.5	0.060	2.99E-8 (2.58E-3)	1.64E-6 (1.41E-1)	1.22E-7 (1.05E-2)	1.00E-8
NEC 4MX1 D D424100V-80	4	3.5E-7	0.5, 5.9, 5.9	0.020	1.72E-7 (1.48E-2)	8.49E-6 (7.43E-1)	2.14E-6 (1.84E-1)	4.45E-9
Samsung4MX1D KM41C4000Z-8	28	3E-7	0.5, 5.5, 5.5	0.14	6.76E-9 (5.84E-4)	3.68E-7 (3.18E-2)	1.27E-8 (1.10E-3)	Hi est. 2.91E-8
Siemens 4MX1D HYB514100J-10	12	4.5E-7	0.5, 6.7, 6.7	0.060	5.45E-8 (4.71E-3)	2.90E-6 (2.50E-1)	1.92E-7 (1.66E-2)	2.74E-8 (2.37E-3)
T1 4MX1 D TMS44100DM-80	8	3.5E-7	0.5, 5.9, 5.9	0.040	6.18E-8 (5.34E-3)	3.36E-6 (2.90E-1)	4.19E-7 (3.62E-2)	1.60E-7 (9.13E-3)
Toshiba 4MX1 D TC514100Z-10	13	5E-7	0.5, 7.1, 7.1	0.065	4.72E-8 (4.07E-3)	2.55E-6 (2.21E-1)	1.83E-7 (1.57E-2)	1.41E-8 Hi est. 5.22E-7
Actel A 1020 FPGA	25	2E-6	0.5, 14, 14	0.125	5.97E-8 (5.16E-3)	3.22E-6 (2.78E-1)	2.02E-7 (1.75E-2)	NA

Table 1: Sample SEU rates in upsets/bit-sec for 60 mil shielding, rates in () are in upsets/bit-day

Device	**Heavy Ion** $J_0 = 3/J_0 = 3$ **ratio**	**Heavy Ion** $M = 8/J_0 = 3$ **ratio**	**Heavy Ion** $M = 10/J_0 = 3$ **ratio**	**proton to Heavy Ion** $M = 10/J_0 = 3$ **ratio**
Hit HM628512 512Kx8 SRAM	1	0.07	3.3	no data
Fuj 4Mx1 D MB814100-1PSZ	1	0.24	13	0.05
Hit 4MX1 D HM514100ZP8	1	0.29	16	8(HI est.)
Micron 4MX1 D MT4C1004C	1	0.25	13	0.08
NEC 4MX1 D D424100V-80	1	0.08	4	0.002
Samsung4MX1D KM41C4000Z-8	1	0.53	29	2.3(HI est.)
Siemens 4MX1 D HYB514100J-10	1	0.28	15	0.14
TI 4MX1D TMS44100DM-80	1	0.14	8	0.25
Toshiba 4MX1 D TC514100Z-10	1	0.25	14	0.07
Actel A 1020 FPGA	1	0.30	16	NA

Table 2: Single Event Upset Rate Ratios

STRV RADMON: An Integrated High-Energy Particle Detector

Martin Buehler, George Soli, Brent Blaes, and Gemma Tardio
VLSI Technology Group
Jet Propulsion Laboratory
California Institute of Technology
Pasadena CA 91109

Abstract - **The RADMON (Radiation Monitor) was developed as a compact device with a 4-kbit SRAM particle detector and two p-FET total dose monitors. Thus it can be used as a spacecraft radiation alarm and in situ total dose monitor. This paper discusses the design and calibration of the SRAM for proton, alpha, and heavy ion detection. Upset rates for the RADMON, based on a newly developed space particle flux algorithm, are shown to vary over eight orders of magnitude. On the STRV (Space Technology Research Vehicle) the RADMON's SRAM will be used to detect trapped protons, solar flares, and cosmic rays and to evaluate our ability to predict space results from ground tests.**

1 Mission Overview

The STRV (Space-Technology Research Vehicle) consists of two spacecrafts as shown in Figure 1. Each vehicle is approximately a cube 17 inches on a side that will be launched into a geosynchronous transfer orbit (GTO) with a 200-km perigee, a 36,000-km apogee, and an 11-hour period. This orbit provides the opportunity to profile the Earth's radiation belts.

The STRV contains a number of radiation measuring instruments including the JPL RADMON (Radiation Monitor), the DRA CREDO (Cosmic Environment Dosimeter), and the ESA REM (Radiation Environment Monitor). The CREDO and REM use PIN diodes or surface barrier (SB) detectors with pulse-height detection circuitry to discriminate particle energy. The area of the CREDO and REM detectors is between 22 to 400 mm^2; whereas, the area of the RADMON is only $0.17mm^2$. Thus the upset rates are much higher for the CREDO and REM than for the RADMON.

The purpose of this paper is to present the methodology for calculating the RADMON's upset rate using a newly developed algorithm for the space environment and the RADMON's device parameters. After launch, the upset rate for the CREDO, REM and RADMON will be compared to determine the validity of the ground predictions and the relative sensitivity of the three particle detectors.

The space particles relevant to RADMON upsets are:

- trapped protons (TP)
- galactic cosmic rays (GCRs)
- solar flare particles (SFP).

Electrons in the electron belt will not upset the RADMON. Electrons provide a total dose which causes a slow drift in the RADMON baseline which is constantly monitored.

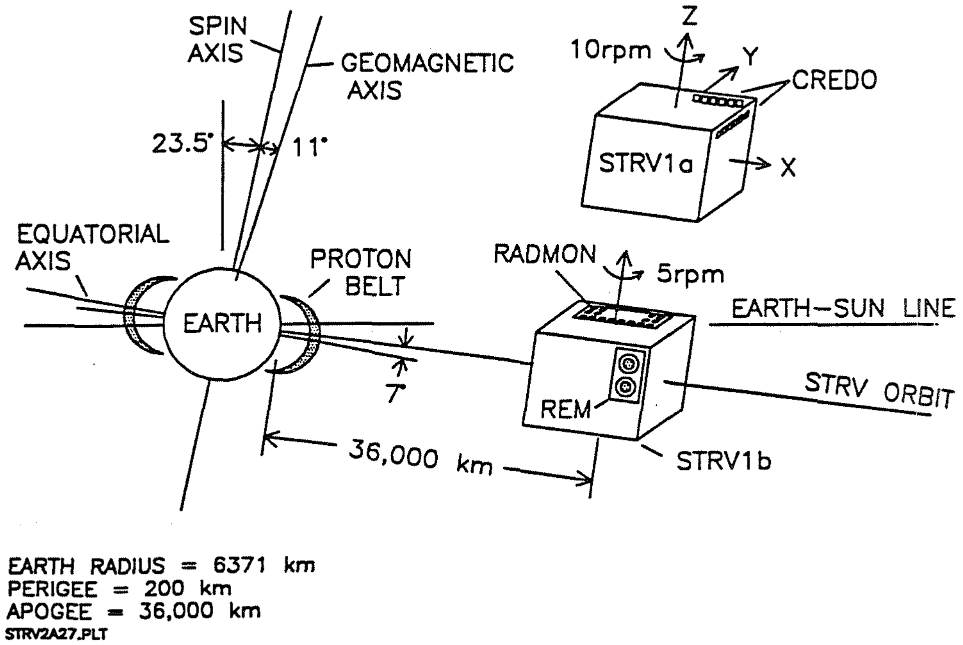


Figure 1: STRV orbit relative to the earth's proton belt

2 SEU SRAM

The RADMON, shown in Figure 2, is a 28-pin chip with a 4-kbit SRAM for SEU (single-event upset) detection and two p-FETs [1] for total dose monitoring. The SRAM contains a six-transistor memory cell which is illustrated in the schematic cross section of Figure 3. This figure shows that the memory cell has been modified in two ways to increase the cell's sensitivity to upsets. First, the drain area, σ_m, of diode Dn2 has been increased to increase its particle capture cross section. Second, the diode Dn2 is biased with an offset voltage, V_o, which is applied through p-FET Mp2. This allows the cell to be operated near its metastable point which in turn allows the cell's upset sensitivity be adjusted via V_o.

The interaction of high-energy particles with the RADMON is described in terms of the various layers the particle passes through as seen in Figure 4. In passing through the layers, the particle looses energy and in general increases its LET (linear energy transfer) or its ability to deposit charge. Thus knowing the thickness of the shield and device overlayer is key to calculating particle upset rates.

The response of the RADMON to protons is illustrated in Figure 5 where 0.55, 1.0, and 2.0 MeV protons upset the cells. The figure also includes the spontaneous upset curve which has a mean value of $V_{os\mu}$. The spontaneous curve defines the metastable point for each of the SRAM cells. This curve represents the baseline for the device.

In operation the RADMON is operated at a V_o greater than $V_{os\mu}$ or $\delta V_o = V_{op\mu} - V_{os\mu}$ where $V_{op\mu}$ is the mean particle offset voltage defined by the $\sigma_m/2$ crossing point shown in Figure 5. For the STRV, the RADMON is operated with three σV_o values corresponding to a proton threshold, an alpha particle threshold, and a heavy ion threshold. These thresholds

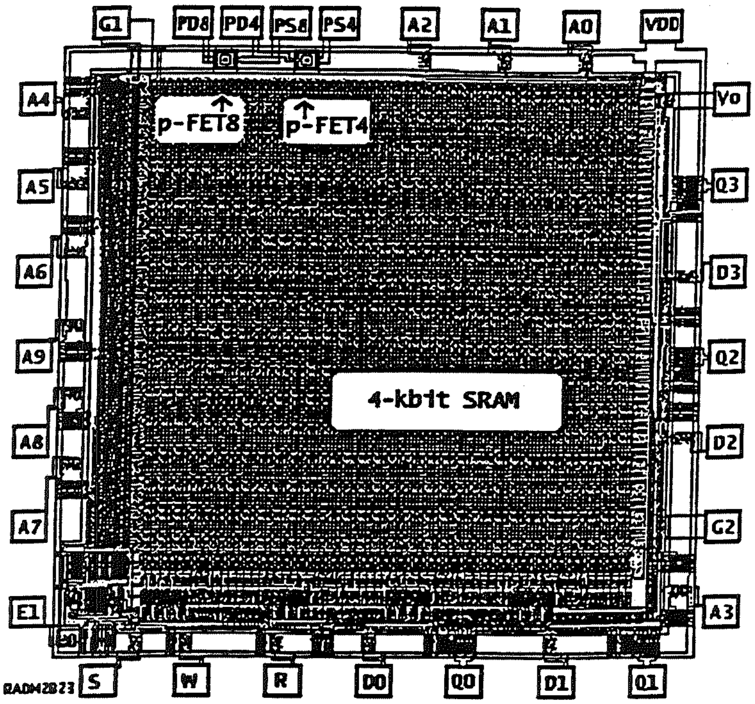


Figure 2: STRV-RADMON chip 2.57 × 2.70mm^2

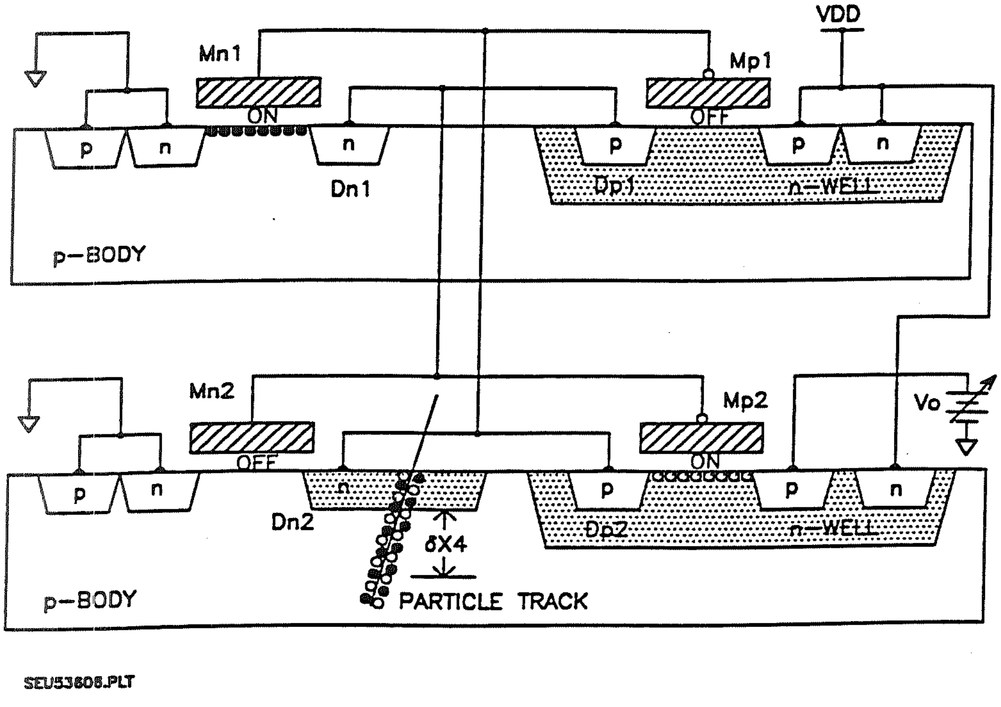


Figure 3: Cross section of SEU-SRAM showing a particle track through diode DN2 which can cause a bit flip if sufficient charge is deposited

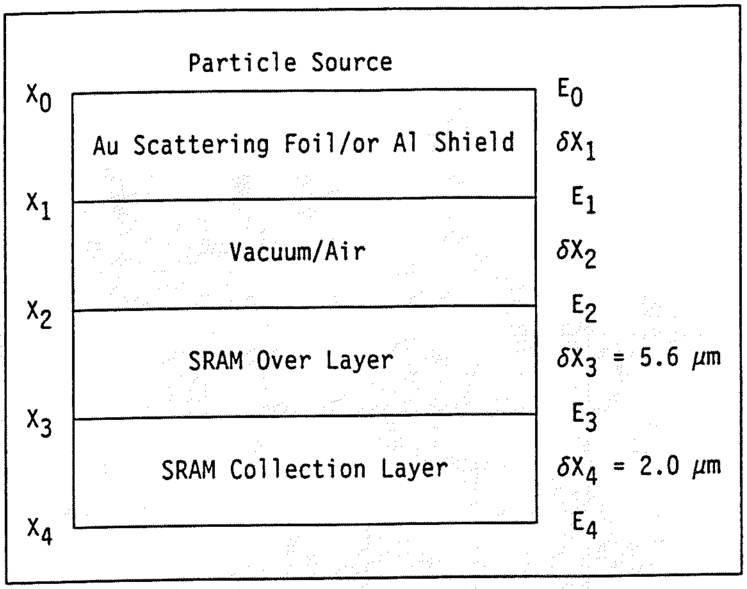


Figure 4: Schematic view of the particle path from the source to the SRAM.

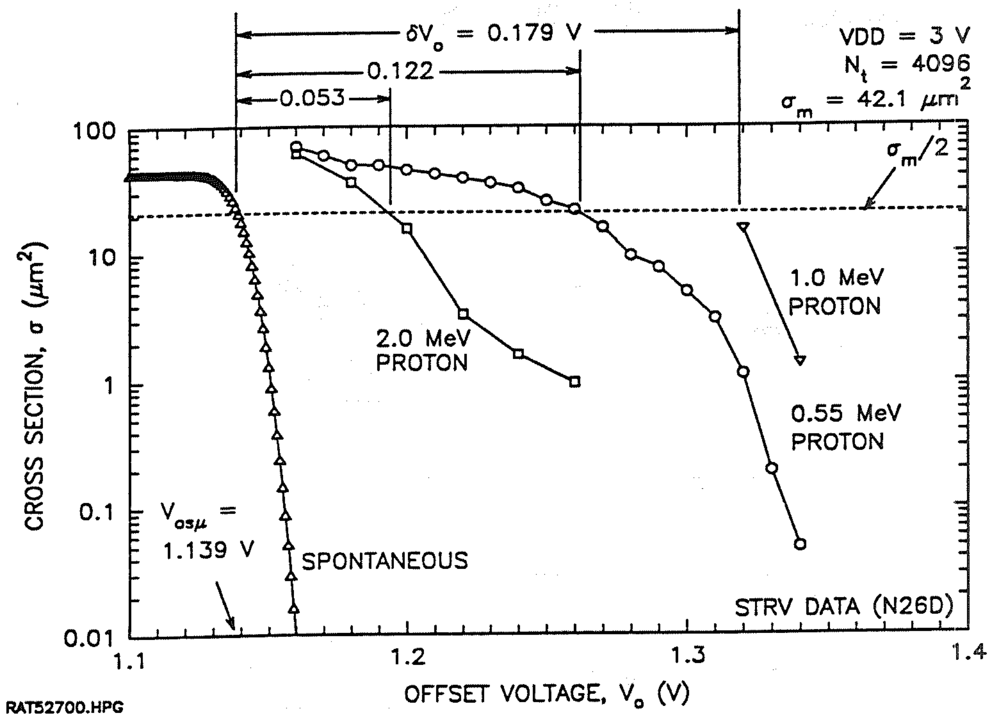


Figure 5: SRAM cross section/bit for protons with three different incident energies ($E_1 = E_2$)

are listed in Table 1.

	VDD	$V_{os\mu}$	V_o	LET
THRESHOLD	V	V	V	$MeV cm^2/mg$
PROTON	3	1.139	0.610	0.086
ALPHA	3	1.139	0.250	0.353
HEAVY	5	1.720	3.280	4.628

Table 1: RADMON Particle Thresholds and LETs.

The vertical axis of Figure 5 displays the particle cross section. It is calculated using:

$$\sigma = N/(\phi_m \cdot t_m \cdot N_t) \tag{1}$$

where ϕ_m is the measured particle flux, t_m is the measurement of time and N_t is the number of bits in the SRAM in this case $N_t = 4096$. This equation is applicable for $N \ll N_t$ which is satisfied when the number of flipped bits is less than 10 percent of N_t. This equation indicates that the cross section is proportional to the number of flipped bits, N. The proportionality factor, the denominator, normalizes the data with respect to flux and measurement time.

In order to predict the RADMON's upset rate, $\sigma_m, \delta Q_c$, and $\delta X4$ must be known. The device cross section is taken from the physical layout and in this case $\sigma_m = 42.1 \mu m^2$. The critical charge is related to V_o through:

$$\delta Q_c = C_u \cdot \delta V_o \tag{2}$$

where C_u is the upset capacitance. The collection depth, $\delta X4$, is shown in Figure 3. Both C_u and $\delta X4$ are determined experimentally and in this discussion $C_u = 28.9 fC/V$ and $\delta X4 = 2 \mu m$ are used.

The RADMON's proton response, shown in Figure 5, reveals a non-monatonic behavior between the particle energy and δV_o. This behavior is explained by analyzing the proton charge depth curves [2] shown in Figure 6. The vertical lines indicate the width of the overlayer and the width of the collection depth; that is, $\delta X_3 = 5.6 \mu m$ and $\delta X4 = 2 \mu m$. The figure shows that the 0.55 MeV proton stops in $\delta X4$ and the 1.0 and 2.0 MeV protons stop beyond $\delta X4$. The 0.2 MeV proton stops in the overlayer so it can not be detected. The explanation for the non-monatonic energy versus δV_o behavior follows from the charge deposited in the collection layer, δQ_c, as indicated in Figure 6. This charge is proportional to δV_o as given in Equation 2 and thus explains the non-monatonic behavior.

It is concluded that the SRAM detects protons in the 0.5 to 2.0 MeV range. Higher energy protons can be detected only if their energy is brought in to the detection range by using a shield which drops the proton energy into the 0.5 to 2.0 MeV energy range. The STRV contains 16 RADMONs; half are shielded by a 2.0-mm Al shield and the other half are shielded by a 5.5-mm Al shield. The effect of shielding on the detection of protons is discussed in a later section.

C-2

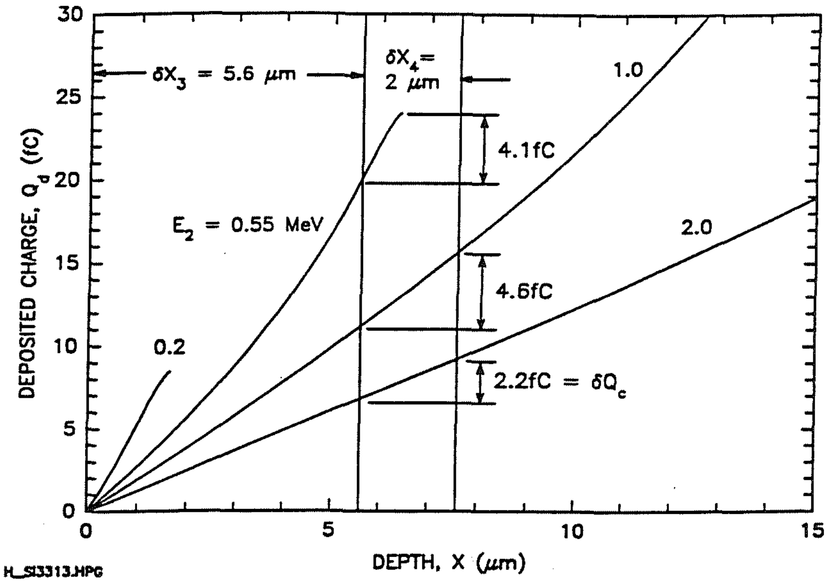


Figure 6: Proton charge deposited in silicon [2]

3 Galactic Cosmic Ray and Solar Flare Particle Upsets

Galactic cosmic ray (GCR) and solar flare particle (SFP) upsets are illustrated in Figure 7. These curves were obtained by fitting Adam's curves [3]; the GCR are for solar minimum conditions. The curves show that shielding has a significant effect on the SFPs but relatively little effect on the GCRs.

The number of upset per SRAM is given by:

$$U = N \cdot \Omega \cdot t \cdot \sigma_m \cdot F \tag{3}$$

where N is the number of bits per SRAM, Ω is the solid angle, t is the observation or stare time, σ_m is the area sensitive to upsets, and F is the integral particle flux. The Petersen Equation[4] is commonly used to calculate SRAM upsets. The flux expression is:

$$F(\frac{1}{\mu m^2} \cdot ster \cdot sec) = 4.4 \times 10^{-12}/[L(MeV \cdot cm^2/mg)]^2 \tag{4}$$

where L is the linear energy transfer. The conversion factor for L in silicon is

$$L(MeV \cdot cm^2/mg) = L(fC/\mu m)/10.25$$

where $L(fC/\mu m) = \delta Q_c(fC)/\delta X4(\mu m)$. The Petersen equation holds for a 0.63-mm Al shield in a 10 percent worse case environment (10%WC).

The effect of shielding on the particle flux was obtained by fitting the Adam's data [3]. The flux is:

$$F(1/\mu m^2 \cdot ster \cdot sec) = F_o \cdot (L/L_o)^{a+b \cdot LOG(1+D/D_o)} \tag{5}$$

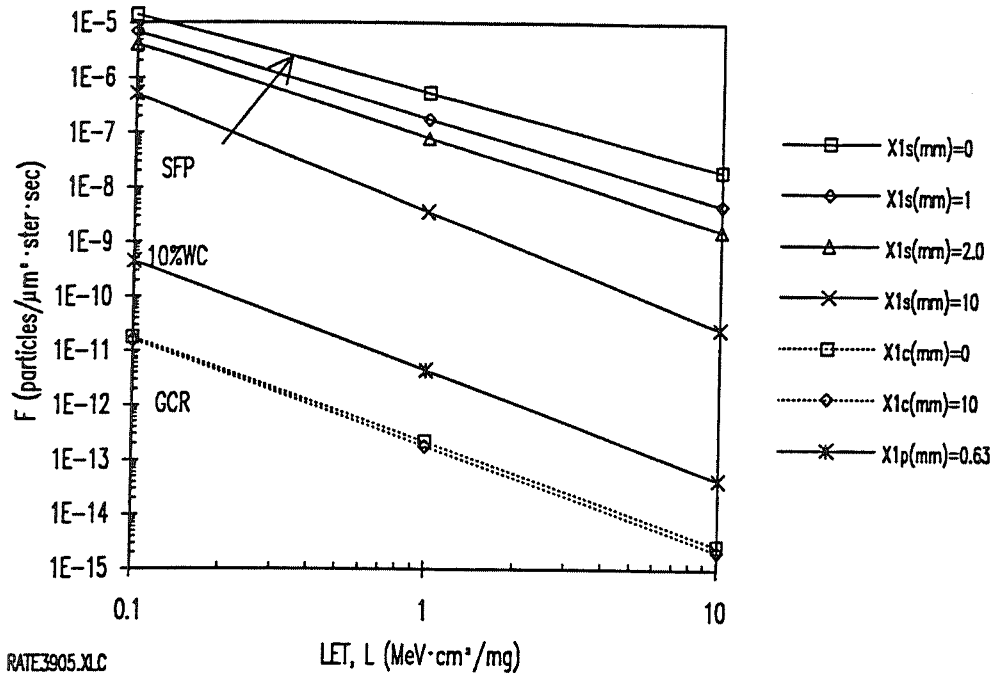


Figure 7: Integral LET interplanetary spectra for various aluminum shields [3] due to galactic cosmic rays (GCR), 10% worse case environment (10%WC) and solar flare particles (SFP).

PARAMETER	UNITS	GCR	LSF
F_o	$1/(\mu m^2 \cdot ster \cdot sec)$	1×10^{-5}	1×10^{-2}
L_o	$MeV \cdot cm^2/mg$	1×10^{-4}	1×10^{-3}
D_o	mg/cm^2	78.36	0.5990
a	unitless	-1.92	-1.425
b	unitless	-1.35	-0.974

Table 2: GCR and LSF Flux Parameters

where D is the shielding thickness in g/cm^2. The shielding thickness is given by $\delta X1 = D/\rho$ where ρ is the shield density in g/cm^2. The values for F_o, L_o, a and b are listed in Table 2.

The upsets expected for the RADMON are shown in Figure 8 where the three LET thresholds are labeled. For the SFPs the RADMON will experience numerous upsets during the 600 sec stare time. For the GCRs, the RADMON will experience less than one upset per 600 sec stare time. The results are summarized in Table 1.

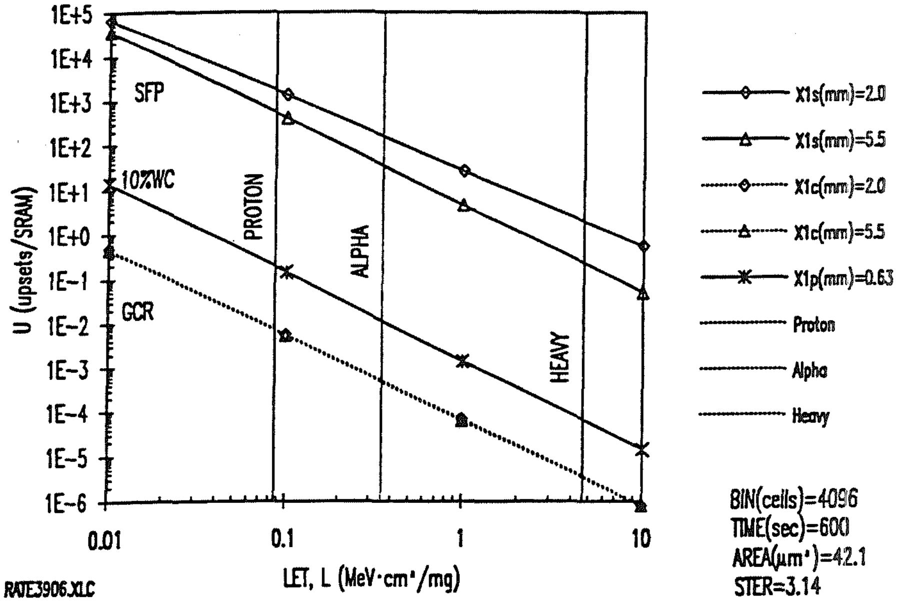


Figure 8: STRV RADMON SEU SRAM upset rate due to galactic cosmic rays (GCR), 10% worse case environment (10%WC), and solar flare particles (SFP).

4 Proton Belt Upsets

The profile of the proton belt is shown in Figure 9. It shows that the STRV remains in the vicinity of proton belt peak for several hundred seconds. Before reaching the RADMON the protons must pass through the Al shields. From the proton range curves [2], it can be shown that the protons must have $E_0 > 18.5$ MeV to pass through a 2.0-mm Al shield and $E_0 > 32.5$ MeV to pass through a 5.5-mm Al shield. The proton energy must be lowered by the shield so that it has an energy between 0.5 and 2.0 MeV when it enters the RADMON's collection in order to be detected. This effect is shown graphically in Figure 10 where two solid horizontal lines have been drawn at the bounds of the RADMON's detection range. The problem now is to determine δE_0, the incremental incident proton energy that can be detected by the RADMON.

The number of protons that upset the RADMON during the stare time, t, is:

$$N = \sigma_m \cdot \phi_d \cdot t \cdot N_t \cdot \delta E_0 \tag{6}$$

where ϕ_d is the differential incident proton flux which is determined from the peak of the proton belt curves shown in Figure 9. For $E_0 < 30$ MeV, $\phi_d = 600p/(cm^2 \cdot sec \cdot MeV)$ and

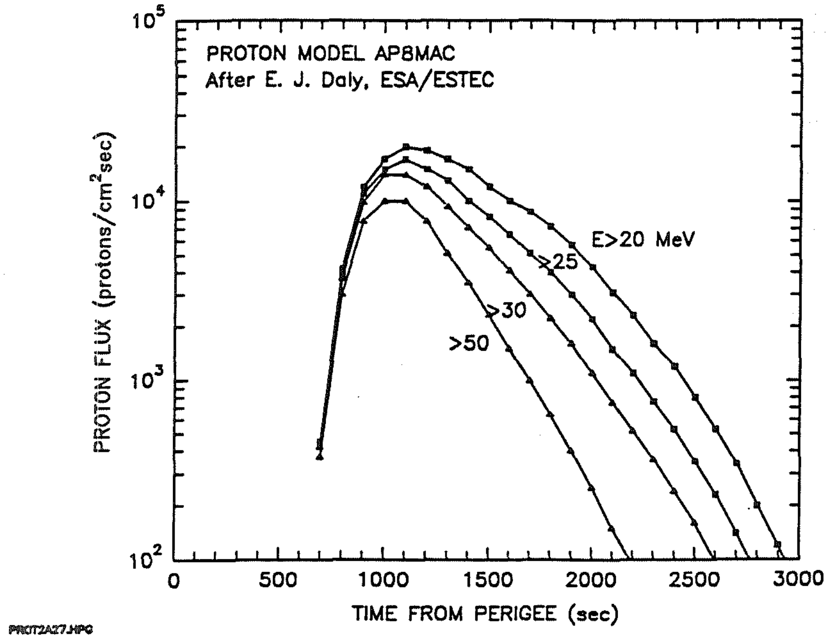


Figure 9: STRV traverse of the earth's proton belt.

for $E_0 > 30$ MeV, $\phi_d = 200p/(cm^2 \cdot sec \cdot MeV)$. The E_0 is given by:

$$\delta E_0 = (dE_0/dE_1)\delta E_1 \tag{7}$$

where δE_1 is the incremental exit proton energy. The dE_0/dE_1 is determined from the proton range curves shown in Figure 10 where $dE_0/dE_1 = 0.36$ for both the 2.0- and 5.5-mm Al shields.

.A value for δE_1 is obtained by assuming that most of the proton energy is lost in the shield and virtually none is lost in the chip overlayer. Thus $\delta E_1 = \delta E_2 = \delta E_3 = 2.0 - 0.5 = 1.5$ MeV. Using Equation 7, $\delta E_0 = 0.54$ MeV for both Al shields. Thus the RADMON will detect protons through the 2.0-mm Al shield with $E_o = 18.77 \pm 0.27$ MeV and through the 5.5-mm Al shield with $E_o = 32.77 \pm 0.27$ MeV. This shows that the RADMON is really and energy detector for high-energy protons.

The number of trapped-proton induced upsets, calculated using Equation 6, is listed in Table 3. It shows that several hundred upsets can be expected near the peak of the proton belt. The above calculation does not include particle straggling in the Al shield. Including this effect will increase the number of upsets.

	PROTON BELT	SOLAR FLARE	10% WORSE CASE	GCR
ENVIRONMENT	U(upsets/SRAM)	U(upsets/SRAM)	U(upsets/SRAM)	U(upsets/SRAM)
Al-SHIELD	2.0-mm 5.5-mm	2.0-mm 5.5-mm	0.63-mm	2.0-mm 5.5-mm
PROTON	3.3E+2 1.1E+2	1.7E+3 5.4E+2	1.93E-1	7.5E-3 7.2E-3
ALPHA		1.5E+2 3.5E+1	1.15E-2 5.0E-4	4.7E-4
HEAVY		2.0E+0 2.3E-1	6.68E-5 3.6E-6	3.3E-6

Table 3: RADMON Anticipated Upset Rates

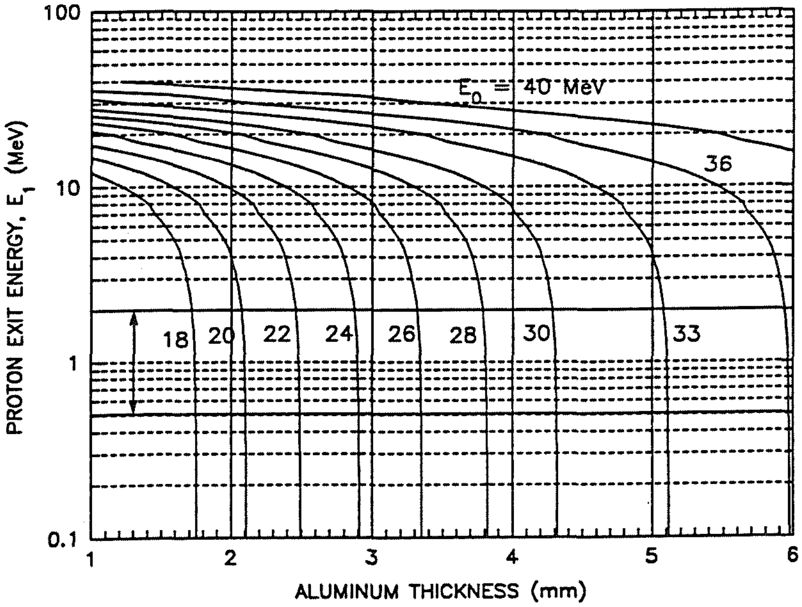


Figure 10: Proton exit energies for various aluminum thickness and incident proton energies. For $X_{A1} = 2mm, E_0 \approx 20MeV$ and for $X_{A1} = 5.5mm, \delta E_0 \approx 35MeV$

5 Discussion

The number of upsets estimated for the RADMON varies over eight orders of magnitude and depends on the type of the space environment encountered. The upset estimates depend on the accuracy of the environmental models and device parameters such as the collection depth. A simple algorithm was developed for estimating space particle flux that includes the effect of shielding. The flight of the RADMON on the STRV along with the CREDO and REM will provide data to quantify the sensitivity of the three particle detectors and validate ground predictions.

References

[1] M. G. Buehler, B. R. Blaes, G. A. Soli, and G. R. Tardio, "On-Chip p-MOSFET Dosimeter," submitted for publication IEEE Trans. on Nuclear Science (Dec 1993).

[2] J. F. Ziegler, J. P. Biersack, and U. Littmark, The Stopping and Range of Ions in Solids, Pergammon Press (New York, New York, 1985).

[3] J. H. Adams, "The Variability of Single Event Upsets Rates in the Natural Environment," IEEE Trans. on Nuclear Science, Vol. NS-30, 4473-3389 (1993).

[4] E. L. Petersen, J. B. Langworthy, and S. E. Diehl, "Suggested Single Event Upset Figure of Merit," IEEE Trans. on Nuc. Science, Vol. NS-30, 4533-4539 (1983).

6 Acknowledgments

The research described in this paper was performed by the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Ballistic Missile and Defense Organization, Innovative Science and Technology Office. The RADMONs were fabricated through MOSIS, ISI, USC. The authors are indebted to E. J. Daly, ESA, for the data shown in Figure 9 and for his observation that the RADMON detects high-energy protons that fall in a narrow band of energies. File: STRV3B04.DOC.

Test Results for SEU and SEL Immune Memory Circuits

D. Wiseman, J. Canaris, S. Whitaker, J. Gambles, K. Arave, and L. Arave
NASA Space Engineering Research Center for VLSI System Design
University of New Mexico, 2650 Yale Suite #101
Albuquerque, New Mexico 87106

Abstract – **Test results for three SEU logic/circuit hardened CMOS memory circuits verify upset and latch-up immunity for two configurations to be in excess of 120 MeV $\cdot$ cm^2/mg using a commercial, non-radiation hardened CMOS process. Test chips from three separate fabrication runs in two different process were evaluated.**

1 Introduction

It is well known that special radiation resistant integrated circuit (IC) processes exist that are designed to provide protection against single event latch-up (SEL) and single event upset (SEU) phenomena. This paper reports experiments which demonstrates that it is possible to provide a high degree of SEL and SEU immunity using a commercial, non-radiation hardened, CMOS process.

A test IC was designed consisting of five shift registers. Three of the shift registers utilized memory cells designed to be tolerant of SEU's through three different logic/circuit design hardening techniques [1, 2, 3]. The other two were traditional shift register designs to be used for comparisons. All of the structures were built using latch-up hardening techniques. The IC was hardened against single event induced latch-up through guard barring, layout techniques, and the use of a starting material with an epitaxial layer. Test samples were processed through MOSIS in Hewlett Packard's commercial 1μm double metal CMOS (CMOS34) process at the Corvallis, Oregon facility and in Hewlett Packard's commercial 0.8μm triple metal CMOS (CMOS26b) process at the Fort Collins, Colorado facility.

Tests to determine the SEU susceptibility of the specially designed memory cells were conducted at Brookhaven National Laboratories (BNL) on January 13, 1993, June 17, 1993 and August 2, 1993. In this paper the memory cell designs are reviewed, the measurement techniques are described and the results of the SEU testing are reported. Results imply that standard, high density, commercial CMOS processes can achieve SEU and SEL protection equal to or better than radiation hardened processes for space applications.

2 Design

The test chip consists of five sections each with a uniquely designed shift register. Three of the shift registers were constructed from flip flops designed to tolerate SEUs. Two were based on standard flip flop forms. Figure 1 shows the logic for a classical edge sensitive flip flop. The flip flop consists of a master section and a slave section each of which is a level sensitive data latch.

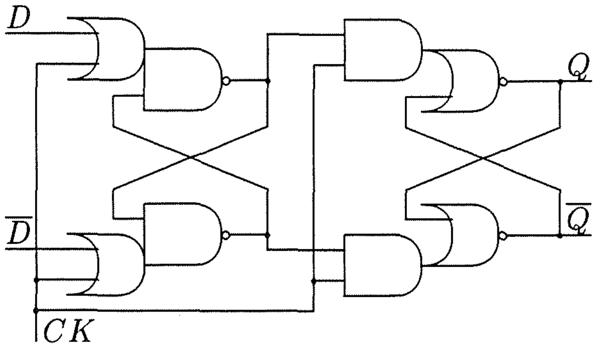


Figure 1: Classical master-slave flip flop.

The shift register of test chip section 5 is based on a direct implementation of Figure 1 using AndNor and OrNand cells from an SEU immune logic family [2]. The SEU immune logic family is based on three fundamental concepts. First, information must be stored in two different places. This provides a redundancy and maintains a source of uncorrupted data after an SEU. Second, feedback from the noncorrupted location of stored data must cause the lost data to recover after a particle strike. Finally, current induced by a particle hit flows from the n-type diffusion to the p-type diffusion. If a single type of transistor is used to create a memory cell then p-transistors storing a 1 cannot be upset and n-transistors storing a 0 cannot be upset. Figure 2 is a transistor level logic diagram of an SEU immune AndNor as described in [2].

The shift register of test chip section 4 is based on an unhardened complex complementary gate [4] implementation of Figure 1. This design was included to allow a direct comparison with the cell based on [2]. Section 3 implements a transmission gate flip flop using single n-transistors for the pass gates and weak feed back cross coupled inverters for the latch circuits. This circuit is also susceptible to SEU and is shown in Figure 3. This is the main style of flip flop used in most unhardened designs at the NASA Space Engineering Research Center and was included for comparison purposes.

The shift register of section 2 is based on the latch cell shown in Figure 4 and detailed in [3]. This is an SEU hardened design utilizing logic and circuit design based hardening techniques.

The shift register of section 1 is based on the latch cell shown in Figure 5 and detailed in [1]. This is also an SEU hardened design which is based on logic and circuit design based hardening techniques.

3 Test

Tests to determine the SEU susceptibility of the specially designed memory cells were conducted at Brookhaven National Laboratories. The objective was to determine the threshold linear energy transfers (LETs) and cross-sections for SEU and latch-up due to heavy ions. The Single Event Upset Test Facility (SEUTF) utilizes a Tandem Van De Graaff accelerator for providing various ions and energies. The ion beam is directed at a test board mounted

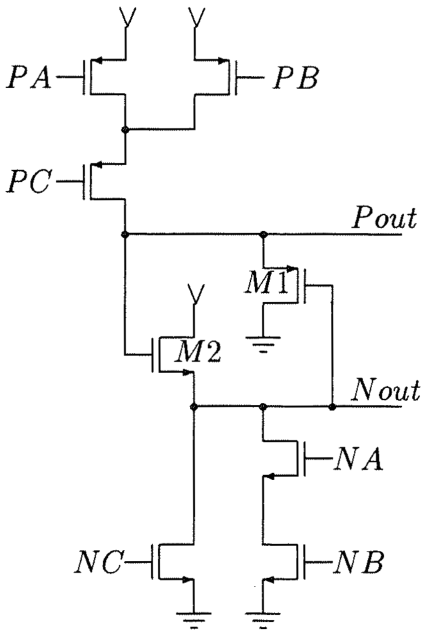


Figure 2: SEU immune AndNor.

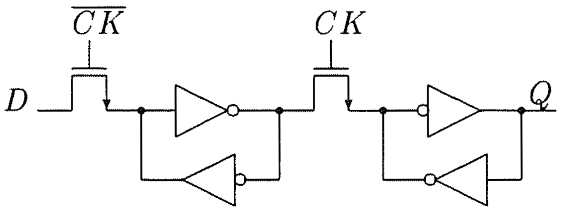


Figure 3: Transmission gate flip flop.

2.6.4

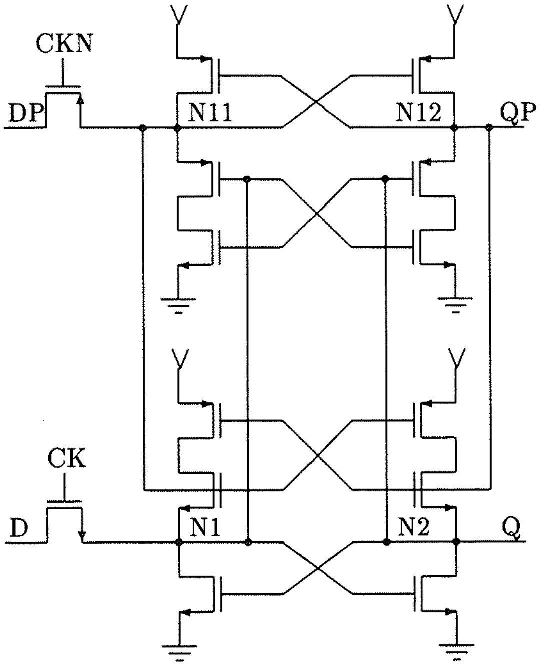


Figure 4: Low power SEU tolerant latch.

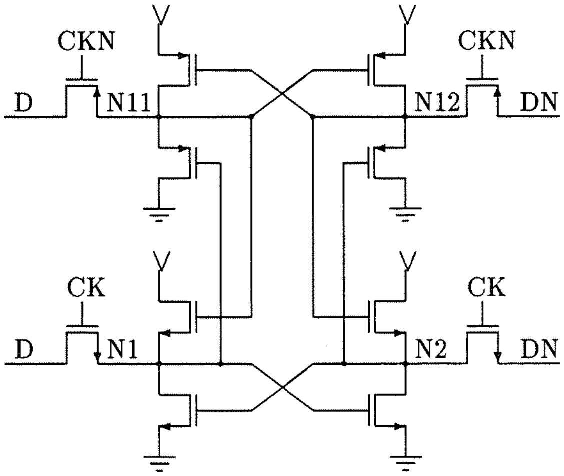


Figure 5: SEU hardened latch.

inside a vacuum chamber. The facility provides a computer-driven monitor and control program for the ion beam and test board setup.

Four of the test chips were placed on a wire-wrapped board, along with additional control logic, and mounted into the radiation test chamber at BNL. Three of the chips, with their die exposed, were placed in a heavy ion beam and individually irradiated. The outputs of the part being irradiated were compared in real time with the outputs of the fourth chip; the fourth chip acted as an experimental control unit. Counters were placed at the output of the board to record differences between the outputs of the chip being irradiated, and the outputs of the control chip. In this manner, upsets due to the heavy ion beam could be observed and quantified. The data sent through the shift register was comprised of an equal number of 1's and 0's. However, at any given time, different data patterns were sent through each register, to illuminate possible data dependent effects.

The current drawn by the chips was continuously monitored during the tests. Also, a current limit was placed on the voltage source such that if latch-up occurred in any of the parts, the power would be removed from the entire test board. The test board and chips were running at 1 MHz with a 5 volt supply. The board drew a total of 10 mA of current while operating under these conditions. The test board also included circuitry that would emulate upsets. By using this circuitry, the counters could be calibrated, and the functionality of the entire test set-up could be verified.

Experiments were conducted using Cl-35 at 206 MeV, Ni-58 at 265 MeV, I-270 at 320 MeV, and Au-197 at 350 MeV ions, beamed at various angles. The LET was steadily increased from 20 to 120 over the course of the experiment. The ion used for the beam was changed whenever the penetration depth of the beam for the desired LET would be less than 20 μm.

4 Results

For the tests conducted at BNL on January 13, 1993, the non-hardened designs exhibited upsets under every condition as expected. One of the hardened designs [3] exhibited an SEU threshold of LET=35 MeV $\cdot$ cm^2/mg while the threshold of the other two designs [1, 2] exceeded 120 MeV $\cdot$ cm^2/mg. No latch-up was observed in any of the 10 parts subjected to radiation, demonstrating an SEL threshold in excess of 120 MeV $\cdot$ cm^2/mg. Table 1 summarizes the experimental data for one of the test chips. Sec. 5 is the hardened design based on [2], Sec. 3 and 4 are the non-hardened designs. Sec. 2 was based on [3] while Sec. 1 was based on [1].

To verify that these results were not a fabrication anomaly, an additional processing run was made of the test chip and tested at high LET levels at BNL on June 17, 1993. The results were similar to the January experiment and are summarized in Table 2. In addition, to show independence of processing, the test chip was redesigned in a 0.8μm CMOS process and was tested at BNL on August 2, 1993 along with conducting more extensive tests on the 2nd run of the 1μm parts. Table 3 summarizes the testing results for the 2nd 1μm fabrication run while the results for the 0.8μm run are presented in Table 4. In all cases, the results were similar.

During the January, June and August tests at BNL, no SEL was observed in any part.

Six chips have been subjected to LETs of 120 MeV $\cdot$ cm^2/mg, and no SELs or irregular current anomalies have been observed, whatsoever. A previous run of the same test chip without guard barring and SEL layout hardening techniques showed an SEL threshold of approximately 31 MeV $\cdot$ cm^2/mg when tested at BNL on June 24, 1992.

During the course of the SEL/SEU testing at BNL, total dose data was also acquired. On January 13, 1993, chip A-0 received an accumulated total dose of 598 K Rad(Si), chip A-1 494 K Rad(Si) and chip A-2 532 K Rad(Si) without failure. During the testing on June 17, 1993, chip B-0 received an accumulated total dose of 195 K Rad(Si), chip B-1 182 K Rad(Si) and chip B-2 181 K Rad(Si) without failure.

The data implies that standard, high density, high performance, commercial CMOS processes can achieve SEU and SEL protection, for space applications, equal to or better than radiation hardened processes.

After the testing, Spice simulations were revisited on the SEU immune memory cell based on [3] which achieved an SEU LET of only 35 MeV $\cdot$ cm^2/mg. An additional ratioing constraint was discovered. This circuit design error has been corrected and the design has been used for flip flops on a Reed Solomon encoder which is to be fabricated in a $1\mu m$ triple metal process through American Microsystems Inc. at Pocatello, Idaho. Testing of this chip is scheduled for early next year. Also a CMOS26 version of the test chip, incorporating this correction, has been submitted to MOSIS for fabrication through HP in Corvallis.

References

[1] S. Whitaker, J. Canaris and K. Liu, "SEU Hardened Memory Cells for a CCSDS Reed Solomon Encoder," *IEEE Transactions on Nuclear Science*, vol. 38, No. 6, pp. 1471-1477, Dec. 1991.

[2] J. Canaris, "An SEU Immune Logic Family," *Proceedings of the 3rd NASA Symposium on VLSI Design*, Moscow, ID, pp. 2.3.1-2.3.12, Oct. 1991.

[3] N. Liu and S. Whitaker, "Low Power SEU Immune CMOS Memory Circuits," *IEEE Transactions on Nuclear Science*, vol. 39, No. 6, pp. 1679-1684, Dec. 1992.

[4] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design* 2nd Edition, Addison-Wesley Publishing Company, Reading, MA, 1992.

This research was supported in part by NASA under Space Engineering Research Center Grant NAGW-3293.

								Upsets	Upsets	Upsets	Upsets	Upsets
Run	Ion	Energy	Depth	LET	Time	Flux	Fluence	Sec. 1	Sec. 2	Sec. 3	Sec. 4	Sec. 5
		MeV	um	MeV sqcm	sec	#/cm2/sec	#/cm2					
				/ mg								
1	Cl-35	206	35.8	20.0	132.3	8.853E+04	1.171E+07	0	0	40	21	0
2	Cl-35	206	35.8	20.0	134.5	8.786E+04	1.182E+07	0	0	51	25	0
3	Cl-35	206	35.8	20.0	118.6	9.932E+04	1.178E+07	0	0	40	21	0
4	Cl-35	206	31.8	22.5	120.6	8.300E+04	1.001E+07	0	0	50	28	0
5	Cl-35	206	31.8	22.5	112.7	8.893E+04	1.002E+07	0	1	42	21	0
6	Cl-35	206	31.8	22.5	131.6	7.614E+04	1.002E+07	0	0	78	22	0
7	Cl-35	206	28.6	25.0	150.9	7.814E+04	1.179E+07	0	0	58	25	0
8	Cl-35	206	28.6	25.0	133.4	7.513E+04	1.002E+07	0	0	57	23	0
9	Cl-35	206	26.0	27.5	138.5	7.230E+04	1.001E+07	0	0	62	30	0
10	Cl-35	206	26.0	27.5	137.1	7.306E+04	1.002E+07	0	0	64	34	0
11	Cl-35	206	23.9	30.0	153.7	6.513E+04	1.001E+07	0	0	83	28	0
12	Cl-35	206	23.9	30.0	147.4	6.793E+04	1.001E+07	0	1	76	39	0
13	Cl-35	206	23.9	30.0	147.9	6.772E+04	1.002E+07	0	0	78	38	0
14	Cl-35	206	22.0	32.5	166.5	6.015E+04	1.001E+07	0	1	71	29	0
15	Cl-35	206	22.0	32.5	161.5	6.198E+04	1.001E+07	0	0	68	40	0
16	Cl-35	206	22.0	32.5	164.2	6.095E+04	1.001E+07	0	1	76	36	0
17	Cl-35	206	20.4	35.0	175.8	5.695E+04	1.001E+07	0	2	115	38	0
18	Cl-35	206	20.4	35.0	174.8	5.726E+04	1.001E+07	0	1	87	29	0
31	Ni-58	265	32.1	35.0	30.9	3.260E+05	1.007E+07	0	4	97	40	0
32	Ni-58	265	32.1	35.0	30.7	3.277E+05	1.006E+07	0	2	79	45	0
33	Ni-58	265	32.1	35.0	31.6	3.182E+05	1.006E+07	0	8	78	46	0
34	Ni-58	265	28.1	40.0	38.0	2.645E+05	1.005E+07	0	12	107	45	0
35	Ni-58	265	28.1	40.0	39.7	2.529E+05	1.004E+07	0	13	105	43	0
36	Ni-58	265	24.9	45.0	42.7	2.353E+05	1.005E+07	0	28	146	63	0
37	Ni-58	265	24.9	45.0	43.7	2.296E+05	1.003E+07	0	17	134	60	0
38	Ni-58	265	22.4	50.0	49.6	2.025E+05	1.004E+07	0	19	145	84	0
39	Ni-58	265	22.4	50.0	50.1	2.006E+05	1.005E+07	0	18	135	75	0
40	Ni-58	265	20.4	55.0	55.5	1.809E+05	1.004E+07	2	45	162	93	0
41	Ni-58	265	20.4	55.0	56.6	1.774E+05	1.004E+07	0	40	164	87	0
42	Ni-58	265	20.4	55.0	57.6	1.743E+05	1.004E+07	1	39	154	88	0
64	I-127	320	31.0	55.0	57.1	2.054E+05	1.173E+07	0	17	162	86	0
65	I-127	320	31.0	55.0	49.7	2.021E+05	1.004E+07	0	12	143	72	0
66	I-127	320	28.4	60.0	58.8	1.705E+05	1.003E+07	0	16	154	66	0
67	I-127	320	28.4	60.0	60.1	1.668E+05	1.002E+07	0	25	136	79	0
68	I-127	320	26.2	65.0	67.8	1.479E+05	1.003E+07	0	25	154	69	0
69	I-127	320	26.2	65.0	69.3	1.447E+05	1.003E+07	0	26	153	87	0
70	I-127	320	24.3	70.0	74.9	1.339E+05	1.003E+07	0	30	153	94	0
71	I-127	320	24.3	70.0	74.5	1.345E+05	1.002E+07	0	27	147	111	0
72	I-127	320	22.7	75.0	100.2	9.996E+04	1.002E+07	0	23	159	113	0
73	I-127	320	22.7	75.0	84.3	1.189E+05	1.002E+07	0	32	175	126	0
74	I-127	320	21.3	80.0	78.6	1.276E+05	1.003E+07	0	33	159	116	0
75	I-127	320	21.3	80.0	75.2	1.334E+05	1.003E+07	0	30	160	121	0
76	I-127	320	20.0	85.0	78.9	1.272E+05	1.004E+07	0	32	154	136	1
77	I-127	320	20.0	85.0	81.4	1.240E+05	1.009E+07	0	35	162	134	0
78	I-127	320	20.0	85.0	80.6	1.243E+05	1.002E+07	0	35	185	143	0
111	Au-197	350	27.5	85.0	19.0	6.891E+05	1.309E+07	0	31	186	121	0
112	Au-197	350	27.5	85.0	14.6	6.926E+05	1.011E+07	0	21	119	86	0
113	Au-197	350	26.0	90.0	16.1	6.296E+05	1.014E+07	0	23	131	96	0
114	Au-197	350	26.0	90.0	15.4	6.611E+05	1.018E+07	0	22	119	92	0
115	Au-197	350	24.6	95.0	15.8	6.424E+05	1.015E+07	0	22	122	110	0
116	Au-197	350	24.6	95.0	16.8	6.019E+05	1.011E+07	1	15	104	110	0
117	Au-197	350	24.6	95.0	17.6	5.768E+05	1.015E+07	0	18	128	92	0
118	Au-197	350	23.4	100.0	16.6	6.103E+05	1.013E+07	1	20	150	100	0
119	Au-197	350	23.4	100.0	16.6	6.095E+05	1.012E+07	1	25	145	98	0
120	Au-197	350	23.4	100.0	16.4	6.170E+05	1.012E+07	0	21	131	98	0
121	Au-197	350	21.3	110.0	18.6	5.450E+05	1.014E+07	0	18	135	122	0
122	Au-197	350	21.3	110.0	18.5	5.466E+05	1.011E+07	2	24	158	123	0
123	Au-197	350	21.3	110.0	18.5	5.476E+05	1.013E+07	0	33	171	128	0
124	Au-197	350	19.5	120.0	20.9	4.830E+05	1.009E+07	0	38	146	124	0
125	Au-197	350	19.5	120.0	20.3	4.984E+05	1.012E+07	0	31	155	138	0

Table 1 - Test summary part A-0 - BNL 1/13/93

2.6.8

Run	Ion	Energy MeV	Depth um	LET MeV sqcm / mg	Time sec	Flux #/cm2/sec	Fluence #/cm2	Upsets Sec. 1	Upsets Sec. 2	Upsets Sec. 3	Upsets Sec. 4	Upsets Sec. 5
1	Au-197	350	28.5	82.2	74.5	1.358E+05	1.012E+07		Bad	Run		
2	Au-197	350	28.5	82.2	74.3	1.350E+05	1.004E+07	1	16	145	70	0
3	Au-197	350	28.5	82.2	75.0	1.338E+05	1.003E+07	1	13	107	74	0
4	Au-197	350	28.5	82.2	76.6	1.309E+05	1.002E+07	0	15	148	83	0
5	Au-197	350	28.5	82.2	74.5	1.346E+05	1.003E+07	1	14	148	73	0
6	Au-197	350	28.5	82.2	76.6	1.309E+05	1.003E+07	1	8	138	88	0
7	Au-197	350	28.5	82.2	75.1	1.335E+05	1.002E+07	0	9	118	77	0
8	Au-197	350	26.0	90.0	82.9	1.209E+05	1.002E+07	0	8	136	86	0
9	Au-197	350	26.0	90.0	84.9	1.181E+05	1.002E+07	0	14	138	112	0
10	Au-197	350	26.0	90.0	99.6	1.100E+05	1.096E+07	2	10	145	101	0
11	Au-197	350	26.0	90.0	47.2	2.128E+05	1.004E+07	0	16	157	108	0
12	Au-197	350	26.0	90.0	48.6	2.065E+05	1.004E+07	0	24	131	103	0
13	Au-197	350	26.0	90.0	48.5	2.068E+05	1.003E+07	1	20	156	108	0
14	Au-197	350	23.4	100.0	53.9	1.863E+05	1.004E+07	0	21	145	121	0
15	Au-197	350	23.4	100.0	55.1	1.821E+05	1.003E+07	0	18	147	130	0
16	Au-197	350	23.4	100.0	57.0	1.760E+05	1.003E+07	0	20	139	118	0
17	Au-197	350	23.4	100.0	57.4	1.750E+05	1.004E+07	0	22	144	104	0
18	Au-197	350	23.4	100.0	58.5	1.712E+05	1.003E+07	0	26	134	106	0
19	Au-197	350	23.4	100.0	59.7	1.681E+05	1.004E+07	0	14	159	115	0
20	Au-197	350	21.3	110.0	69.0	1.453E+05	1.003E+07	0	24	155	121	0
21	Au-197	350	21.3	110.0	86.8	1.154E+05	1.002E+07	0	17	145	124	1
22	Au-197	350	21.3	110.0	87.0	1.152E+05	1.003E+07	0	16	160	122	0
23	Au-197	350	21.3	110.0	87.9	1.140E+05	1.003E+07	0	30	164	103	0
24	Au-197	350	21.3	110.0	88.3	1.138E+05	1.005E+07	0	11	155	148	0
25	Au-197	350	21.3	110.0	92.2	1.087E+05	1.002E+07	0	15	165	126	0
26	Au-197	350	19.5	120.0	105.9	9.456E+04	1.002E+07	0	22	150	130	0
27	Au-197	350	19.5	120.0	104.8	9.559E+04	1.002E+07	0	12	155	139	0
28	Au-197	350	19.5	120.0	101.6	9.856E+04	1.002E+07	0	17	151	151	2
29	Au-197	350	19.5	120.0	106.4	9.411E+04	1.002E+07	0	17	139	123	1
30	Au-197	350	19.5	120.0	107.8	9.293E+04	1.001E+07	0	18	165	128	0
31	Au-197	350	19.5	120.0	108.7	9.216E+04	1.002E+07	0	18	148	144	0
32	I-127	320	31.0	59.7	44.2	2.455E+05	1.086E+07	0	12	136	83	0
33	I-127	320	31.0	59.7	44.7	2.247E+05	1.004E+07	0	11	109	79	0
34	I-127	320	31.0	59.7	40.9	2.458E+05	1.006E+07	0	8	115	76	0
35	I-127	320	31.0	59.7	41.1	2.443E+05	1.005E+07	0	12	120	78	0
36	I-127	320	31.0	59.7	41.6	2.414E+05	1.004E+07	0	10	120	88	0
37	I-127	320	31.0	59.7	41.3	2.435E+05	1.006E+07	0	11	109	77	0

Table 2 - Test summary - BNL 6/17/93

Run	Ion	Energy MeV	Depth um	LET MeV sqcm / mg	Time sec	Flux #/cm2/sec	Fluence #/cm2	Upsets Sec. 1	Upsets Sec. 2	Upsets Sec. 3	Upsets Sec. 4	Upsets Sec. 5
1	F-19	142	120.4	3.41	202.4	4.940E+04	1.000E+07	0	0	1	0	0
3	F-19	142	120.4	3.41	226.6	4.418E+04	1.001E+07	0	0	0	0	0
5	F-19	142	120.4	3.41	53.9	1.862E+05	1.003E+07	0	0	0	0	0
7	F-19	142	120.4	3.41	54.8	1.836E+05	1.005E+07	0	0	0	0	0
9	F-19	142	83.9	4.89	78.2	1.282E+05	1.002E+07	0	0	1	0	0
11	F-19	142	83.9	4.89	78.4	1.279E+05	1.002E+07	0	0	0	0	0
13	F-19	142	83.9	4.89	82.1	1.221E+05	1.002E+07	0	0	1	0	0
15	F-19	142	56.5	7.26	119.5	8.379E+04	1.001E+07	0	0	1	0	0
17	F-19	142	56.5	7.26	141.6	7.074E+04	1.002E+07	0	0	2	0	0
19	F-19	142	56.5	7.26	167.4	5.984E+04	1.002E+07	0	0	2	0	0
21	F-19	142	43.0	9.56	145.1	6.902E+04	1.001E+07	0	0	3	0	0
23	F-19	142	42.8	9.60	156.1	6.416E+04	1.002E+07	0	0	4	0	0
25	F-19	142	42.8	9.60	140.9	7.104E+04	1.001E+07	0	0	2	0	0
27	Cl-35	208	62.7	11.50	42.2	2.351E+05	9.931E+06	0	0	15	0	0
29	Cl-35	208	59.2	12.18	45.1	2.227E+05	1.005E+07	0	0	23	2	0
31	Cl-35	208	49.5	14.55	55.5	1.809E+05	1.004E+07	0	0	50	4	0
33	Cl-35	208	42.7	16.89	64.5	1.556E+05	1.003E+07	0	0	48	12	0
35	Cl-35	208	37.6	19.20	76.0	1.320E+05	1.002E+07	0	0	36	17	0
37	Cl-35	208	30.4	23.72	97.3	1.030E+05	1.002E+07	0	0	58	25	0
39	Cl-35	208	25.6	28.16	114.1	8.780E+04	1.002E+07	0	2	79	18	0
41	Cl-35	208	62.7	11.50	44.1	2.276E+05	1.004E+07	0	0	27	0	0
43	Cl-35	208	59.2	12.18	47.3	2.124E+05	1.005E+07	0	0	22	2	0
45	Cl-35	208	59.2	12.18	47.5	2.115E+05	1.005E+07	0	0	32	3	0
47	Cl-35	208	49.5	14.55	57.1	1.754E+05	1.001E+07	0	0	31	5	0
49	Cl-35	208	42.7	16.89	66.3	1.513E+05	1.003E+07	0	0	40	12	0
51	Cl-35	208	37.6	19.20	74.9	1.339E+05	1.002E+07	0	0	57	19	0
53	Cl-35	208	30.4	23.72	91.2	1.099E+05	1.002E+07	0	1	71	18	0
55	Cl-35	208	25.6	28.16	111.6	8.978E+04	1.002E+07	0	3	85	17	0
57	Cl-35	208	62.7	11.50	44.9	2.240E+05	1.005E+07	0	0	20	1	0
59	Cl-35	208	59.2	12.18	48.7	2.062E+05	1.005E+07	0	0	30	4	0
61	Cl-35	208	49.5	14.55	61.3	1.635E+05	1.003E+07	0	0	35	8	0
63	Cl-35	208	42.7	16.89	74.5	1.346E+05	1.002E+07	0	0	48	9	0
65	Cl-35	208	37.6	19.20	81.9	1.223E+05	1.002E+07	0	0	44	14	0
67	Cl-35	208	30.4	23.72	101.7	9.851E+04	1.002E+07	0	1	71	16	0
69	Cl-35	208	25.6	28.16	118.5	8.455E+04	1.002E+07	0	6	73	22	0
71	Ni-58	265	38.5	29.16	50.3	1.996E+05	1.003E+07	0	5	66	32	0
73	Ni-58	265	33.2	33.85	57.3	1.753E+05	1.004E+07	0	13	76	33	0
75	Ni-58	265	29.2	38.50	63.0	1.594E+05	1.003E+07	0	25	114	48	0
77	Ni-58	265	26.0	43.11	69.5	1.443E+05	1.003E+07	0	37	120	50	0
79	Ni-58	265	24.0	46.73	74.8	1.341E+05	1.003E+07	0	24	146	45	0
81	Ni-58	265	38.5	29.16	46.3	2.171E+05	1.005E+07	0	6	79	43	0
83	Ni-58	265	33.2	33.85	53.9	1.862E+05	1.003E+07	0	8	92	48	0
85	Ni-58	265	29.2	38.50	59.4	1.688E+05	1.003E+07	0	24	97	38	0
87	Ni-58	265	26.0	43.11	68.1	1.473E+05	1.003E+07	1	23	118	45	0
89	Ni-58	265	24.0	46.73	74.6	1.346E+05	1.005E+07	0	33	130	68	0
91	Ni-58	265	38.5	29.16	53.2	1.886E+05	1.004E+07	0	5	78	36	0
93	Ni-58	265	33.2	33.85	59.9	1.675E+05	1.004E+07	0	8	80	38	0
95	Ni-58	265	29.2	38.50	67.8	1.478E+05	1.003E+07	0	18	108	47	0
97	Ni-58	265	26.0	43.11	72.1	1.392E+05	1.004E+07	0	26	139	60	0
99	Ni-58	265	24.0	46.73	82.1	1.221E+05	1.002E+07	0	39	140	51	0
101	I-127	320	30.5	60.74	16.9	5.995E+05	1.011E+07	0	28	115	43	0
103	I-127	320	28.1	65.93	19.5	5.171E+05	1.008E+07	0	25	143	75	0
105	I-127	320	26.0	71.10	22.5	4.498E+05	1.010E+07	0	42	160	75	0
107	I-127	320	24.3	76.28	24.0	4.209E+05	1.010E+07	1	40	146	82	0
109	I-127	320	30.5	60.74	19.9	5.065E+05	1.010E+07	0	25	161	61	0
111	I-127	320	28.1	65.93	21.1	4.774E+05	1.009E+07	1	29	120	61	0
113	I-127	320	26.0	71.10	23.2	4.348E+05	1.010E+07	0	31	158	83	0
115	I-127	320	24.3	76.28	24.5	4.112E+05	1.010E+07	0	43	157	74	0
117	I-127	320	30.5	60.74	20.0	5.025E+05	1.007E+07	0	29	126	49	0
119	I-127	320	28.1	65.93	21.5	4.689E+05	1.007E+07	0	45	135	73	0
121	I-127	320	26.0	71.10	23.8	4.232E+05	1.007E+07	0	31	116	66	0
123	I-127	320	24.3	76.28	26.0	3.884E+05	1.009E+07	1	39	121	84	0
125	Au-197	348.9	27.5	84.79	14.9	6.819E+05	1.015E+07	0	29	146	79	0
127	Au-197	348.9	25.8	90.55	16.2	6.260E+05	1.015E+07	2	39	147	69	0
129	Au-197	348.9	24.2	96.41	17.5	5.782E+05	1.013E+07	1	39	151	77	0
131	Au-197	349	25.9	90.00	28.1	3.582E+05	1.005E+07	0	18	119	85	0
133	Au-197	349	24.6	95.00	70.3	1.425E+05	1.002E+07	0	41	147	66	0
135	Au-197	349	23.3	100.00	48.4	2.083E+05	1.008E+07	0	30	124	93	0
137	Au-197	349	21.2	110.00	51.6	1.945E+05	1.003E+07	0	39	141	90	1
139	Au-197	349	19.5	120.00	62.8	1.598E+05	1.003E+07	1	31	152	105	5
141	Au-197	349	28.4	82.20	14.7	6.931E+05	1.017E+07	0	32	132	66	0
143	Au-197	349	25.9	90.00	15.6	6.537E+05	1.017E+07	0	30	131	71	0
145	Au-197	349	23.3	100.00	17.1	5.911E+05	1.010E+07	0	40	165	100	0
147	Au-197	349	21.2	110.00	19.0	5.341E+05	1.012E+07	0	38	155	91	0
149	Au-197	349	19.5	120.00	21.3	4.747E+05	1.009E+07	0	36	143	115	1
151	Au-197	349	28.4	82.20	14.7	6.920E+05	1.015E+07	0	34	150	80	0
153	Au-197	349	25.9	90.00	16.6	6.073E+05	1.011E+07	0	45	140	81	0
155	Au-197	349	23.3	100.00	18.8	5.368E+05	1.011E+07	0	31	141	81	0
157	Au-197	349	21.2	110.00	22.6	4.464E+05	1.007E+07	0	35	134	105	1
159	Au-197	349	19.5	120.00	25.1	4.017E+05	1.008E+07	0	36	156	109	5
161	Au-197	349	20.3	115.00	26.5	3.799E+05	1.008E+07	0	47	143	114	3
163	Au-197	349	20.3	115.00	31.4	3.214E+05	1.008E+07	1	31	143	98	2
165	Au-197	349	20.3	115.00	93.4	1.072E+05	1.002E+07	1	31	134	98	2
167	Au-197	349	20.3	115.00	19.4	5.198E+05	1.010E+07	0	26	126	103	1
169	Au-197	349	20.3	115.00	18.9	5.332E+05	1.010E+07	2	39	136	77	5
171	Au-197	349	20.3	115.00	91.6	5.470E+05	5.009E+07	3	163	674	475	14
173	Au-197	349	20.3	115.00	94.9	5.280E+05	5.009E+07	4	196	639	506	14
175	Au-197	349	20.3	115.00	106.0	4.726E+05	5.009E+07	2	165	643	517	20

Table 3 - Test summary 2nd fab run of CMOS 34 part - BNL 8/2/93

Run	Ion	Energy MeV	Depth um	LET MeV sqcm / mg	Time sec	Flux #/cm2/sec	Fluence #/cm2	Upsets Sec. 1	Upsets Sec. 2	Upsets Sec. 3	Upsets Sec. 4	Upsets Sec. 5
1	F-19	142	122.9	3.35	59.9	1.775E+05	1.064E+07	0	0	0	0	0
3	F-19	142	122.9	3.35	61.8	1.624E+05	1.004E+07	0	0	0	0	0
5	F-19	142	122.9	3.35	66.7	1.504E+05	1.003E+07	0	0	0	0	0
7	F-19	142	84.1	4.89	109.4	9.161E+04	1.002E+07	0	0	0	0	0
9	F-19	142	84.1	4.89	95.7	1.046E+05	1.002E+07	0	0	0	0	0
11	F-19	142	84.1	4.89	84.9	1.182E+05	1.003E+07	0	0	0	0	0
13	F-19	142	43.0	9.57	183.7	5.483E+04	1.007E+07	0	0	4	0	0
15	F-19	142	43.0	9.57	172.2	5.814E+04	1.001E+07	0	0	2	0	0
17	F-19	142	43.0	9.57	173.0	5.788E+04	1.001E+07	0	0	0	0	0
19	Cl-35	208	59.2	12.18	57.1	1.759E+05	1.004E+07	0	0	25	1	0
21	Cl-35	208	62.7	11.50	53.9	1.861E+05	1.003E+07	0	0	17	0	0
23	Cl-35	208	49.5	14.55	66.2	1.516E+05	1.003E+07	0	0	24	5	0
25	Cl-35	208	42.7	16.89	75.7	1.323E+05	1.002E+07	0	0	22	11	0
27	Cl-35	208	37.6	19.20	79.7	1.258E+05	1.002E+07	0	0	46	11	0
29	Cl-35	208	30.4	23.72	96.9	1.034E+05	1.002E+07	0	0	51	26	0
31	Cl-35	208	25.6	28.16	104.5	9.584E+04	1.002E+07	0	0	74	28	0
33	Cl-35	208	59.2	12.18	44.1	2.276E+05	1.004E+07	0	0	18	5	0
35	Cl-35	208	62.7	11.50	41.5	2.417E+05	1.004E+07	0	0	20	0	0
37	Cl-35	208	49.5	14.55	54.8	1.833E+05	1.004E+07	0	0	27	6	0
39	Cl-35	208	42.7	16.89	62.8	1.597E+05	1.002E+07	0	0	32	16	0
41	Cl-35	208	37.6	19.20	79.7	1.259E+05	1.003E+07	0	0	35	12	0
43	Cl-35	208	30.4	23.72	109.0	9.196E+04	1.002E+07	0	0	56	21	0
45	Cl-35	208	25.6	28.16	140.3	7.134E+04	1.001E+07	0	0	75	22	0
47	Cl-35	208	62.7	11.50	52.9	1.898E+05	1.004E+07	0	0	20	0	0
49	Cl-35	208	59.2	12.18	51.4	1.956E+05	1.004E+07	0	0	15	1	0
51	Cl-35	208	49.5	14.55	61.5	1.632E+05	1.003E+07	0	0	19	5	0
53	Cl-35	208	42.7	16.89	67.5	1.486E+05	1.003E+07	0	0	41	13	0
55	Cl-35	208	37.6	19.20	85.7	1.168E+05	1.002E+07	0	0	42	16	0
57	Cl-35	208	30.4	23.72	109.3	9.170E+04	1.002E+07	0	0	48	24	0
59	Cl-35	208	25.6	28.16	101.6	9.864E+04	1.002E+07	0	0	70	28	0
61	Ni-58	265	38.5	29.16	47.7	2.102E+05	1.003E+07	0	0	80	27	0
63	Ni-58	265	33.2	33.85	61.2	1.640E+05	1.004E+07	1	2	162	69	0
65	Ni-58	265	33.2	33.85	74.4	1.347E+05	1.003E+07	1	0	86	50	0
67	Ni-58	265	29.2	38.50	67.8	1.477E+05	1.002E+07	0	5	102	45	0
69	Ni-58	265	26.0	43.11	76.6	1.307E+05	1.001E+07	0	4	135	71	0
71	Ni-58	265	24.0	46.73	83.9	1.195E+05	1.002E+07	0	5	132	73	0
73	Ni-58	265	38.5	29.16	54.5	1.843E+05	1.004E+07	0	1	63	42	0
75	Ni-58	265	33.2	33.85	68.5	1.465E+05	1.003E+07	0	1	79	48	0
77	Ni-58	265	29.2	38.50	83.3	1.203E+05	1.003E+07	0	2	109	54	0
79	Ni-58	265	26.0	43.11	90.6	1.106E+05	1.003E+07	0	7	114	69	0
81	Ni-58	265	24.0	46.73	140.9	7.108E+04	1.001E+07	0	14	147	87	0
83	Ni-58	265	38.5	29.16	18.4	5.481E+05	1.009E+07	0	1	74	35	0
85	Ni-58	265	33.2	33.85	20.1	5.019E+05	1.011E+07	0	4	73	45	0
87	Ni-58	265	29.2	38.50	23.0	4.380E+05	1.008E+07	0	4	82	56	0
89	Ni-58	265	26.0	43.11	26.1	3.862E+05	1.008E+07	0	5	98	74	0
91	Ni-58	265	24.0	46.73	30.6	3.288E+05	1.006E+07	0	3	119	72	0
93	I-127	320	30.5	60.74	13.9	7.307E+05	1.016E+07	0	7	145	71	0
95	I-127	320	28.1	65.93	15.3	6.622E+05	1.015E+07	0	20	155	86	0
97	I-127	320	26.0	71.10	17.1	5.928E+05	1.012E+07	0	18	143	99	0
99	I-127	320	24.3	76.28	17.9	5.675E+05	1.013E+07	0	21	172	120	0
101	I-127	320	30.5	60.74	15.0	6.789E+05	1.018E+07	0	13	132	66	0
103	I-127	320	28.1	65.93	16.2	6.245E+05	1.012E+07	1	19	174	90	0
105	I-127	320	26.0	71.10	17.6	5.733E+05	1.011E+07	0	16	152	81	0
107	I-127	320	24.3	76.28	18.7	5.391E+05	1.010E+07	0	21	153	113	0
109	I-127	320	30.5	60.74	15.2	6.660E+05	1.014E+07	0	14	160	92	0
111	I-127	320	28.1	65.93	16.8	6.046E+05	1.013E+07	0	12	153	81	0
113	I-127	320	26.0	71.10	18.2	5.552E+05	1.009E+07	0	13	161	111	0
115	I-127	320	24.3	76.28	19.4	5.220E+05	1.012E+07	1	21	157	99	0

Table 4 - Test summary CMOS 26 part - BNL 8/2/93

Session 3
Architectures I

Chairman: Kelly Cameron

Space Qualified High Speed Reed Solomon Encoder[1]

Jody W. Gambles
NASA Space Engineering Research Center for VLSI System Design
University of New Mexico, 2650 Yale Suite #101
Albuquerque, New Mexico 87106
jgambles@groucho.mrc.unm.edu, 505-277-9707

Tom Winkert
Goddard Space Flight Center
Greenbelt, Maryland 20771
tom@skater.gsfc.nasa.gov, 301-286-2917

Abstract - **This paper reports a Class S CCSDS recommendation Reed Solomon encoder circuit baselined for several NASA programs. The chip is fabricated using United Technologies Microelectronics Center's UTE-R radiation-hardened gate array family, contains 64,000 p-n transistor pairs, and operates at a sustained output data rate of 200 MBits/s. The chip features a pin selectable message interleave depth of from 1 to 8 and supports output block lengths of 33 to 255 bytes. The UTE-R process is reported to produce parts that are radiation hardened to 1^6 Rads (Si) total dose and 1.0^{-10} errors/bit-day.**

1 General Description

This paper reports a JAN Level S radiation hardened gate-array implementation of a Reed Solomon (RS) encoder meeting the Consultative Committee for Space Data Systems (CCSDS) recommendations for code length and error detecting and correcting capability[1]. The gate-array meets the functional specification as the full custom CMOS VLSI encoder described by Whitaker and Liu[4]. A RS code is a powerful, relatively low overhead, cyclic symbol error correcting code which is particularly useful in correcting data suffering burst errors. In a spacecraft telemetry encoding system, a RS code may be used by itself or it may be concatenated with additional encoding for improved performance. While RS is used for burst error correction, CCSDS recommends convolutional encoding for correction of disbursed errors that result from Gaussian type noise. The RS code not only protects against burst errors resulting from the communications channel but also those resulting from the Viterbi decoding, used for decoding the CCSDS Convolutional code, where the decoded bit errors tend to clump together when its decoding capability is exceeded. The CCSDS recommendation calls for a (255,223) RS code where a data block contains 255 symbols, consisting of 223

[1] This research was supported by NASA under Space Engineering Research Grant NAGW-3293.

information symbols and 255 − 223 = 32 RS check symbols. Since two check symbols are required for each error to be corrected, 32 RS symbols can correct 16 symbols errors. Such a code may be referred to as a RS16 code. Each symbol of this code has a length of eight bits, or one byte. Due to the flexible nature of the algorithms being implemented, the circuit will support the encoding of shortened, as well as full length RS codes. Specifically, the codes which are supported are of the form: $(255 - i, 223 - i)$, where i can be any integer from 0 to 222.

The code is defined over the finite field $GF(2^8)$. The field defining primitive polynomial is $p(x) = x^8 + x^7 + x^2 + x^1 + x^0$ and the generator polynomial is given by:

$$G(x) = \prod_{i=112}^{143} (x - \beta^i)$$

where $\beta = \alpha^{11}$. The encoder represents data in the Dual Basis defined by the following transforms:

$$[z_0, z_1, \ldots, z_7] = [u_7, u_6, \ldots, u_0] \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$[u_7, u_6, \ldots, u_0] = [z_0, z_1, \ldots, z_7] \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

where $[z_0, z_1, \ldots, z_7]$ is the symbol represented by the dual basis and $[u_7, u_6, \ldots, u_0]$ is the symbol represented by the normal basis.

2 Features

A block diagram of the chip is shown in Figure 1. Before circuit operation can begin, the encoder must be reset and the chosen interleave depth, I, must be set by bringing the interleave depth control pins, S_0, S_1 and S_2 to the appropriate state. The encoder can be programmed to interleave the data at depths of one, two, ... or eight. Interleaving of two or more encoded messages allows higher burst error correction capabilities.

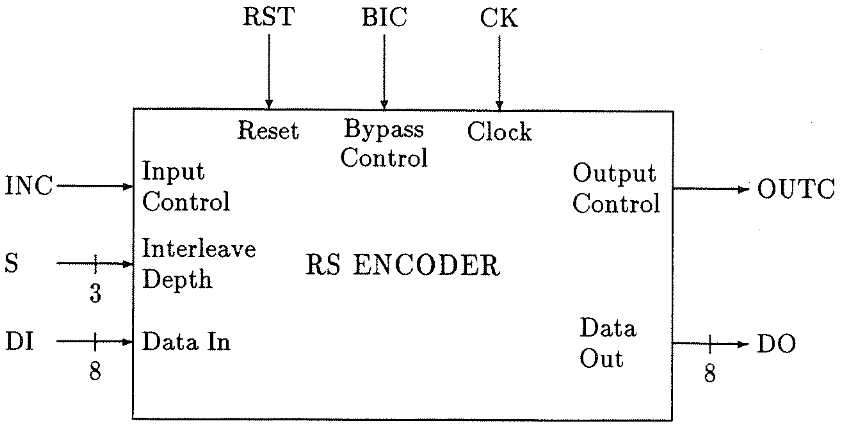


Figure 1: Block Diagram for the RS Encoder.

2.1 Encoder Operation

The encoder has eight data input ports, DI0-7, and eight data output ports, DO0-7. The information symbols are input at a constant rate in a byte serial fashion and the RS16 code is output byte serial after a fixed latency of one clock cycle. The input control (INC) pin must be brought high coincident with the first information symbol of the message to be encoded and remain high while all message symbols are clocked into the encoder. The INC pin is then brought low to signal the encoder that the last information symbol has been input. For a RS code encoding k information symbols, the first k codeword symbols are exactly the input symbols. While data is being clock into the encoder it is also passed directly to the output, with the fixed one clock cycle latency. The output control (OUTC) signal is brought high by the encoder to signal that the input data being encoded is being passed directly to the output. The OUTC output returns low after the last information symbol has been passed to the output. Beginning the next clock cycle after the last information symbol has been output, the 32 RS check symbols are concatenated to the output. The data and control signal timing is shown in Figure 2. The output data rate for the chip is 200 Mbits/sec when clocked at the maximum clock rate of 25 MHz.

2.2 Bypass Operation

A bypass mode included in the chip allows data flow through the encoder without being encoded. This is accomplished by bringing the bypass input control (BIC) pin high coincidentally with the first byte of data to be passed unprocessed by the chip. After the fixed one clock cycle latency, the data entering on the input appears on the output and continues to be passed through the chip as long as the BIC remains high. The INC pin should be held low during bypass operation to disable the generation of check check symbols within the encoder and keep the storage registers cleared.

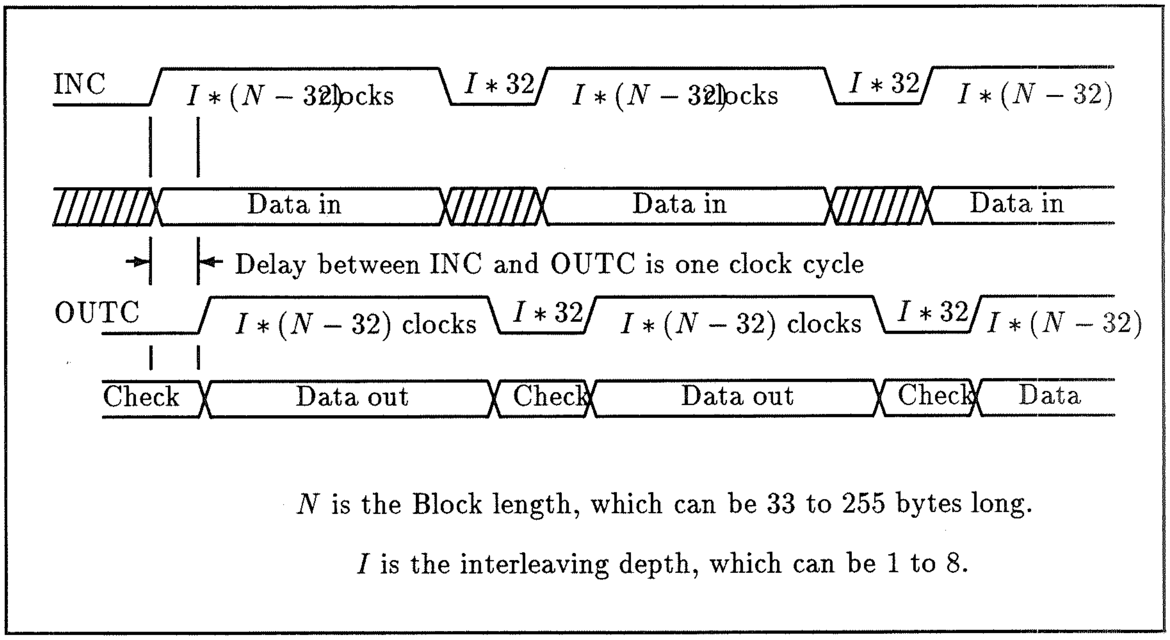


Figure 2: Input/Output Timing Diagram.

3 Chip Applications

Several NASA programs have baselined the use of this chip. They include the X-Ray Timing Explorer (XTE) and Tropical Rainfall Measuring Mission (TRMM). By encoding the telemetry data from these missions with the RS chip, a highly reliable data link will be established.

In addition to the RS coding, Convolutional encoding is performed after RS encoding to provide a higher coding gain than RS encoding alone. Figure 3 shows the theoretical performance of using various coding schemes[1].

The vertical axis shows the probability that there will be an uncorrectable bit error while the horizontal axis shows the information bit energy to noise ratio for a given communications link. For a desired probability of 10^{-5}, an information bit energy to noise ratio of 9.5 db is needed without any encoding and 2.3 db for a concatenated code (using both the RS and Convolutional codes). Therefore, using the concatenated code will give the desired bit error rate in the presence of considerable more noise than without encoding.

The selectable interleaving (I in Figure 2) provides mission flexibility to choose the interleaving while taking into account the tradeoffs for different interleave factors. No interleaving results in small data blocks while higher interleaving results in large data blocks. Large blocks are desirable because they provide better burst error correction. However, if a data dropout occurs, data will be lost on a block basis. Using large blocks will result in more data loss, while small blocks, because of their modularity, will result in less data lost. An interleave depth of 5 has been selected as a compromise for the XTE and TRMM programs.

The flexibility to vary the data block length (N in Figure 2) is important. Although

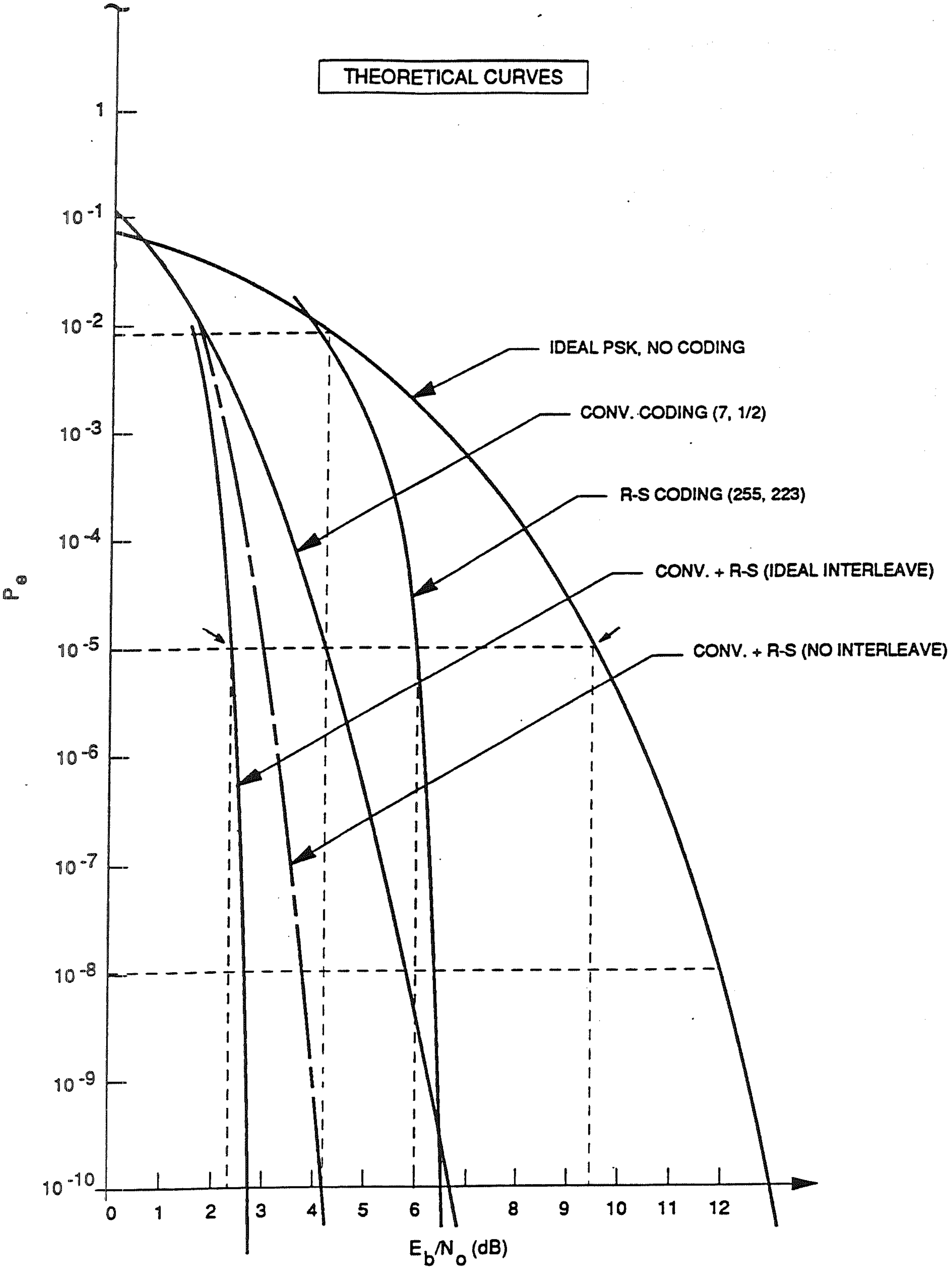


Figure 3: Relative Performance of Concatenated Coding with RS Outer Code (RS Ideal Interleaving and No-Interleaving Comparison)

the highest data throughput is achieved when the maximum size data block is used, a full data block may be impractical. The full code block is 255 bytes per interleave factor. When processed by a 32 bit (4 byte) word computer, the block will not fall on word boundaries. The XTE and TRMM programs will use the largest possible block that will fit on word boundaries. The resulting code is then a (252,220) code. For every 252 bytes, 220 are data and 32 are check symbols. Without the flexibility of the chosen RS algorithm to obtain the (252,220) code, zero fill would have to be used. In other words, the chip would have to receive 3 leading bytes of zeros to make 223 bytes of data instead of 220. At the output of the encoder the zeros should then be taken out so bandwidth is not wasted. Clearly, allowing for variable block sizes makes interfacing to the chip a lot cleaner for shortened code blocks.

CCSDS recommends that a 32 bit telemetry frame synchronization marker precede each data block. CCSDS also recommends that this marker not be included in the RS symbol calculations. The bypass mode allows the 32 bits to flow through the chip and not be included in the RS data block. This is also a cleaner approach than trying to insert the frame marker at the output of the RS chip.

Finally, the chip is available in a 84 pin QFP package which allows surface mounting of the device. This is the package used for XTE and TRMM.

4 The Implementation

4.1 Space Enhancement Features

The encoder has been fabricated using the United Technologies Microelectronics Center (UTMC) UTE-R Radiation-Hardened double metal gate-array process. The UTE-R process features submicron effective channel lengths (0.9μ leff, 1.2μ drawn) in a twin-tub, P-well epitaxail bulk CMOS technology employing special low-temperature processing techniques that enhance the total dose radiation hardness of the field and gate oxides[3]. The process is reported to be capable of producing parts tolerant of total dose radiation levels up to 1 Mrad (Si). UTMC gate-array products are listed on QPL-38510 and their double metal gate array process has been qualified for JAN Class S production.

In addition, the chip is designed to provide protection against Single Event Upsets (SEU) in two ways. First, the process and gate array cells are reported to be hardened against errors to less than $1.0X10^{-10}$ errors/bit-day. Second, the control structure and data path are configured to completely reset after each message insuring that an SEU of the data registers will effect at most one encoded message.

4.2 The Design

The encoder contains 32 multipliers and 33 adders which operate in parallel so that the mathematics required for check symbol generation can be performed at the data input clock rate. The encoder also contains 2048 bit registers (8 bits x 8 level interleave x 32 generated symbols). The input data, DI0-7, is framed by the INC input signal. As the data is input into the chip, it is presented to the check symbol generator and also passed directly to the

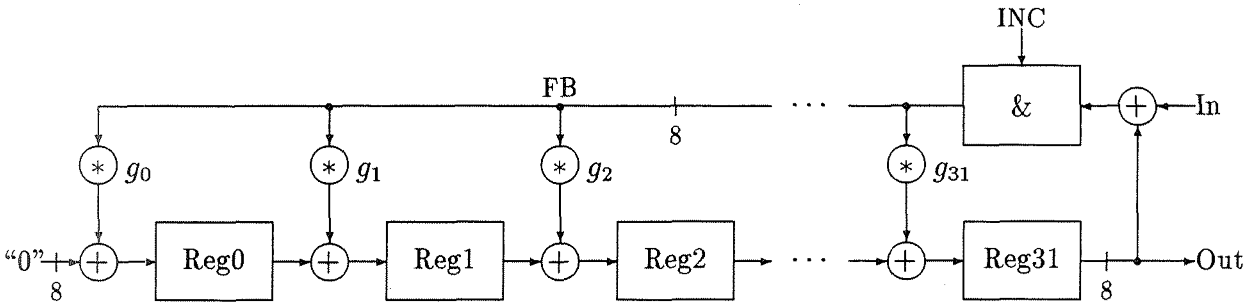


Figure 4: Check Symbol Generator Block Diagram.

output port DO0-7. At the end of the input data block, INC transitions low and the output of the check symbol generator passes out DO0-7. With INC low, 0's are input to the check symbol generator clearing out the registers. By the time that all of the check symbols have been clocked out of the encoder, all of the registers will be completely reset for the coding of the next block. Figure 4 shows a block diagram of the logic for the check symbol generator.

Each register is a 1 to 8 byte shift register, depending on the interleave depth set up during initialization. The 32 constant multiplier cells each multiply the 8-bit feedback value, FB, by a programmed constant, g_n. Each of these constant multipliers consists of an 8X8 XOR/ZERO cell array. A ZERO cell is simply defined as the absence of a XOR. The value of the constant, g_n, determines which array elements are XOR and which elements pass data unchanged (ZERO cell). The XOR/ZERO constant multipier array also performs the addition operation of the value from the previous register. An example of one of the 32 constant multiplier/addition cells is shown in Figure 5.

4.3 Design Evaluation Summary

The design was completed using Mentor Graphics IDEA version 7.1 and UTMC Mentor workstation toolkits. Test vector set fault grading was conducted using the Mentor QuickFault© tool. A total of 10100 nets were traversed during simulation with 98.58% toggled to both one and zero. All 143 untoggled nets are associated with unused JTAG circuitry included in the UTMC I/O pads. The projected fault detection is 99%.

The prototype parts where delivered on September 3, 1992. Heavy ion testing was conducted at Brookhaven National Laboratory's Single Event Upset Test Facility jointly by MRC and Goddard Space Flight Center personnel on December 3, 1992. The SEU Linear Energy Transfer threshold (LET_{th}) was found to be approximately 38 with the 10% threshold ($\text{LET}_{0.1}$) approximately 58. The test report concludes that the RS parts have Single Event Latchup (SEL) and SEU thresholds high enough to provide nearly zero probability for SEUs or SEL during extended mission lifetimes[2].

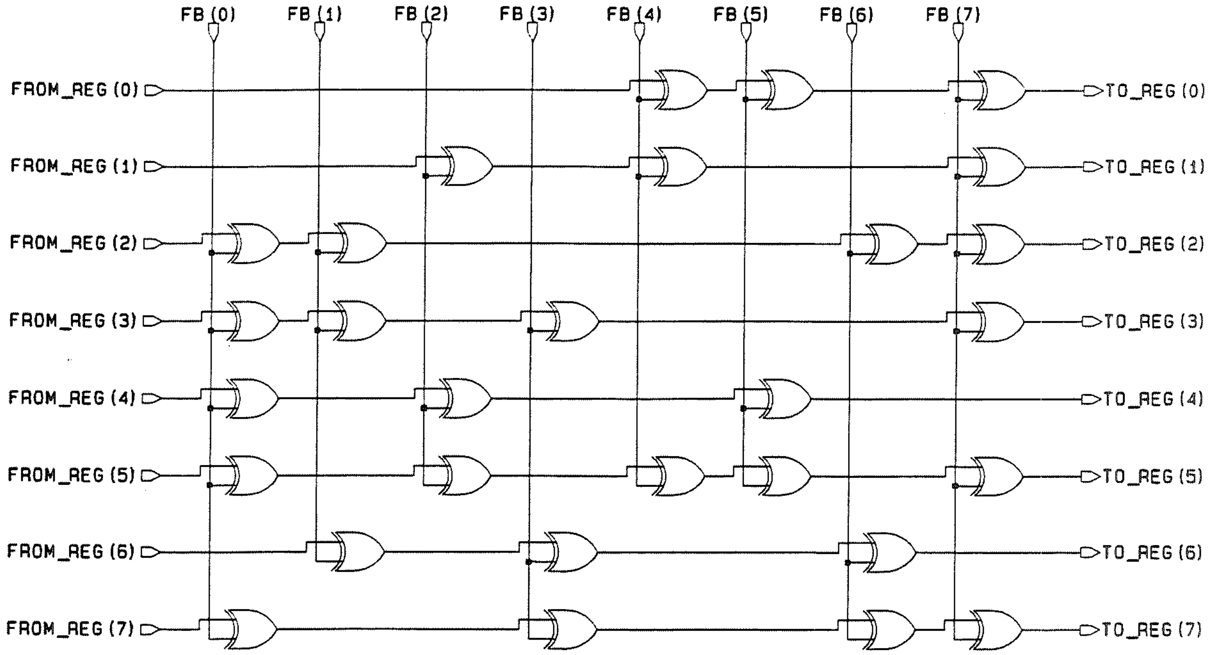


Figure 5: Constant Multiplier/Addition Cell.

5 Summary

A Reed-Solomon error correcting code encoder that meets the CCSDS recommendation for telemetry channel coding has been designed for the Goddard Space Flight Center. The chip was designed using the UTMC 1.2μ radiation-hardened gate-array technology and operates up to a 200Mbit/sec output data rate. The UTE-R process has been approved for the production of MIL-STD-883 Level S and JAN MIL-M-38510 Class S parts. Level B parts are also available, at reduced cost. In addition to meeting the CCSDS coding recommendation, the encoder includes shortened code and bypass mode operations that act to facilite clean interfacing of the chip to other system components. The chip has been baselined into flight hardware for the TRMM and XTE missions. The RS encoder chip is available from ICs as p/n RH-RS16-SE, (505) 277-9700.

Ackowledgement: This project was supported in part by NASA under grant NAGW-3293. The authors wish to acknowledge the support from Warner Miller at Goddard Space Flight Center.

6 Key to Acronyms Used

CCSDS Consultative Committee For Space Data Systems.

CMOS Complementary Metal Oxide Silicon.

JAN Joint Army Navy.

JTAG Joint Test Action Group – IEEE 1149.1 Standard for boundry scan.

LET Linear Energy Transfer (MeV·cm^2/mg).

MRC Microelectronics Research Center

QFP Quad Flat Pack(age).

RS Reed-Solomon.

SEL Single Event Latchup.

SEU Single Event Upset.

TRMM Tropical Rainfall Measuring Mission.

UTE-R UTMC's 1.2-micron gate array technology.

UTMC United Technologies Microelectronics Center.

VLSI Very Large Scale Integration.

XTE X-Ray Timing Explorer.

References

[1] Consultative Committee for Space Data Systems. "Advanced Orbiting Systems, Networks and Data Links" (CCSDS 700.0-G-3), November 1992.

[2] K. A. LaBel *et.al.* "Heavy Ion Results for Electronic Devices". In Workshop Record of 1993 IEEE Radiation Effects Data Workshop.

[3] United Technologies Microelectronics Center. Radiation-hardened CMOS. Technical Description, April 1990.

[4] S. Whitaker and K. Liu. "A High Speed CCSDS Encoder for Space Applications". In *2nd NASA Symposium on VLSI Design*, pages 1.3.1–1.3.9, Moscow, Idaho, Novemeber 1990. NASA Space Engineering Research Center, University of Idaho.

An ASIC Memory Buffer Controller for a High Speed Disk System

Robert F. Hodson and Steve Campbell
Christopher Newport University
Newport News VA 23606
rhodson@pcs.cnu.edu

Abstract -

The need for large capacity, high speed mass memory storage devices has become increasingly evident at NASA during the past decade. High performance mass storage systems are crucial to present and future NASA systems. Spaceborn data storage system requirements have grown in response to the increasing amounts of data generated and processed by orbiting scientific experiments. Predictions indicate increases in the volume of data by orders of magnitude during the next decade [1]. Current predictions are for storage capacities on the order of terabits (Tb), with data rates exceeding one gigabit per second (Gbps).

As part of the design effort for a state of the art mass storage system, NASA Langley has design a 144 CMOS ASIC to support high speed data transfers. This paper discusses the system architecture, ASIC design and some of the lessons learn in the development process.

1 SODR Design Overview

In response to these anticipated requirements, the Flight Electronics Division at NASA Langley Research Center (LaRC) has proposed development of an advanced mass storage system. This storage system is known as the SODR (Spaceflight Optical Disk Recorder) system. The target system performance aims for 2.4 Gbps data transfer rates, 1.2 Tb storage capacities, and access times of 250ms. Other requirements demand modularity and expandability of the system, as well as provisions for a variety of configurations.

The basis of the SODR system is a new optical disk from General Electric. These new disk drives, being developed separately from NASA, employ a magneto-optical process which uses a laser to "soften" the disk media so that a magnetic field can be used to write data. The magnetic polarity of the media indicates the logic of the data. The nature of the process allows both reading and writing the media and uses both sides of the 14 inch drive platter. This in effect places two independent drives within one device. This new technology from GE allows single side data transfers at 150 Mbps, 5 gigabyte (GB) storage capacity, and 150 ms access times. The dual nature of the drive allows 300 Mbps transfers, and a capacity of 10 GB per drive.

The high rates involved with the SODR drive system led to selection of an ANSI Standard High Performance Parallel Interface (HPPI) as the input interface. The HPPI format is one of few interface standards capable of the sustained rates incurred in SODR. The HPPI interface supports 32-bit communication at peak rates of 800 Mbps using twisted pair copper cabling between equipment up to 25m apart. Engineers at GE decided to use the ANSI Standard

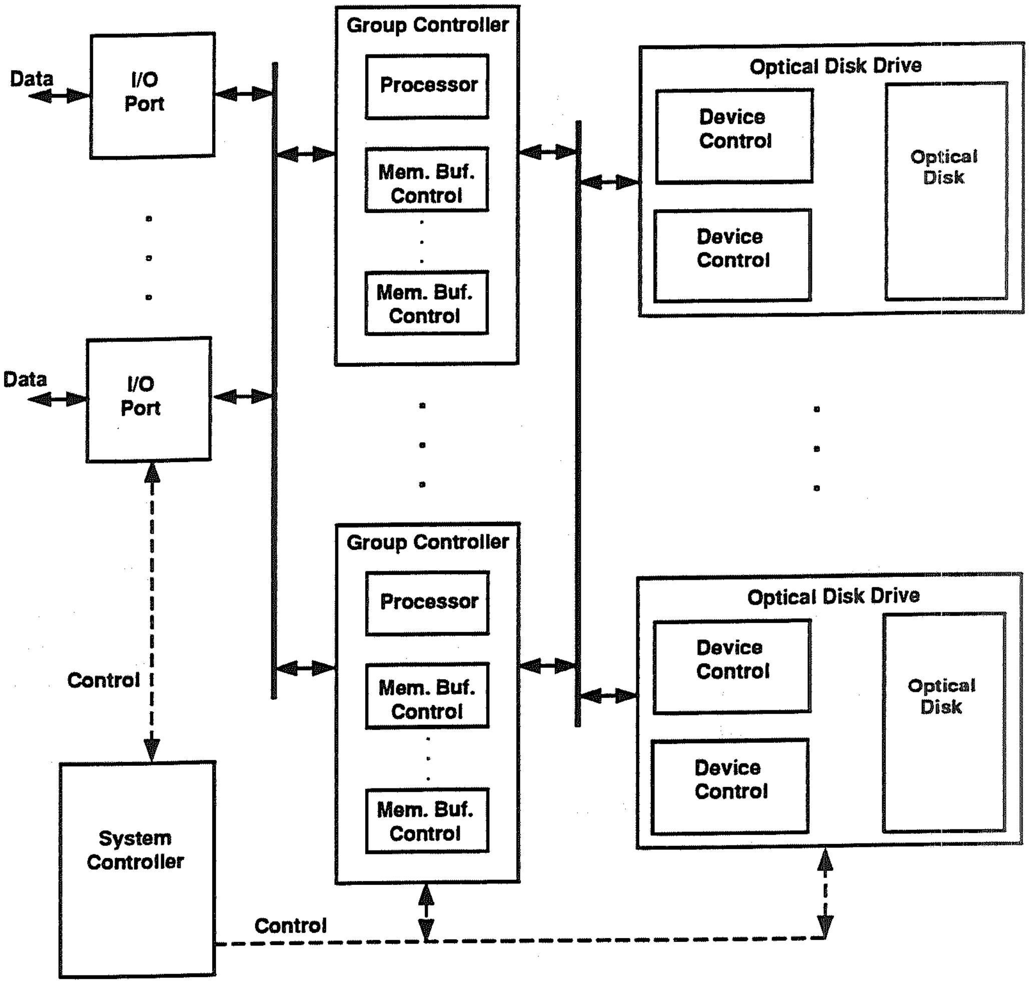


Figure 1: Fully Configured SODR Drive System

Small Computer Systems Interface II (SCSI II) as the format for their optical disk drive, and NASA developers decided upon a simple microprocessor interface for testing and development purposes.

Figure 1 shows a multi-drive SODR system. The SODR system targets NASA's requirements for modularity and expandability of a high performance system by accommodating multiple HPPI I/O ports, multiple SCSI II disk drive ports, and multiple disk controllers all controlled by a central command controller, communicating via a common system command bus.

The central controller sets up, directs, and controls overall system operation by allocating system resources. It does this by setting up a data transfer path by way of selecting a single 32-bit HPPI Input/Output port and two SCSI II drives (4 platters). Memory Buffer Controllers (MBCs) then handle details of transfers along 8-bit data paths. While the MBCs provide format conversion between HPPI and SCSI II formats, they also provide data buffering during all data transfers. The data buffering insures that data will not be lost, or stalled, during periods when the SCSI II drives encounter bad media on the disk during write operations or when data must be realigned for transfer during read operations. It was

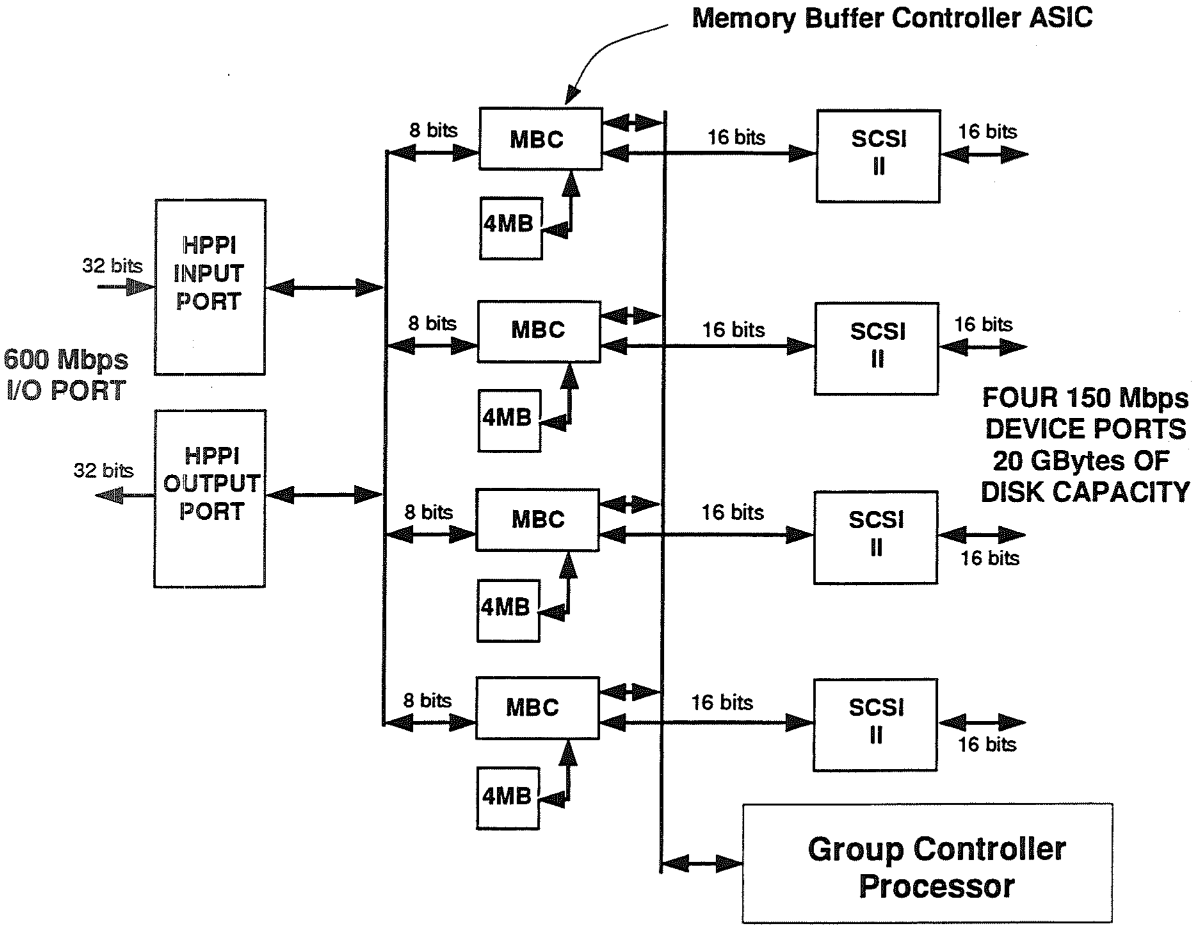

Memory Buffer Controller ASIC
32 bits
HPPI INPUT PORT
600 Mbps I/O PORT
32 bits
HPPI OUTPUT PORT
8 bits
MBC
4MB
16 bits
SCSI II
16 bits
FOUR 150 Mbps DEVICE PORTS
20 GBytes OF DISK CAPACITY
Group Controller Processor

Figure 2: Singe Channel SODR Port

determined that 4MB (1Mx32b) of memory would be capable of handling any rate matching requirements of the data transfers.

The large paths in the system combined with the high data rates required that the data for a single HPPI channel be striped across multiple disk drives. The data striping allows the memory buffering operations to be separated into 8-bit paths, thereby reducing the I/O requirements on any one MBC. By separating the 32-bit HPPI data across four SCSI II Disk Drives sides, the high bandwidth of the HPPI Channel is reduced by a factor of four. Figure 2 shows a single SODR I/O channel configuration with a single HPPI I/O port, multiple MBC ASICs and multiple SCSI II drives.

This view of a minimal SODR system environment in Figure 2 shows how four Memory Buffer Controllers support one 32-bit HPPI I/O port and two 16-bit SCSI II drives (4 sides). Each of the MBCs has a dedicated 4MB memory to buffer transfers along each 8-bit transfer path. One of the MBC ASICs, acting as a MASTER MBC, provides central control over the port. All connections and handshaking are mimicked by the other three devices, yet only one device is in control.

All command interaction with the system controller, shown as the Group Controller

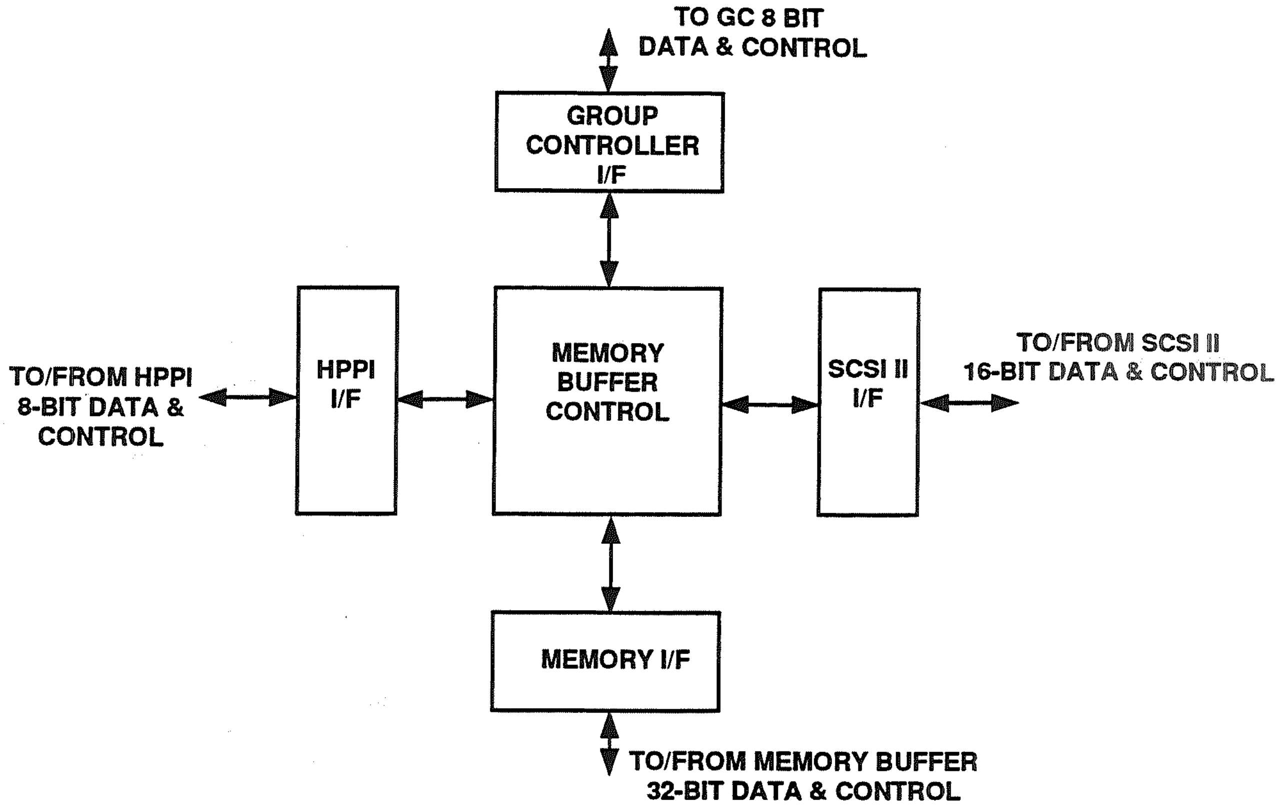


Figure 3: Functional Components of the MBC ASIC

Processor in Figure 2, takes place through the Group Controller Interface via the Group Controller Bus. As stated earlier, this interface is a simple 8-bit microprocessor interface. In addition to accepting command input and providing status information on data transfers, the GC interface provides a diagnostic port through which the operation of the ASIC can be tested. Many of the interfaces are operated and controlled by internal state machines. The independent and sometimes asynchronous nature of the inputs to these state machines, as well as the greedy timing specifications of all the chips involved, would prove to be breeding grounds for timing problems in the design.

2 Memory Buffer Controller Details

An 8-bit Memory Buffer Controller was the target of our ASIC design efforts. It was determined the the chip could be packaged in a 144 pin package and initial gate estimates based on an earlier 1-bit version of the design indicated the chip would be I/O bound rather than core bound. The ASIC is laid out into five functional sections shown below in Figure 3: Group Controller interface (GC I/F), HPPI interface (HPPI I/F), SCSI II interface (SCSI I/F), memory interface (MEM I/F), and the Memory Buffer Master Controller (MBMC).

The MBMC is designed to provide centralized control over the various interfaces within each ASIC. Each MBMC oversees internal command and control signals. This controller sets the mode of operation for the data transfers and coordinates the operation of all other modules in the systems. The MBMC has been implemented as a large synchronous state machine.

The HPPI interface is based on the timing requirements of AMCC HPPI source and destination chip set. The HPPI interface controls the hand shaking required by the HPPI protocol and combines the data into bursts for HPPI data transfers. The data path through the HPPI interface required a 512x9-bit data fifo. The unusual 9-bit wide data path was used so that parity information could be maintained within the interface. The control functions and signals were implemented and generated through small state machines. This interface proved to be the most problematic part of the design due the complexity of the HPPI chip set and the wide variability between minimum and maximum timing requirements of the AMCC devices.

The SCSI interface controls the data transfer to the SCSI II controller chip. An EMULEX SCSI II controller was chosen due to its separate DMA interface and the simple handshaking required to control this interface. The SCSI interface is a 16-bit bidirectional data interface. Like the HPPI I/F, the SCSI I/F incorporated a fifo within it's internal data path. The data fifo within the SCSI I/F was a 256x16-bit fifo. The SCSI I/F also utilized several small state machines to control and generate signals used within the SCSI I/F.

The memory interface provides a 32-bit interface to memory. The provision of a full 32-bit path to memory allows slower transfers to the 256Kx32b SRAM modules, while simplifying address, data, and control features. The limited selection of large, fast, and relatively inexpensive static RAMs influenced the design of this interface.

The Group Controller interface provides primary means for communicating with, and setting up the MBC ASIC. The GC I/F was originally designed as a simple asynchronous interface for use with a HARRIS RTX 2000 processor, but was later redesigned as a generic 8-bit microprocessor interface. The GC I/F is designed as a simple memory mapped I/O device, and provides the Group Controller Processor with a means of controlling and communicating with each of the four MBC ASICs it supervises.

3 Development Environment

The development of the MBC ASIC was done using the resources of the CAEDE (Computer Aided Engineering Design for Electronics) Lab at NASA/LaRC. The CAEDE lab is a SUN network of SUN SPARC2, and SUN IPX workstations. The lab maintains a variety of software packages, the most extensive of which is the CADENCE tool set. The CADENCE tools are used at NASA for schematic capture and hardware simulations. The current CAEDE environment supports both version 4.2a, and version 4.2.1 of CADENCE Composer, the primary digital schematic capture tool. Migration to the newest release of composer is scheduled to coincide with transition to the new SOLARIS OS.

In addition to CADENCE tools, the CAEDE lab acquired a US2 design kit from United Silicon. This design kit provided the tools necessary to hand off the simulated MBC design to US2 for layout and fabrication. The design kit uses US2 parts libraries which provide schematic symbols which are linked to VERILOG models of logical gates for use in schematic

3.2.6

capture within CADENCE. These libraries include standard generic gates, such as two input AND gates, ES2Cell II libraries, which model LS parts such as S171s, and compiled MegaCell libraries, which provide FIFO, RAM, and ROM models for use within CADENCE schematic capture tools. The design kit also provides tools for test vector generation and tools for specifying the pinout of the MBC ASIC.

4 Design Approach

Early in the origins of the SODR system, it had been determined that NASA staff, working in conjunction with faculty and graduate students from Christopher Newport University, would develop an ASIC capable of performing the functions of the Memory Buffer Controller using CADENCE schematic capture tools and US2 parts to develop. After NASA/LaRC had designed and simulated the proposed ASIC, US2 would provide layout and fabrication services. One of the co-objectives of the design of the MBC ASIC was to develop an overall design flow which could be used for the development of future ASICs at NASA/LaRC. By using US2 fabrication and layout services, the core of the design process could be investigated, and later the path through layout could be investigated.

During the early phases of the MBC development, functional models of state machines and other suitably complicated sections were implemented using behavioral models written in VERILOG. The early use of these models provided immense time savings since work could concentrate on the details of the interface designs and low-level logic details could be delayed. These behavioral models were written so that the functionality of preliminary designs could be verified without the hassles of tedious hardware design and verification.

The five functional sections of the MBC design in Figure ?? were divided into four major work groups, each of which was the responsibility of a person on the design team. By breaking the design into sections, the preliminary design of the MBC ASIC proceeded rapidly. The first section was composed of the MEM I/F and the MBMC. This section involved designing a memory interface capable of meeting timing requirements of the memory chips to be used, as well as development of the MBMC which would oversee the control of the ASIC. The MBMC was initially developed as one of the behavioral models described earlier because of the large number or inputs and outputs it encompassed. Implementing a behavioral MBMC allowed simple text edits instead of complex hardware changes whenever changes had to be made early in the design.

The second section included developing a HPPI interface and SCSI II interface which satisfied the HPPI and SCSI protocols, and using US2 tools to generate the fifos used in each of these interfaces. Both of these interfaces were extensively redesigned during the design process due to oversights and misunderstandings about the complex timing of the AMCC and EMULEX interface chips.

The third section entailed modeling the HPPI source and destination chips using VERILOG so that connection to external hardware could be simulated. These VERILOG models proved to be extremely crucial to the design development. Since the HPPI chips are basically large, complex state machines, decoding and verifying the VERILOG files used to behaviorally describe the chips seemed almost impossible. The designer of these VERILOG models was reassigned part-way through the design, and the other team members had not kept in communication during the design of these models. This proved to be a major setback late in the design of the MBC ASIC.

The final fourth section entailed the development of the GC I/F, replacing the behavioral models with hardware, and assembling the components designed by the other three sections of the design team. At this point, system, checkout, and interactive problems first began to appear. As the major components were connected together and simulation of the ASIC began, it became obvious that some sort of library management, as well as design management, should have been implemented from the outset.

5 Problems During The Design Process

During the course of the design of the MBC ASIC, the CAEDE lab upgraded from version 4.2a of CADENCE Composer to version 4.2.1, and back to version 4.2a. The CAEDE lab first upgraded from Composer v4.2a to Composer v4.2.1 as part of a normal software upgrade which was received from CADENCE. At the start of the design, US2 libraries (which were compatible with US2's own schematic capture tools) were not yet supporting Composer, so work proceeded using TTL parts from Composer libraries in Composer version 4.2.1. Once US2 parts libraries were delivered, significant time was lost as the designs were updated to US2 parts, and were then checked to verify functionality. At this stage in the upgrade we learned that the new US2 libraries had only been ported to Composer v4.2a. This discovery led to the CAEDE lab reinstalling the Composer v4.2a software. As design work and simulation progressed, it became evident that CADENCE's version of VERILOG and ES2's version of VERILOG were incompatible. The CADENCE VERILOG v4.2.1 would not compile the fifos generated with ES2 MegaCell tools. This meant using ES2 VERILOG for all design simulations, and as a result the simulation setups had to be changed to utilize the proper VERILOG executables and license files. All of the changes between library versions and Composer software changes required close work with the system administrators.

Early discussions with support staff at ES2 indicated that the schematic simulation waveform package, cWaves, which we were using with CADENCE's Composer was not supported by ES2 VERILOG. This meant using grWaves, (a rudimentary waveform package design to view simulation data generated by CADENCE VERILOG) which entailed a completely different simulation setup and much more work in order to produce simulation waveforms. After valuable time had been lost learning to make grWaves work in our simulation environment, we later learned from support staff at CADENCE that cWaves would indeed work with Composer v4.2a. After some more time was spent following CADENCE setup instructions, we soon had Composer 4.2a running with ES2 parts simulated with ES2 VERILOG, viewed using cWaves running under Composer from CADENCE.

Whenever a 'stable' design environment could be achieved, at least temporarily, the design team could begin to concentrate the timing details of the design, and work would proceed at a comfortable pace. We soon discovered that the flag signals generated by the US2 fifos often produced unwanted glitches and pulses. The nature of the fifo generation tools, and the lack of detailed fifo schematics available from US2 led the design team to replicate the flag circuitry external to the fifos. In addition to the flag problems, simulations indicated abnormally long delays through the pad instances which provided the external connection points on the ASIC. Further investigation revealed inaccurate delay calculations within US2 VERILOG were the cause and the design was again stalled as the design team awaited development and delivery of a new, corrected version of VERILOG from US2.

In addition to the fifos discussed earlier, US2 supplied some 'ES2 FAST' tools which provided utilities for checking, simulating, and packaging our design. These tools required a nominal amount of setup time, yet they would prove to require a large amount of time when they were invoked during simulation and verification of the design. Most of these utilities required three different simulations: minimum, typical, and maximum delays. As the design delivery date drew close, many times several computers were simultaneously running the ES2-FAST simulation and verification tools. None of the CADENCE, and ES2-FAST tools on hand provided a fault coverage tool. This meant we had no idea of how much of the design was toggled during simulation. Tools of this sort were available but had not been purchased to date.

Many of the tools discussed above dragged the Sparc stations to a crawl. Some of this was due to the nature of the tools. VERILOG simulations of complex digital designs is expected to be somewhat slow, but in our case a stronger machines would definitely have improved the productivity of the design team. (CADENCE recommends SPARC 10's with large amounts of both swap space and disk space.)

6 Design Management Issues

Several months into the design, it became evident that a project manager should have been delegated at the outset of the design. Opinions differed as to how certain interfaces should operate, and what signals should be available throughout the ASIC. With personnel from both Christopher Newport University and NASA on the project, the need for a person who would be responsible for the direction and delivery of the design became an issue. The design might have glanced over most of the smaller problems (and possibly some of the larger problems) with the benefit of an authoritative design leader, instead of roles and assumptions made by the design team.

One area of the design process which would have benefited is the library and file system. A lack of communication amongst the design team resulted in the development of several 'parts' which were created on an "AS NEEDED" basis by each the the designers. This meant that in some cases there were several functionally identical parts such as 8-bit 4-to-1 multiplexers. Better communication would have saved some time in the reproduction of such parts. In addition to duplicate parts, parts created by one team member could not always be accessed for editing as the design began to come together. The generation of a standard library where parts could be accessed, edited, and approved, would have saved a substantial amount of time.

Communication was also crucial to keep track of the designs progress. Each member of the design team had to know what ramifications changes in their own designs would possibly have on other parts of the design. As the individual components of the design team were wired together, and the upper levels of pads and logic were added, substantial time had to be invested whenever pins were changed, and whenever the internal logic changed. The time lost due to problems with the simulation waveform packages could have been minimized if the accuracy of any and all information was verified whenever a conflict emerged.

Several people were added to the design team as the delivery date for the ASIC drew near. The overall design process would have been more streamlined if new team members assigned by management were brought onto the project early enough to climb the learning curve required by all of the tools involved, as well as becoming familiar with the design. The time invested training personnel this late in the design did not seem to yield benefits for this design.

7 Design Environment Issues

At the onset of the design, each team member worked on their respective interfaces and designs within their own accounts. Several problems cropped up when people began to need access to other portions of the design. After experimenting with a UNIX work group, and different permission settings in both UNIX and CADENCE, the design was moved from separate accounts to one account. Once all of the design team began to use one account for design and simulation, many of these type problems went away. These migrations and test patches all involved substantial work, and subsequent design re-verification.

As indicated earlier, the nature of large digital simulations caused substantial system problems which resulted in bus errors, core dumps, and swap space errors. Rebooting the terminal in question sometimes helped, and working with the system administrators to increase, and improve the system performance helped. Budget limitations precluded upgrading the CAEDE lab workstations to hardware recommended by CADENCE.

The system administrators took a disliking early on to the incredible amount of data and files generated by this design, let alone the nearly constant interruptions due to system problems, upgrades, incompatibilities, and general user unfamiliarity with system details. We were constantly made aware of the space the design was taking up in the system. The complexity of the file structure proved to be cumbersome as we searched the files for clues whenever we ran into system, netlist, and simulation problems. As the delivery date for the ASIC drew near, we had become quite proficient at finding, and correcting, system problems as they cropped up.

8 Conclusions

One of the primary considerations before starting any ASIC design process such as this one is assuring that a stable, powerful machine networks is in place. The existence and maintenance of such a system requires knowledgeable system administrators who can work closely with the CADENCE tool users. The existence of such a network environment it not enough, the progress, and success of a large ASIC design is directly related to the maturity and stability of the tools which are used in that environment. The interaction and compatibility of the tools used must be investigated, maintained, and assured throughout the design process from schematic capture through layout to fabrication.

Building on a stable design environment, is the designation of a team leader who can layout the design goals, and directions encompassing the ASIC design. This includes establishing performance goals for the design, determining whether the design will be a top-down or bottom-up process, setting up the schedule for the design, determining how the ASIC will interact on a board-level, delegating responsibilities for sections of the design, and assuring that all members of the design team follow schedules and design ideology. This team leader can also server as a point-of-contact for system administrators, vendors, and management thereby minimizing any communication problems.

The design of a Memory Buffer Controller Application Specific Integrated Circuit demands close interaction between the vendors of the schematic capture package, parts vendors, layout services, network system administrators, and the entire design team in order to minimize problems and maximize productivity during the entire design process.

9 Acknowledgements

We would like to thank NASA's Flight Electronics Division for its support and give special thanks to Stephen P. Jurczyk and Thomas A. Shull.

References

[1] Glenn D. Hines, Stephen P. Jurczyk, Robert F. Hodson, *A Spacecraft Mass Storage Optical Disk System*, Proceedings of the Twelfth IEEE Symposium on Mass Storage Systems, April 1993.

Mongoose: Creation Of A Rad-Hard MIPS R3000

Dan Lincoln and Brian Smith
LSI Logic Corporation and NASA GSFC

Goal - **Describes the development of a 32 Bit, full MIPS R3000 code-compatible Rad-Hard CPU, code named Mongoose. Mongoose progressed from contract award, through the design cycle to operational silicon in 12 months to meet a space mission for NASA. The goal was the creation of a fully static device capable of operation to the maximum Mil-883 derated speed, worst-case Post-Rad exposure with full operational integrity. This included consideration of features for functional enhancements relating to mission compatibility and removal of commercial practices not supported by Rad-Hard Technology.**

"Mongoose" developed from an evolution of LSI Logic's MIPS-I Embedded Processor, LR33000, code named Cobra, to its Rad-Hard "equivalent", Mongoose. The term "equivalent" is used to infer that the Core of the processor is functionally identical, allowing the same use and optimizations of the MIPS-I Instruction Set software tool suite for compilation, software program trace, etc.

This activity was started in September of 1991 under a contract from NASA-Goddard Space Flight Center (GSFC)-Flight Data Systems. The approach affected a teaming of NASA-GSFC for Program Development, LSI Logic for system and ASIC design coupled with the Rad-Hard Process Technology and Harris (GASD) for Rad-Hard microprocessor design expertise. The program culminated with the generation of Rad-Hard Mongoose Prototypes one year later.

1 Background

The extremely short schedule required to provide a mission critical 32 Bit Microprocessor dictated severe constraints which forced the following objectives:

1. Conversion of an existing 32 Bit commercial design to a Rad-Hard design to mitigate technical development risk.

2. Complete compatibility with an existing Software Environment to enforce zero tolerance of schedule impact or dilution of design resources.

3. Use of a low risk Rad-Hard Process with fully integrated design tool support.

LSI Logic, as a MIPS partner with license to MIPS R3000 architecture, is a foundry for the MIPS R3000 Chip Set. To address the commercial Microprocessor Controller marketplace, LSI Logic had invested significant resources to convert MIPS R3000 dynamic design to a 100% code compatible fully static core. This produced an associated netlist for technology independence. The result, which now exists in the "Coreware Library" as LSI Logic's LR33000, met a key objective for schedule risk avoidance. Additional reasons for LSI Logic's LR33000 selection as the selection were:

3.3.2

1. Efficient RISC based architecture.
2. Existence of extensive software support tools, including C and Ada Compilers and PROM monitors.
3. Multiple generic vendors exist for the commercial device.
4. A development board (Pocket Rocket) exists for the generation and debug of operational code prior to the creation of the Rad-Hard solution. This allowed parallel hardware and software development.
5. The data base for the original structure had been licensed to LSI Logic Corp. allowing the "Core" structure to be transposed to a generic ASIC data base.
6. LSI Logic also supports a Rad-Hard Gate Array Technology (LRH20K) which is compatible with the design objectives of a space based configuration.

2 Design Goals

The intent of this development was to develop a 32-bit Rad-Hard Processor while:

1. Minimizing technical development risk.
2. Minimizing schedule risk.
3. Identifying all **Desired** and **Required** configuration changes. The distinction between the two is:
    (a) Desired Changes
        i. Additional features.
        ii. Functional Enhancements.
    (b) Required Changes:
        i. Identification and removal of features that are not compatible with Rad-Hard design.
        ii. Modification of the design methodology to make the design inherently solid in a Rad-Hard environment.

The ultimate goal was the creation of a fully static design capable of operation to the maximum derated speed with full functional integrity. This included consideration of features for functional enhancement relating to mission compatibility and removal of commercial practices not supported by the Rad-Hard Technology.

3 Enhancement Features

The intent of Mongoose was to serve as a total Space-Craft Controller. A number of internal functional features were created to service all needs with minimal external hardware:

1. Upload of Program Code via the inclusion of an RS-232 compatible UART. This UART could then be used as an interface port communicating via the RF Link. Incoming data could also be used to load an external EEPROM. The most common use of this interface is to support Ground Test.
2. Expand the Interrupt Structure to allow sensing of additional external interrupts.
3. Include a number of internal support functions to include:
   (a) 32 Bit Counter/Timers: Bus/Memory Timeout Timers
   (b) Watchdog Timer
   (c) DMA Channel Support
   (d) 2-Phase Internal Clocking System
   (e) Boot from EEPROM
   (f) Creation of an Internal Memory Interface
   (g) Debug/Control Block
   (h) Command/Status Interface

4 Non-Supported Features

1. Internal Cache: Clearly, the inclusion of I and D Cache has performance advantages. Its implementation requires a very dense internal RAM capability not supported by a Rad-Hard Gate Array Technology. This results in the necessity of placing I and D (Cache) Memory off-chip.
2. Memory Management Unit (MMU): This unit was not required within the scope of the application.
3. Floating Point Unit (FPU) Co-Processor: This function was desired but did not fit the desired development schedule.
4. DRAM Support: Not applicable since only static RAM's were utilized.

No additional features were considered if it was deemed they added a potential for schedule risk without adding value to the assigned mission.

5 Structural Impact

These changes resulted in major transitions to multiple areas of the structure:

- Interface:
    1. Compatibility with an existing pre qualified 256 pin Rad-Hard Package.
    2. Creation of several I/O Busses:
        - Three major Busses were placed on the peripheral of this design:
            (a) I Cache
            (b) D Cache
            (c) Local Bus
                - The Local Bus was defined as the Port for communication to:
                    i. Boot EEPROM
                    ii. External I/O Devices
                    iii. External DMA
                    iv. External Memories
                        - In each case, the interface included separate Address and Data Busses to maintain optimal bandwidth and complete "handshake" for communications control.

6 Design Definition

The original "Cobra Core" existed as a Standard Cell data base. It was subsequently converted to a Gate Array data base using the Rad-Hard LRH20K Technology to create a solid technical structure.

Subsequently a detailed architectural overview was held with all team members to examine the detailed trades. This included considerations within the following functions:

1. CPU
2. Cache Interface
3. External Memory Timing Requirements
4. Considerations for optimal clock rates
5. Hand-shake vs. pin-out tradeoffs.
6. Parity vs. EDAC vs. pin-out tradeoffs.
7. Size evaluation in gate count and chip area.
8. Pin count evaluation for control, chip access and "output drive" power support.

These considerations resulted in a documented design specification agreed to by all parties. The combined size and performance goals were:

1. Maintain the die size to a maximum of 1.5 cm./side. This supports approximately 50K used gates.

2. Maintain compatibility with the specified 256 Pin CLDC space-qualified Rad-Hard package. This avoids the cost of qualifying an additional package. This goal required controlling both maximum die size and pinout.

3. Operational speed equivalent to 10MHZ clock rate or approximately 8 MIP's. This performance assumes a single-clock memory response when operating from external RAM (with a minimum of 55ns access time).

Internally, a number of design methodology issues were addressed for performance integrity. These included:

1. Clock Distribution & Control
   The core uses two primary clocks, one of which is continuous and one which is disabled during "Stall Operations". Each clock had to be balanced within its own domain as well as between the sections to avoid race conditions.

2. Avoidance of any potential for Bidirect conflicts on internal or external busses.

Some internal latches exist where clocking is also applied as two separate non- overlapping pulses for early access to results of combinational logic. To achieve maximum performance in frequency, the low and high periods of the clock must be controlled. As a result, the incoming frequency was defined to be a 2x rate. This allowed internal generation of a 1x clock with 50then a function of internal layout and buffer symmetry only.

7 Data Base Generation

Design entry and data base generation used a complete repertoire of LSI Logic CAD tools within the internal Modular Design Environment (MDE) suite:

1. LES (Logic Expression Synthesis):

   - Used to generate state machines for:

     (a) Memory Control
     (b) Microboot Control
     (c) DMA Control
     (d) Data Bus Arbitration
     (e) Cache Read/Write Control
     (f) Byte Gathering

2. LOP (Logic Optimization):

   - Optimized netlists generated with LES.

3. LBS (Logic Block Synthesis):
   - Used to generate:
     (a) Counters
     (b) Counter/Timers
     (c) Fast-parallel 32 Bit Adders

4. Liberate
   - Generates schematics from net-lists.

5. LSED (Schematic Editor):
   - Schematic entry or editor for CPU Core and top-level interconnects.

6. VHDL
   - Used to define Cache Interface Control Block followed by the assertion of Synopsis synthesis.

7. Memcomp
   - Used to generate the 3 Port Memory-Based Register File.

8 Design Verification

Test for the original MIPS Core is composed of multiple sequences exercising all aspects of the processor as well as variations in the sequence of operations. Individual tests are then concatenated to form the composite. The Test Environment is also constructed to parse sequences among available processors on the local area network to minimize total simulation time.

Original test sequences (>400,000 test vectors) were first simulated and verified using the Cobra design as a stable reference. Following this, the database was converted to the target Rad-Hard Gate Array Technology without speed optimization or structural modifications. The test suite was again simulated to allow detection and elimination of bugs prior to structural enhancements. The original Cobra structure was then modified (¿200,000 test vectors) to support the external Mongoose interface including:

1. External Cache
   - This introduced two additional interfaces and the need for Cache RAM Behavioral models.

2. Functions eliminated:
   - Parity

- DRAM support
- Bus Snooping

Verification of the Mongoose Processor was performed using LSI Logic's Mixed-Level Simulator, MBSIM. The gate level was combined with Behavioral Models of Cache RAM, EEPROM and an I/O Processor to generate external events such as Interrupts and Bus Requests, allowing simulation of the Mongoose Processor in a "typical system environment". **This is a very major point-the best way to determine complete, well integrated operation is to allow external, asynchronous events to operate concurrent with normal instruction execution to validate the lack of conflicting conditions.**

A Behavioral RAM Loader, originally developed by the LSI Logic Cobra Design Team was used to load MIPS executable code directly into the Behavioral RAMs at the start of each simulation. Self- checking code was written in C or s assembly, compiled and linked on a MIPS platform. It was then loaded directly into the behavioral RAMs. In this way the Processor could be verified using actual MIPS binary code with test sequences generated using a higher level language.

Additional tests were generated to verify "Mongoose specific" features:

1. DMA
2. UART data transfers
3. Counter/Timers
4. Chip Selects
5. Bus Requests/Bus Grants
6. Bus Errors
7. I & D Cache Read/Write Cycles
8. Wait States

Wherever possible, the test code was self-checking. Test vectors were generated directly from the "System Test Environment" by sampling the Mongoose I/O during simulation and writing the results to an output file. This file was then reformatted to serve as an input source for the "Single Chip" simulation. Since the chip utilizes an internal 2 phase clocking system, test results were sampled and verified prior to the end of each half of the functional clock period.

The entire test suite was then compiled and verified, allowing the effort to proceed to layout and speed optimization. Use of the tools allowed timing to be optimized for speed performance at all corner points of the technology, both before and after radiation exposure. Speed characterization assumed worst case external RAM access times of 55ns. This also became the limiting item establishing the maximum system clock rate.

Before addressing the overall results as well as the results of the Radiation Testing, it is appropriate to address particulars of the Radiation Hardened Process Technology. Mongoose was designed referencing the LSI Logic LRH20K Radiation Hardened .8 micron HCMOS Epitaxial Silicon Gate Array Technology. This family has die sizes available to allow usable gate complexities from 5,000 gates to greater than 70,000 gates. Mongoose required a utilized gate count of slightly over 50,000 used gates including supported peripheral functions.

The LRH20K Technology has been characterized using an evaluation chip with 40,000 utilized gates. This technology is rated at 3 MRads (Si) for total dose, greater than 3E9 Rad (Si)/Sec. for Dose Rate loss of data for a 20ns pulse with no latchup detected for dose rates up to 2E10 Rad (Si)/Sec. , the source limit. The saturated cross-section for Single Event Effects is approximately 1E-6 Cm2/Bit, and the LET Threshold at 10rate based on a modified CREME code was calculated to be <8E-9 upsets/Bit-Day. No evidence of latchup was observed in any parts tested to a LET of 200 MeV/Mg/Cm2.

9 Results

The final design met all goals in gate count, die area and worst-case performance. A functional commercial board containing the LR33000 and "software hooks" for a test environment allowing external control, sense and break points was used by NASA-GFSC to develop the original functional code during the Mongoose development. When prototypes became available, they replaced the development board and were operational and on-line within two days. The chip was operated with an external clock source "scaled upward" to establish timing margins. The prototype, originally designed for worst-case 10MHZ operation, was operated successfully to 25MHZ.

Since then, the device has undergone much more extensive testing of the functional code within the true system environment. Two timing items were found relating to external events which resulted in minor logical changes. The original Mongoose has been modified and regenerated to include these enhancements. The change was made without affecting critical layout areas within the original device. The power consumption of the device has not yet been measured but has been calculated at 1.38 watts operating at 10MHZ.

It should be noted that the original performance goals were established by the selection of a non-exotic Rad-Hard RAM with 55ns access time. Selection of a more aggressive memory and optimization of the Core allow further performance improvements.

NASA GSFC performed Heavy Ion Testing at the SEU Test Facility, in DOE's Brookhaven Laboratory on 3 March 1993. The result confirmed previous measurements regarding the radiation tolerance level of the LRH20K Process and the Mongoose Microprocessor in particular. No latch-up was seen using ions up to Iodine, with an LET of 85. Analysis of the data has been completed, with results indicating "Worst Earth Orbit" (Geo-Synchronous), at the worst time in a Solar Cycle for SEU's would see an upset every 100-120 years. This is well within the mission requirements of NASA applications.

One would have to conclude that all work done in the area of significant changes to the microprocessor core is heavily dependent on the quality of the tools, the test suite, test environment and the familiarity of the development team. During this development, a very successful integration of all elements allowed a highly complex "Coreware" design to take place through a tightly-coupled team development approach, meeting a critical NASA flight schedule.

The Mongoose Microprocessor will fly on NASA GSFC's X-Ray Timing Explorer Spacecraft in 1995 and the Joint NASA/NASDA Tropical Rainfall Measurement Mission in 1996. Other applications are in development outside NASA. The Mongoose Microprocessor has been made available by NASA to the "Space Community" through LSI Logic Corporation.

10 Acknowledgements

The authors wish to thank the members of the Mongoose team for their diligence in completion of this project.

References

[1] G. Kane , *mips RISC ARCHITECTURE*, Englewood Cliffs, NJ: Prentice Hall, 1989.

[2] LSI Logic Corporation, LR33000 Self-Embedding Processor User's Manual. Milpitas, CA: 1991.

[3] B. S. Smith, Mongoose ASIC Programmer's Guide. National Aeronautics and Space Administration Technical Memorandum, Greenbelt, MD: Goddard Space Flight Center, 1993.

Digital Receiver for On-Board FM/FSK-FM/BPSK Demodulation

G.Boscagli M.C.Comparini and M.Martone
Alenia Spazio
Via Saccomuro 24, 00131
Rome ITALY
Tel: +396-4151-3613; Fax: +396-4151-3677

Abstract - **An all-digital demodulator FM/FSK-FM/BPSK is presented. The proposed architecture allows the 1-bit IF down- sampling technique with the hardware simplification of receiver analog section. Besides the receiver exhibits great flexibility both in terms of signal demodulation (FSK or BPSK, subcarrier/tone frequency, data rate), and interface capability. The demodulator is going to be realised in a VLSI chip (20 Kgate complexity). This paper describes the demodulator structure and its demodulation performances obtained with a breadboard based on Programmable Logic Devices. The choice of a simple and effective scheme for frequency detection (FM and FSK demodulators) is suggested by system design considerations as well. Signal Processing Algorithms include also a Costas carrier phase recovery scheme in the case of BPSK signal and Data Transition Tracking Loop for the bit clock recovery. Test results address this solution as one of the most suitable for TT&C space application.**

1 Introduction

The speed of digital hardware is continuously increasing together with the limit of the maximum sampling rate, as consequence it is possible to move the boundary between analog and digital domain at higher frequency. Digital implementation of modems allows more complicated operations with a greater level of flexibility but it requires a great amount of signal processing in the digital domain. The progress of DSP techniques and the improvement in the VLSI technologies, level of integration and simulation tools, allow to achieve very attractive performances in term of size, mass and power requirements. With the advent and maturity for space applications of VLSI technology it has become feasible, from a technical and economical point of view, to design equipments which require signal processing loads inconceivable just few years ago. Alenia Spazio is studying and developing a digital FM/BPSK - FM/FSK demodulator for TT&C transponders whose main system requirements are reported in Table 1. Figure 1 shows the receiver block diagram and the relevant interfaces.

An all digital architecture allows to meet the following advantages:

- easy receiver reconfigurability with different interface requirements according to the mission;
- few-bit signal quantization;
- subcarrier & ranging tone frequency flexibility;

PARAMETER	VALUE
Frequency Range	C-Band (5925 to 6425 MHz) Ku-Band(14000 to 14500 MHz)
Input Power	-70 to -110 dBm
TLC Modulation	FM 400 KHz peak deviation Subcarrier frequency: -BPSK 8/16 KHz -FSK up to 25 KHz Data Rate up to 500 bps Ranging
Ranging Modulation	FM 400 KHz peak deviation Tone frequency up to 100 KHz

Table 1: Main system parameters of TT&C FM receiver for satellite application

- data rate flexibility;
- reduced tuning activity.

The paper is focused on the theoretical study, design, implementation and performances of the digital FM/BPSK-FM/FSK demodulator to be implemented using multi-rate digital signal processing in a CMOS ASIC.

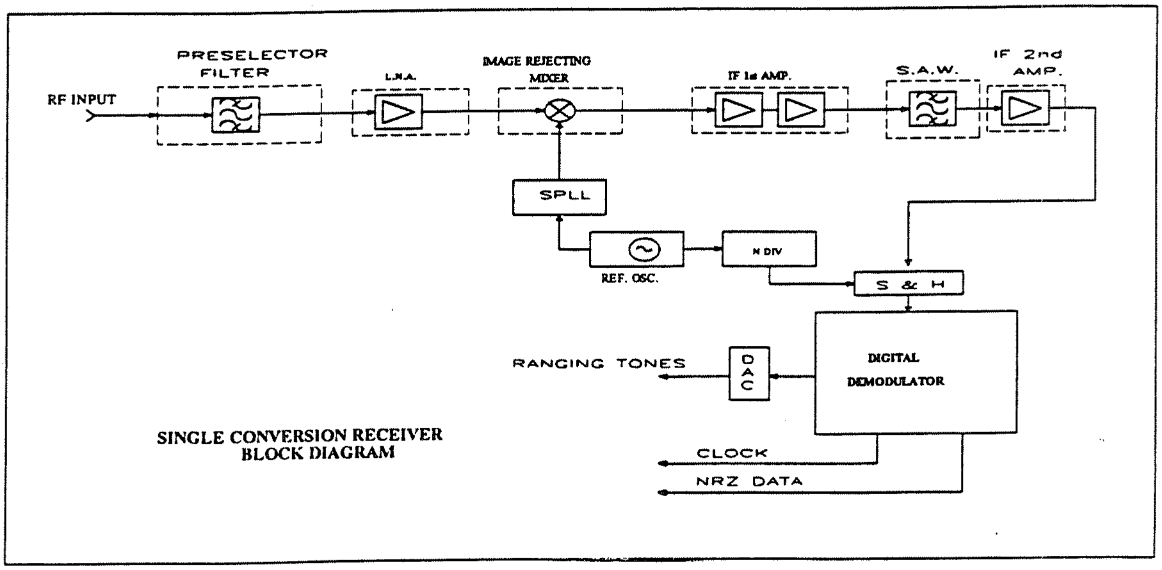


Figure 1: Receiver architectural block diagram

2 FM Demodulation: approach to the digital solution

A. Signal description. The frequency Modulated signal can be represented as in the following:

$$V_{FM} = V_P \cdot \cos(2\pi \cdot F_C t + \frac{\Delta f_p}{F_S} \cdot \cos(2\pi \cdot F_S t)) \qquad (1)$$

where:

$$\begin{aligned} V_p =& \text{ carrier peak amplitude} \\ F_C =& \text{ carrier frequency} \\ \Delta f_p =& \text{ FM peak frequency deviation} \\ M = \tfrac{\Delta f_p}{F_S} =& \text{ Modulation Index} \end{aligned}$$

As shown in Figure 2, this signal can be demodulated using a derivative detector followed by an envelop detector.

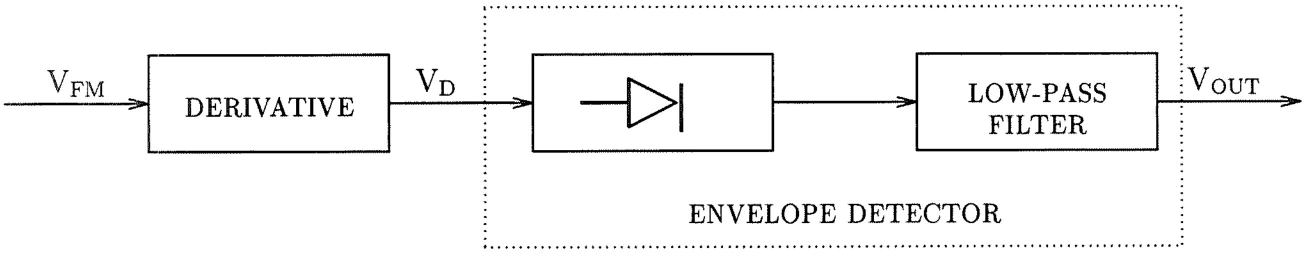


Figure 2: FM demodulator

The analytical representation of the signals is:

$$\begin{aligned} V_D(t) =& \tfrac{d(V_{FM}(t))}{dt} = -V_P[2\pi \cdot F_C - 2\pi \cdot \Delta f_p \cdot \sin(2\pi \cdot F_S t)] \\ & \cdot \sin(2\pi \cdot F_C t + \tfrac{\Delta f_p}{F_S} \cos(2\pi \cdot F_S t) \end{aligned} \qquad (2)$$

$$\begin{aligned} V_{OUT}(t) &= -V_P[2\pi \cdot F_C - 2\pi \cdot \Delta f_p \cdot \sin(2\pi \cdot F_S t)] \\ &= -V_P 2\pi \cdot F_C + V_P 2\pi \cdot \Delta f_p \cdot \sin(2\pi \cdot F_S t) \\ &= DC + Information \end{aligned} \qquad (3)$$

B. Theoretical performances. The performances of the FM demodulator are hereafter reported. These performances are relevant to an ideal demodulator and can be utilised to verify the degradations of the digital approach.

The IF signal at the demodulator input is characterised by the following expressions:

$$S(t) = V_{FM}(t) + n(t) \qquad (4)$$

$$SNR_{IF} = IF\ Signal\ to\ Noise\ Ratio\ = \frac{V_P^2}{2N_O B_{IF}} = \rho \qquad (5)$$

where:

- $n(t)$ is a band-path process and can be assumed as Additive White Gaussian Noise
- N_O = one-side noise spectral density
- B_{IF} = IF receiver bandwidth

The signal power at the demodulator output is: $\frac{\Delta f_P^2}{2}$. The Noise Power spectral density at the demodulator output in case of high If signal to noise ratio (greater than 10 dB) is:

$$n_{u-1}(f) = \frac{N_O f^2}{V_P^2} \tag{6}$$

As the IF signal to noise ratio drops from 10 to 0 dB, the output signal power remains nearly unchanged, while the output noise increases. A second contribute to the noise spectral density arises; in case of ideal IF filtering it can be expressed as in the following:

$$n_{u-2}(f) = \frac{B_{IF}}{2\sqrt{3}}\left[1 - erf(\sqrt{\rho})\right] \tag{7}$$

The overall noise power in the video bandwidth around the subcarrier frequency F_S is:

$$N_U = 2\int_{F_1}^{F_2} \left[n_{u-1}(f) + n_{u-2}(f)\right] df \cong 2\frac{N_O F_S^2 B_V}{V_P^2} + 2\frac{B_{IF} B_V}{2\sqrt{3}}\left[1 - erf(\sqrt{\rho})\right] \tag{8}$$

$$F_1 = F_S - \frac{B_V}{2} \quad F_2 = F_S + \frac{B_V}{2}$$

The above approximation is correct in case of narrow video bandwidth (B_V) around the central frequency (F_S). From the above expressions we can derive the Video Signal to Noise Ratio (SNR_V) at the FM demodulator output:

$$\begin{aligned} SNR_V = \frac{\frac{\Delta f_P^2}{2}}{N_U} &= \frac{\frac{\Delta f_P^2}{2}}{2\left\{\frac{N_O F_S^2 B_V}{V_P^2} + \frac{B_{IF} B_V}{2\sqrt{3}}\left[1 - erf(\sqrt{\rho})\right]\right\}} \\ &= \frac{\frac{\rho}{2}\left(\frac{\Delta f_P}{F_S}\right)^2\left(\frac{B_{IF}}{B_V}\right)}{1 + \frac{1}{\sqrt{3}}\left(\frac{B_{IF}}{F_S}\right)^2 \rho\left[1 - erf(\sqrt{\rho})\right]} \end{aligned} \tag{9}$$

In Figure 3 the theoretical value of SNR at FM demodulator output is plotted as function of input SNR and for two different values of subcarrier frequency (16 and 27 KHz). The selected values for the IF bandwidth, the video bandwidth and the frequency deviation are:

$$B_{IF} = 1.3MHz, \; DF = 400KHz, \; B_V = 200H$$

On the x axis the IF signal to noise ratio (SNR_{IF} is reported in dB, and on the y axis the video signal to noise ratio is shown in dBHz (subcarrier power on noise spectral density) . The theoretical plot seems to characterise quite well the curve obtained via experimental results on an analog FM demodulator. These curves show two different slopes:

- the first slope is relevant to high IF Signal to noise ratio (over than 10 dB). This is called the **Linear Region**, infact the output SNR is proportional to the input SNR. In this area the output signal to noise ratio can be represented as in the following:

- $$SNR_V = \frac{\frac{\Delta f_P^2}{2}}{\frac{N_O F_S^2 B_V}{V_P^2}} = \frac{\rho}{2}\left(\frac{\Delta f_P}{F_S}\right)^2 \left(\frac{B_{IF}}{B_V}\right) \qquad (10)$$

- as the IF signal to noise ratio drops below 10 dB, there is a sharp increase in the output noise power with a reduction of output SNR (**Noise Increasing Region**).

The above expression (9) is not correct for IF signal to noise ratio below 2 dB. As ρ drops below 2 dB the output noise power levels off a large value, but the output signal is being suppressed. This is called the **Signal Suppression Region** and the output SNR is proportional to SNR_{IF}^2. It must be underlined that the subcarrier frequency value doesn't affect the performances of an FM demodulator in the Noise Increasing Region (RF SNR < 9 dB).

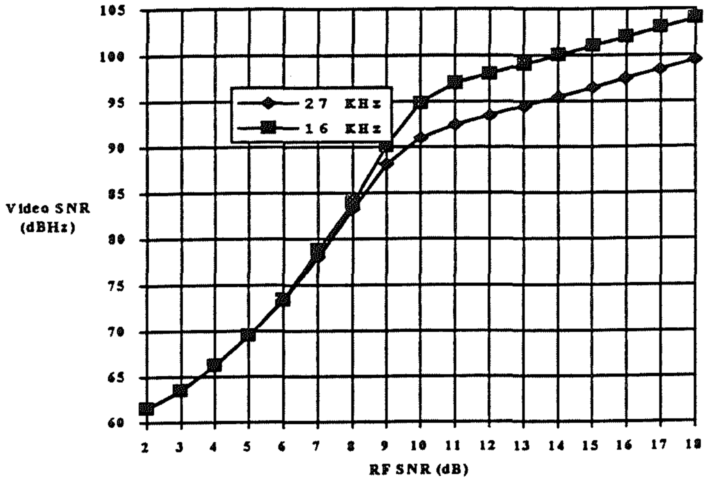


Figure 3: Theoretical Signal to Noise Ratio at Demodulator Output for two subcarrier frequency values

C. **Digital solution.** A simple and economical way to realise an FM demodulator in presence of high frequency deviation (Δf_P) is reported in the Figure 4, where:

- the derivative is approximated by the incremental ratio
- the diode is replaced by the absolute value
- the low pass filter is replaced by the integrator and dump (I/D).

The key point of this approach is the realisation of the derivative detector with the incremental ratio. Because of the signal information is included in the carrier frequency, an interesting solution in terms of complexity reduction (hardware saving) is relevant to the 1-bit A/D converter. In this case the signal processing operations are reported in Figure 5.

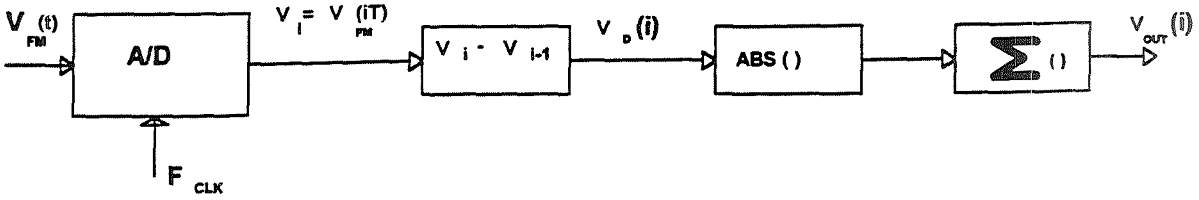


Figure 4: Digital FM demodulator

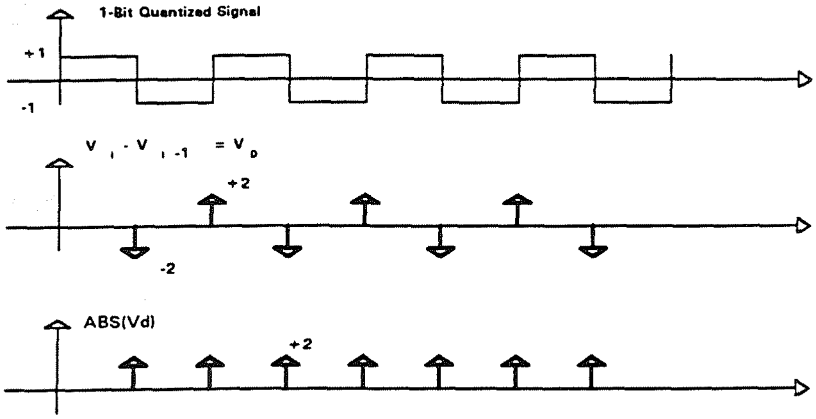


Figure 5: DSP for 1-bit quantization case

The absolute value detects the zero crossing of input carrier and the I/D filter accumulates the transition number over a fixed integration time. The number accumulated by the I/D filter depends on the input carrier frequency. If the modulation is absent the demodulator output is a constant value related to the carrier frequency. When the frequency modulation is present, the demodulator outputs is constituted by the modulating signal oscillating around the carrier frequency constant value. This FM demodulator is a frequency counter. It's obvious that the A/D sampling time must be greater than the carrier frequency to detect all the signal transitions for a good frequency estimation.

D. Analog to Digital conversion. The FM received signal is amplified, S.A.W. filtered (1.5 MHz bandwidth) and frequency converted at a proper IF frequency (Fif) ready to be quantised. One of the advantages of this architectural solution is the realisation of IF sampling via hard quantization. This technique allows to use higher processing speed without traditional high consuming flash A/D. The bandpass sampling technique (down- sampling) has been used with the removal of IF/VIDEO analog downconversion and with the simplification of the receiver analog section. In the downsampling scheme the A/D converter clock frequency is less than the input carrier frequency. In this case the analog down-conversion from Fif to baseband is avoided, since the particular sampling process produces intrinsically the spectrum translation at lower frequency (to be processed by digital section).

3 ASIC Architectural Description

Figure 6 shows a top level diagram of the FM/FSK-FM/BPSK fully digital demodulator and its relevant interfaces. The demodulator chip is constituted by the functional blocks included inside the dot line. The main receiver signal processing functions are:

- FM demodulator with Subcarrier and/or Ranging Tone extraction
- Filtering section
- BPSK and FSK signal demodulator
- Bit-Synchroniser

The FM demodulator section is realised as reported in Figure 4. The output signal is constituted by the modulated data or the ranging tone ready to be converted in the analog domain.

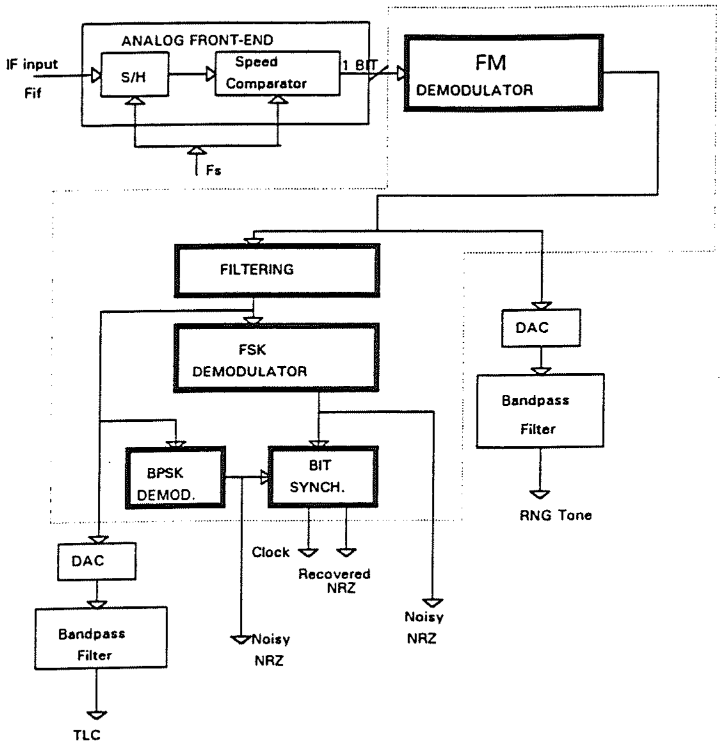


Figure 6: Digital demodulator: top level diagram

The filtering section at the FM demodulator output is in charge of removing the DC component and increasing the signal to noise ratio for the modulated data at the FSK or BPSK demodulator input. The filter is a second order bandapss IIR structure realized in second direct form with linear phase characteristic in the band of interest. To simplify the hardware design the filter coefficients have been approximated as power of two.

The FSK filtered signal is demodulated with the same simple and quite effective architecture used for FM demodulation. The output signal is a biased noisy NRZ. The DC component is due to this demodulation scheme and its value is related to the central frequency (F1 + F2)/2 where Fi is the frequency of ith tone. This component is removed via bias-subtraction instead of high pass filtering, resulting in considerable hardware simplification. In this case the hypothetical high pass filter would require a great hardware effort for the numeric representation.

The bias-subtraction choice is a consequence of two major motivations:

- the independence from data stream transition density;
- the high sampling rate in terms of number of samples per bit to optimise Bit-synchroniser performances.

As reported in Table A the standard signal foresees a BPSK modulation as well. In this case a coherent demodulation is realised via a digital Costas loop, the subcarrier phase is recovered and the data (Noisy NRZ) are present on the in-phase arm. In both cases (FSK or BPSK modulation) the noisy NRZ signal is routed to the Bit Synchroniser circuit for bit and clock recovery. The Bit Synchroniser is a digital data transition tracking loop. This approach allows to realise a full flexible architecture in terms of:

- subcarrier and/or ranging tone frequency
- subcarrier modulation (BPSK or FSK)
- data rate.

The receiver is configured according to the system requirements simply changing the ASIC control word which contains informations about mission parameters such as:

- I/D dump rate
- Bit Synch. NCO frequency for the data rate selection
- Noisy NRZ bias.

Besides the digital approach to FM demodulation and data recovery minimises the testing effort with respect to the analog counterpart.

4 Performance Results

Figure 7 shows the test bed configuration, whose main task is the generation of the proper signal at digital demodulator input. Figure 8 shows the IF signal spectrum for signal to noise ratio of 10 dB about. The FM signal is centred around 70 MHz with a peak frequency deviation of 400 KHz.

The IF signal is down-sampled and 1 bit quantized; the selected sampling frequency is in the range of 34 MHz. The signal samples are routed to the digital demodulator for FM demodulation. The Frequency Demodulator section is the key point of the receiver digital signal processing. The measured performances are plotted in Figure 9, on the x axis the input SNR is reported, while on the y axis the subcarrier to noise spectral density at demodulator output is represented. The degradations with respect to the theoretical values can be deduced in the non linear region (2-7 dB of input SNR) which is the typical range for the TT&C application.

From Figure 10 through Figure 13 the FM demodulated signals are shown for different frequency values and for an IF signal to noise ratio of 5 dB. These frequency values are typical for the ranging tone applications.

- B.E.R. measurements for FM/FSK signal are reported in tables B and C.
- Table B is relevant to a frequency deviation of 1 KHz and a bit rate of 125 bps.
- Table C is relevant to a frequency deviation of 5 KHz and a bit rate of 500 bps.
- Figure 14 summarises the B.E.R. performances.

These results outline the selected architecture as the best compromise in terms of demodulator output performances (subcarrier/ranging tone SNR, Bit Error Rate) versus receiver flexibility, hardware complexity and power saving.

SNRif	B.E.R.
-1	1.1e-1
0	5.6e-4
1	5e-4
2	3e-5
3	2e-6

Table 2: B

SNRif	B.E.R.
-1	5e-2
0	4.5e-4
1	3.5e-6
2	<5e-7 (*)

(*) No errors detected in 5 hours.

Table 3: C

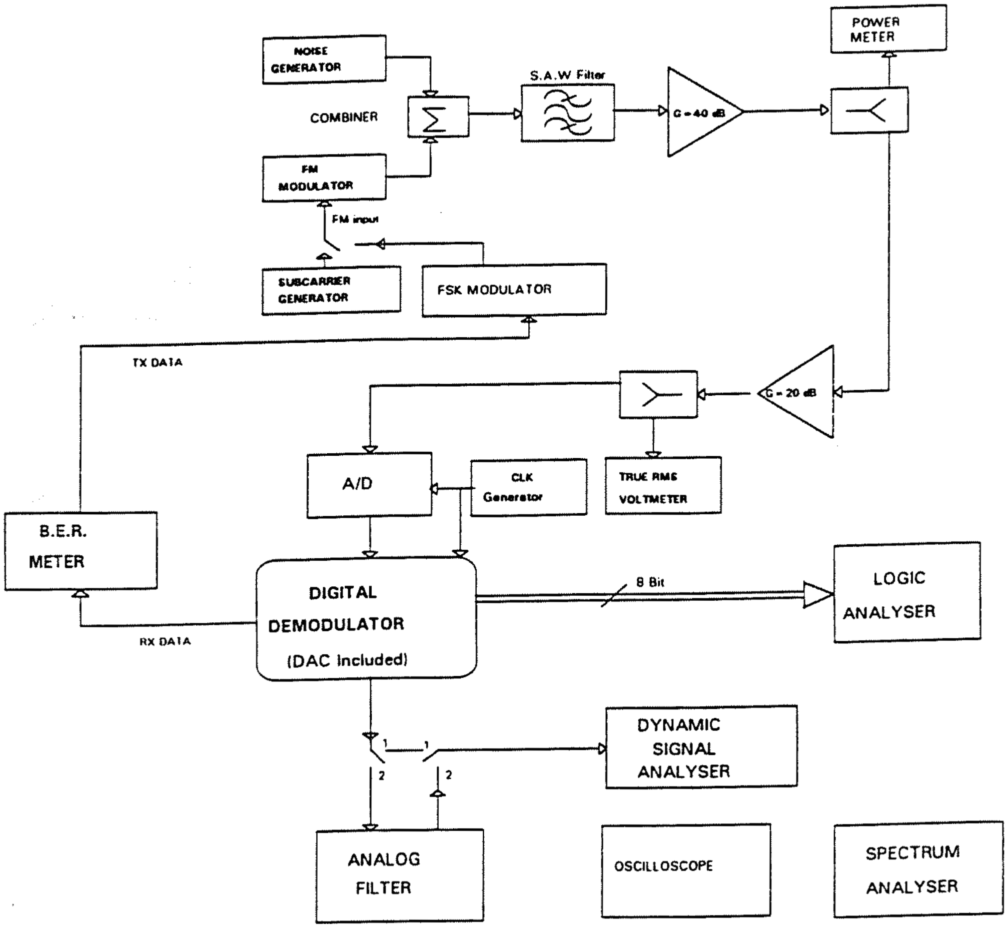


Figure 7: Test-Bed Block Diagram

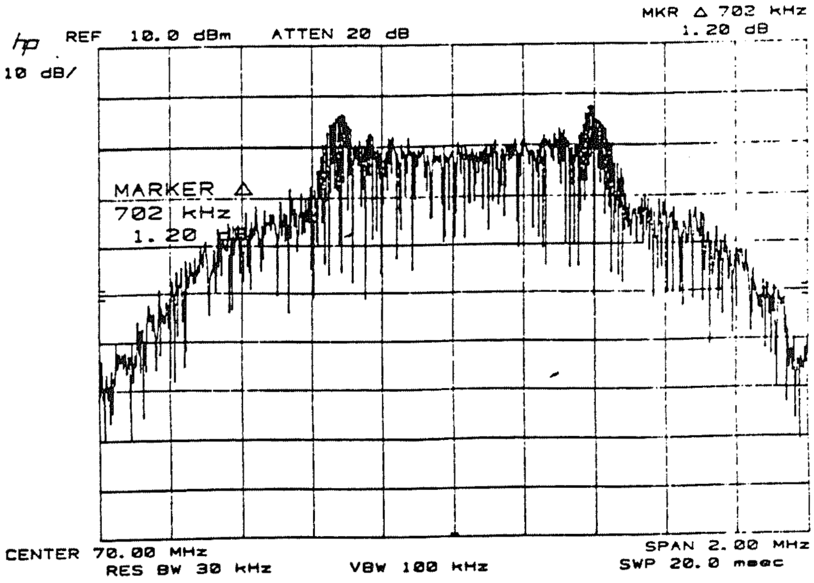


Figure 8: Equipment generated spectrum.

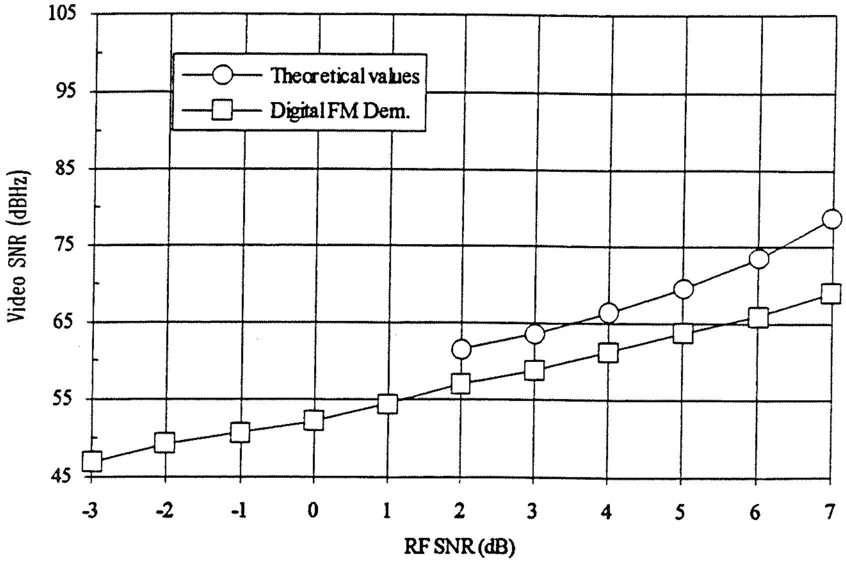


Figure 9: Digital FM receiver performances

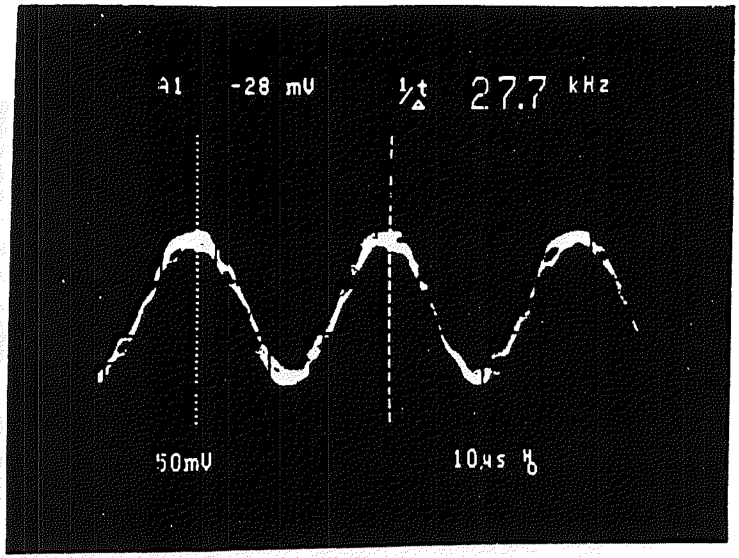


Figure 10: Demodulated Ranging Tone (Tone freq = 27.7 KHz, SNRif = 5 dB)

3.4.12

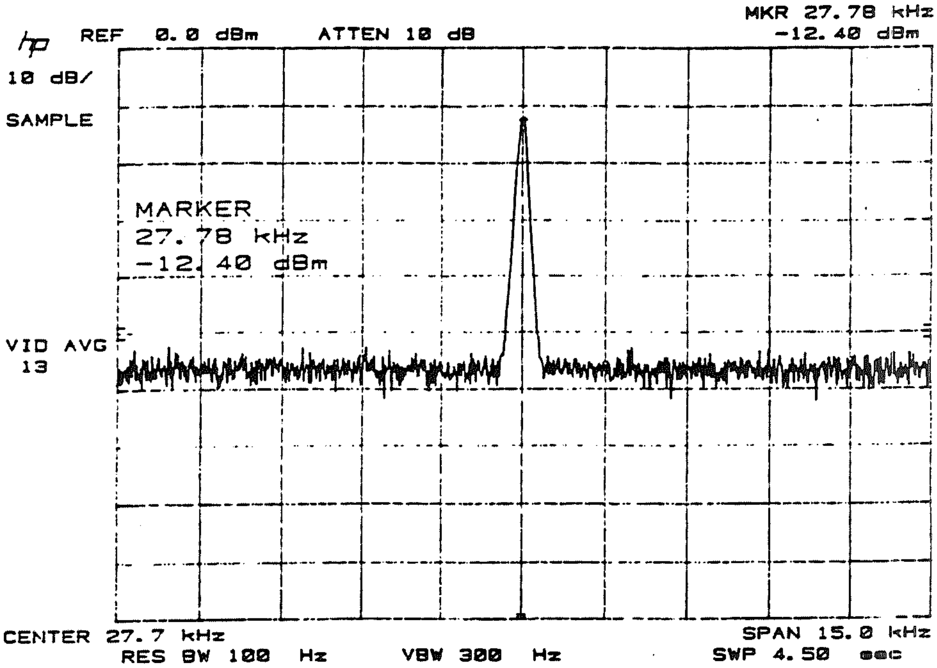


Figure 11: Demodulated Ranging Tone Spectrum (Tone freq = 27.7 KHz, SNRif = 5 dB)

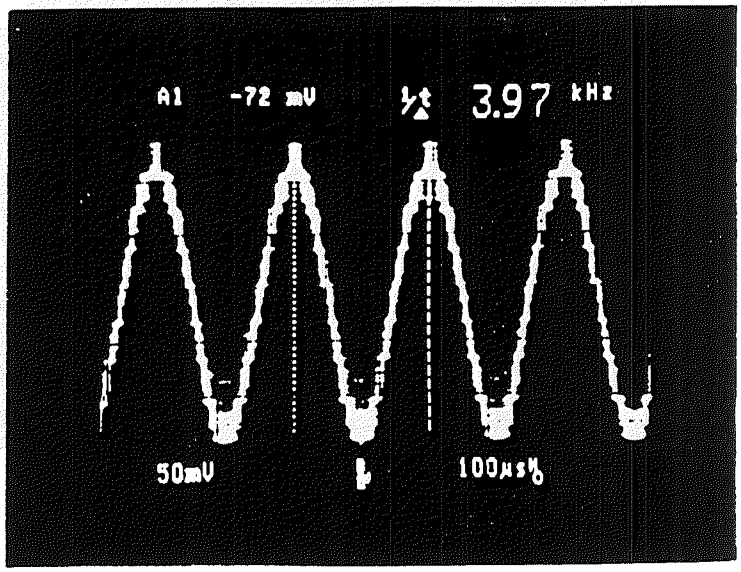


Figure 12: Demodulated Ranging Tone (Tone freq = 3.97 KHz, SNRif = 5 dB)

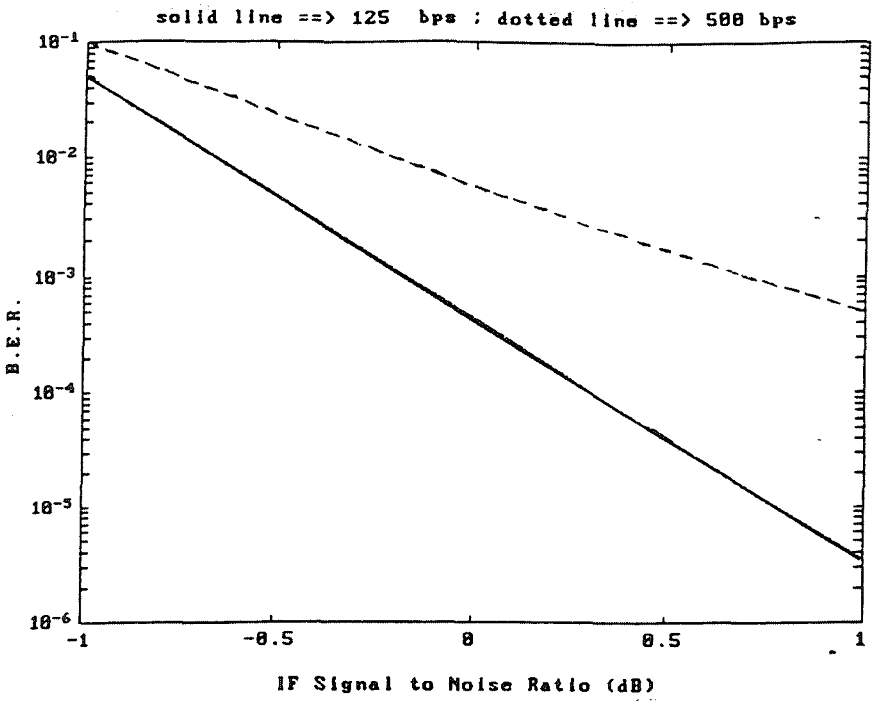


Figure 13: Demodulated Ranging Tone Spectrum (Tone freq = 3.97 KHz, SNRif = 5 dB)

5 Conclusion

A FM/FSK-FM/BPSK fully digital receiver for TT&C space applications has been described. It represents the best compromise in terms of complexity, performances, and level of integration with respect to conventional analog detectors and to digital schemes as reported in [6],[7] and [8]. The paper is mainly focused on a new approach to non-coherent frequency detection. Emphasis has been given to the FM and FSK demodulators with respect to the other well known circuits such as the digital Costas Loop [3] and the Data Transition Tracking loop [2]. Some considerations leading to the final architecture have been depicted and the major test results of the implemented experimental Breadboard have been shown. An ASIC chip is actually being implemented with a predicted die Area occupation of about 20000 gates.

References

[1] P.F.Panter,"Modulation, Noise, and Spectral Analysis", New York, Mc Graw Hill, 1965.

[2] J.J.Spilker,Jr. ,"Digital Communications by Satellite", Prentice Hall, 1977.

[3] [3] M.K.Simon,W.Lindsey, "Optimum Performance of Suppressed Carrier Receivers with Costas Loop Tracking", IEEE Transaction on Comms., Vol.Com-25, No.2, Febr.77.

High Performance MPEG-Audio Decoder IC

M. Thorn, G. Benbassat, K. Cyr, S. Li, M. Gill,
D. Kam, K. Walker, P. Look, C. Eldridge and P. Ng
Digital Compression, Semiconductor Group
Texas Instruments

Abstract - **The emerging digital audio and video compression technology brings both an opportunity and a new challenge to IC design. The pervasive application of compression technology to consumer electronics will require high volume, low cost IC's and fast time to market of the prototypes and production units. At the same time, the algorithms used in the compression technology result in complex VLSI IC's. The conflicting challenges of algorithm complexity, low cost and fast time to market have an impact on device architecture and design methodology. The work presented in this paper is about the design of a dedicated, high precision, MPEG audio decoder.**

1 Introduction

One of the compression standards that is being widely adopted by companies in the consumer electronics, computer and telecommunications industry is the International Standard Organization(ISO)'s Motion Picture Expert Group (MPEG) compression standard for digital audio and video signals [1]. The ISO standard specifies the three parts that can make up an ISO MPEG data stream: compressed video data, compressed audio data and the system coding for the multiplexing and synchronization of the audio and video data. The standard allows a variety of bit rates, 32kbps to 448kbps, and PCM sampling frequencies, 32kHz, 44.11kHz, 48kHz, for the audio data.

The audio compression standard is based on a psycho-acoustic model [2] of the ear. The input digital audio signals, PCM, are split into 32 subbands using an analysis filter bank. The subband filter outputs are then down-sampled and quantized using dynamic bit allocation in such a way that the quantization noise is masked by the sound and remains imperceptible. The psycho-acoustic model is used to control the quantization and bit allocation of the encoder. These quantized and coded samples are then packed into audio frames that conform to the ISO MPEG standard requirements.

An ISO MPEG stream consists of 1 or more audio and video streams multiplexed together. The basic presentation unit for the audio bit stream is an audio frame and the audio data stream is composed of a sequence of frames. An audio frame corresponds to a fixed number of audio PCM samples. The side information is also transmitted along with the coded audio sub-band samples in a frame. The side information is the information necessary for controlling the decoder such as the bit allocations for each subband sample, scale factors, etc.

The ISO MPEG standard incorporates a number of variants: 3 possible PCM sampling frequencies, 32kHz, 44.11kHz and 48kHz; a wide range of bit rates, 32 kbps to 448kbps; and three "layers". The three layers differ in complexity of the encoder and performance. Layer

II is a more efficient but more complex coding scheme than layer I. The audio data framing also differs for the 3 layers. For example, a Layer I frame consists of 384 samples and a Layer II frame consists of 1152 samples.

The decoder IC can accept any layer 1 and 2 MPEG compliant bit stream. It accepts a system bit stream, extract and decode the audio portion and provide the necessary hooks for audio-video synchronization. The decoder IC can also decode audio only bit streams.

2 Architecture

The decoder IC can accept input data streams in the following formats: full ISO 11172 MPEG system stream, MPEG audio packets, MPEG audio frames and a bypass mode allows the inputing of uncompressed PCM data. It will decode mono, dual, stereo and joint stereo modes. The IC supports all MPEG PCM sampling rates and input data rates including free format. The decoder IC accepts a system data stream, extract and decode the audio portion and provide the necessary hooks for audio-video synchronization.

The decoder IC has two very different tasks to perform and the architectural hardware tradeoffs for implementation are discussed. First, it must decode the content of the bit stream and dequantize the sub-band samples according to the header and side information for each frame of data. Second, it must reconstruct the PCM audio signal from the subbands using a subband synthesis filter.

Error concealment and synchronization are two accessory tasks that the decoder must perform. If the decoder detects an error in the bitstream then it proceeds to an error concealment scheme, such as muting or replaying last frame to minimize the effect of the transmission error. The audio IC must also be capable of buffering enough data and controlling its flow such that synchronization of audio and video can occur. With an optional external DRAM the decoder IC can support up to 1 second of audio delay for Audio/Video synchronization.

The architectural considerations were whether to implement the design using a single, programmable processor; a non-programmable, hardware dedicated solution or a combination of the two. Die area, cost and performance considerations factored into the decision.

In order to provide 18 bit precision PCM output, the subband filtering calculations must be done with the appropriate precision. If the arithmetic unit is done as dedicated hardware then advantage can be taken of the relatively low speed required, approximately 200 ns per multiply, to reduce the cost of the arithmetic unit. The regularity of the calculations also makes it appropriate for a dedicated hardware solution.

The other tasks of bitstream decoding, error concealment and dequantization are better handled by a programmable architecture. However, the audio channel to be decoded may be multiplexed with other audio channels and/or video channels. As a consequence, the bit stream being received can be at bit rates ranging from 1.5 Mbps to 15 M bits per sec. So, the precision, datapath width, and the number of arithmetic operations required by the subband filtering when combined with the requirements a bursting input stream which must be demultiplexed would result in a high precision, very high speed programmable processor. Therefore, a mixed architecture combining dedicated hardware for the arithmetics and programmable processor for input stream decoding was chosen.

Figure 2 is a functional block diagram of the MPEG audio decoder. The design is composed of multiple autonomous units working together through shared memory. The operation of each unit is data-driven. Each unit starts processing whenever data arrives at its input buffer and it writes the processed data to its output buffer.

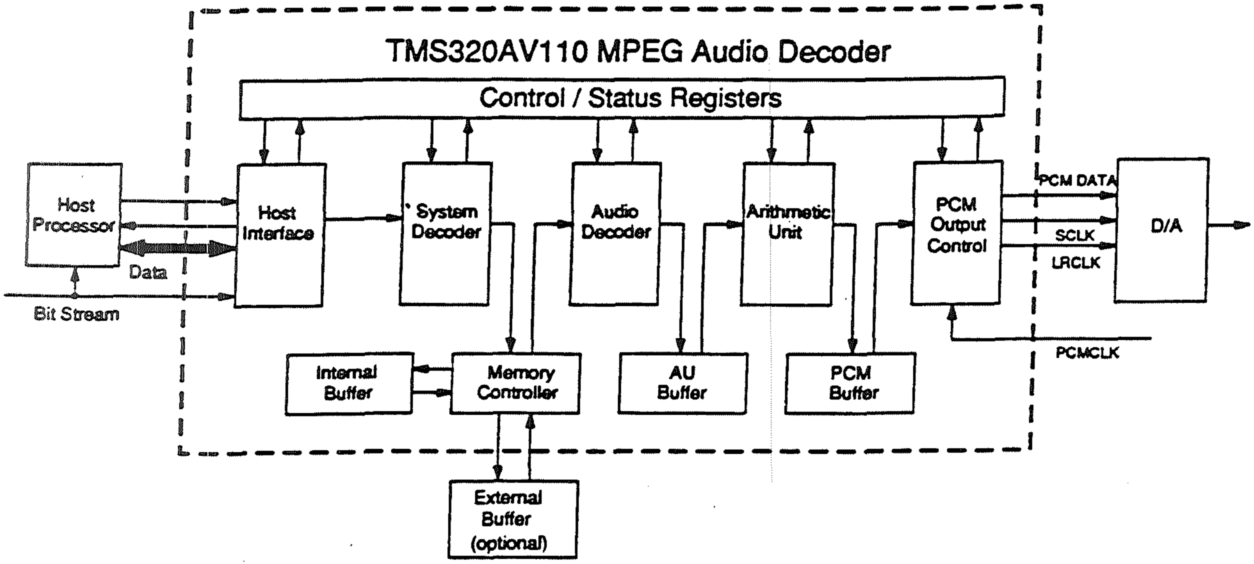


Figure 1: TMS320AV110 MPEG Audio Decoder Block Diagram

A typical operation cycle is as follows: Coded data arrives at the Host Interface asynchronously to the decoder processing clock. The Host Interface synchronizes the incoming data to the decoder processing clock and transfers the data to the System Decoder. The System Decoder extracts the audio data for the desired audio channel from the MPEG data stream and writes it to the internal or external audio buffer. The Audio Decoder reads the compressed data from the buffer, performs various decoding operations, and writes the unpacked frame data to the Arithmetic Unit buffer. The Arithmetic Unit performs subband synthesis filtering and outputs the reconstructed PCM samples to the PCM buffer. The PCM output module shifts the samples out to an external D/A converter and also provides the left/right channel clock and the PCM bit clock.

3 Module Descriptions

3.1 Host Interface:

The host bus interface handles all communication with the user, both for compressed audio data input and for control information. It consists of an 8-bit generic microprocessor interface, a DMA interface, a serial data input interface, and a number of individual control lines and internal registers. Interrupt capability to the host processor is also provided, with a single interrupt line driven by an interrupt and an interrupt mask register. The host interface also provides a synchronization function since the data transfer on the bus is asynchronous to the IC processing clock.

3.5.4

3.2 Control/Status Registers:

The Control/Status Registers module contains control registers for configuring the device for various options/features and status registers for indicating the status of the decoding process. These registers can be written to or read from via the host bus interface. A chip select and a seven bit address are used to address the registers.

3.3 System Decoder:

The System Decoder module has two primary tasks. The first task is to decode a full MPEG stream which is a mixture of audio, video, and private streams and extracts the audio portions of it and pass them over to the memory controller. The second task is to decode the timing information, time stamps, and tag the audio data with the appropriate time stamp for audio-video synchronization. The System Decoder also decodes subsets of MPEG streams and when the IC is configured for PCM bypass mode then it will process PCM data.

3.4 Memory Controller:

The Memory Controller block handles data read/write to and from either the external DRAM buffer or the internal buffer. It is designed to support 1 megabit or 4 megabit external DRAM. If no external DRAM is detected by the Audio Decoder then the smaller internal buffer will be used. In addition, the Memory Controller arbitrates memory access request conflicts between the System Decoder and Audio Decoder.

3.5 Audio Decoder and Microprogram:

The Audio Decoder module is a programmable processor with dedicated hardware and instructions customized for MPEG audio frame decoding. It is a 16 bit accumulator machine with single operand instructions. The module contains a 16 bit full function ALU, a sequential field extraction unit, an instruction sequencer, RAM, ROM, and associated random logic. The microcodes are 18 bits wide with a 5 bit instruction field, a 2 bit operand type field and a 11 bit operand field. There are a total of 28 instructions and 6 of them are special instructions with dedicated MPEG bit-stream decoding functions. A block diagram of the Audio Decoder and a listing of the instructions are shown in Figure 2 and 3.5.

The Audio Decoder is a microcoded machine in order to provide flexibility and features in implementing the various functions in the decoding process. A simplified flowchart (Figure 4) is used to explain the functions of the major software blocks. The microprogram is contained in ROM.

An initialization sequence is executed during chip reset or restart. The initialization process sets up a number of the program constants and parameters. The audio decoder automatically detects the size of the input buffer and also determines whether an external DRAM is being used. The use of external memory allows more robust error concealment techniques to be employed.

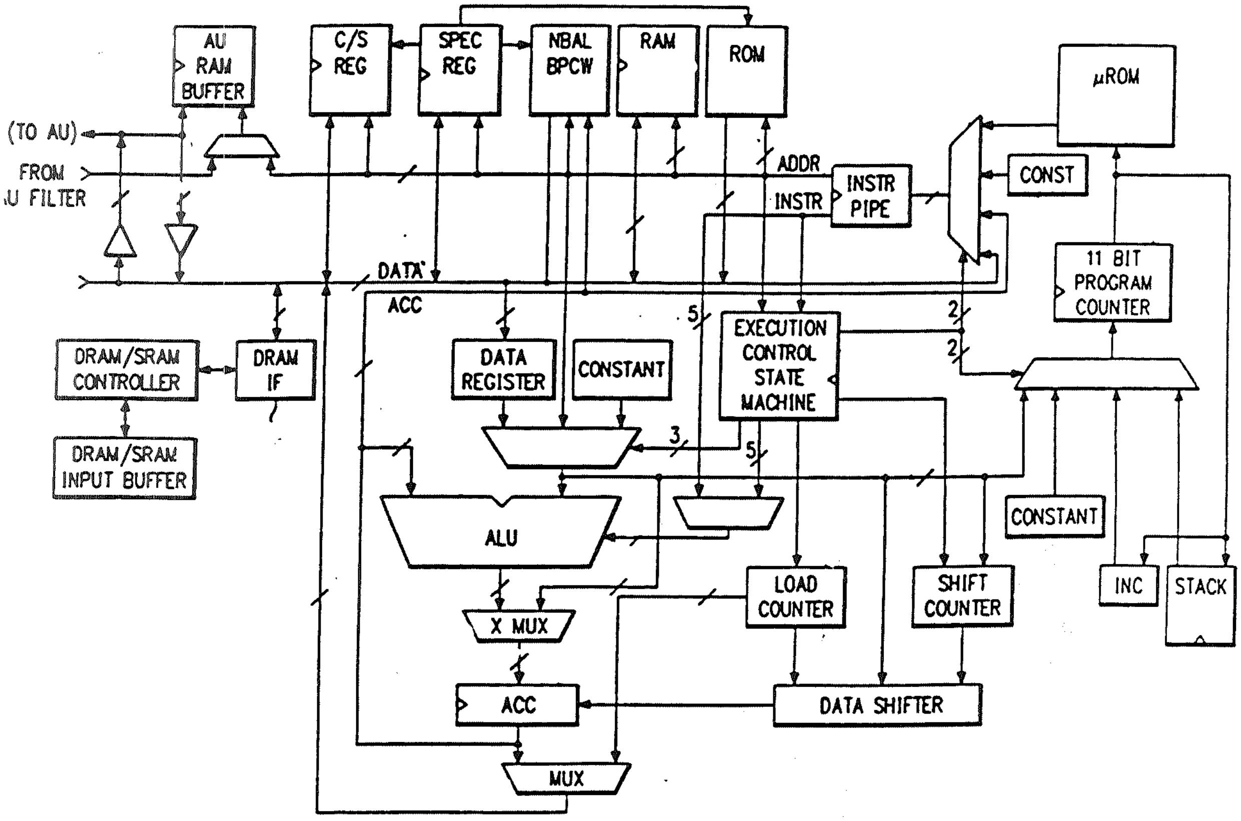


Figure 2: Audio Decoder Module Block Diagram

Menomic	Description
ADD	Acc + Op → Acc
SUB	Acc − Op → Acc
AND	Acc AND Op → Acc
OR	Acc OR Op → Acc
XOR	Acc XOR Op → Acc
SRL	Logic right shift Acc
SLL	Logic left shift Acc
SRC	Circular right shift Acc
SRA	Arithmetic right shift Acc
GSYN	Search for Syn Word
GBT	Get data bits from input buffer
GBTC	Get and CRC check data bits from input buffer
GBTF	Flush shifter/Get data bits from input buffer
GANC	Get bits and write to ancillary data buffer
BZ	Branch on zero
BNZ	Branch on not zero
BO	Branch on overflow
BNEG	Branch on negative
BC	Branch on carry-out
BPOS	Branch on greater than zero
B	Unconditional branch
CALL	Branch and push PC to stack
READ	Input Buffer read
WRT	Input Buffer write
RET	Pop PC from stack
GTBL	ROM Table → Acc
LDA	Op → Acc
STA	Acc → Memory

Figure 3: Audio Decoder Instructions

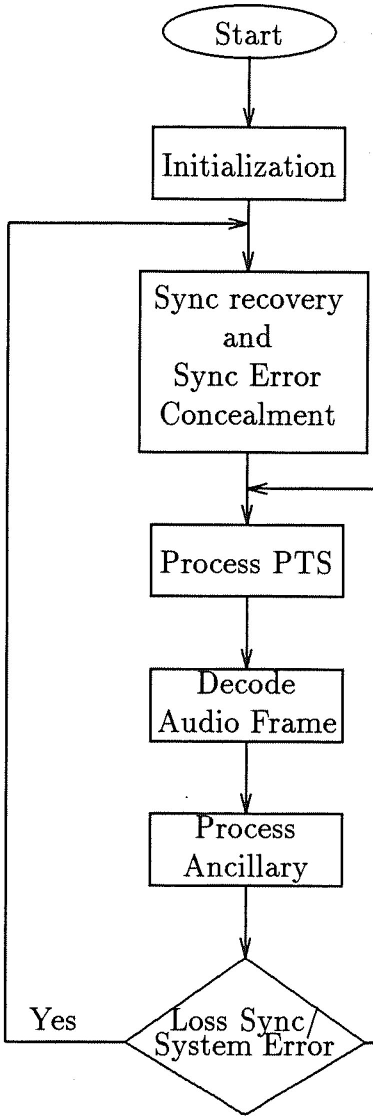


Figure 4: MPEG Audio Frame Decoding Flowchart

3.5.8

There are three different synchronization status (unlocked, attempt to lock, and locked) for the audio decoder. The sync status is unlocked after the initialization sequence and the audio decoder attempts to find a synchronization word (0xFFF) within the audio bit stream. The audio decoder may search for up to a maximum of four consecutive sync words before moving into a locked sync status. Normal decoding of an audio frame is performed only during a locked sync status. Sync error concealment is performed during an attempt to search for a locked sync state. There are four types (ignore error, mute, repeat, and skip) of sync error concealment modes supported by the audio decoder. The ignore error mode simply ignores the error and continues with decoding the frame. The mute mode causes the frame to be muted. The repeat mode is only supported with external DRAM and the previous valid frame is decoded instead of the current frame. The skip mode causes the current frame to be skipped.

The search for a possible presentation time stamp, PTS, is always performed before the decoding of any normal audio frame. If a PTS is found to be associated with the current frame, then a PTS tag will be set to mark the frame when it appears at the PCM output. Presentation time stamps are used to synchronize audio and video frames.

During the normal decoding of audio frames, the latency feature allows the audio decoder to search for the next sync word before decoding the current frame. The audio decoder also supports CRC detection. If a CRC error is detected, then the decoder enables the CRC error concealment procedure. There are four types (ignore error, mute, repeat, and skip) of CRC error concealment mode that are similar to those of the sync error cases. If there is no external memory and the repeat mode is selected then the audio decoder defaults to mute mode for both CRC and sync error cases.

A complete audio frame is decoded after the last bit of ancillary data is processed. A rather elaborate header syntax check and sync word confirmation is then performed on the immediate frame to be decoded. If there is no lost sync or syntax error condition, the decoding process continues in a loop. If a sync word is in error or syntax error is detected in the header, the decoder moves into the process of sync error recovery.

3.6 Arithmetic Unit:

The Arithmetic unit is a hardwired math engine that performs the dequantization and subband synthesis filtering. The module receives quantized samples and scale factors from the audio decoder by means of the AU buffer memory. After the audio decoder has written a block of samples into the AU buffer and signaled the AU that the buffer is full then it is free to continue decoding the audio input data.

Once the AU has been told that it has a block of data in the buffer it begins the dequantization process. After it has dequantized the samples then the results are written to a scratch subband memory. When all of the 32 input samples are dequantized then the inverse synthesis filter process begins. This process consists of a cosine transform and a FIR filter. The results of the cosine transform are written into the FIR memory. The FIR process then takes the samples from the FIR memory and computes a 16 tap filter before writing a PCM result to the PCM buffer.

In order to meet the PCM output rate the AU must process a block of 32 samples in 8,000 clocks. After summing up the amount of multiplies, adds, and subtracts it was determined that a multiplier with a 5 cycle latency and 4 cycle throughput would be acceptable. The multiplier's last stage is also used to compute adds and subtracts in a single cycle.

The width of the datapath in the arithmetic unit was set so that the result would have 18 bits of precision after processing. This value was determined by comparing the results of fixed point simulations to the results of a similar simulation using double precision floating point. A block diagram of the Arithmetic Unit is shown in Figure 5.

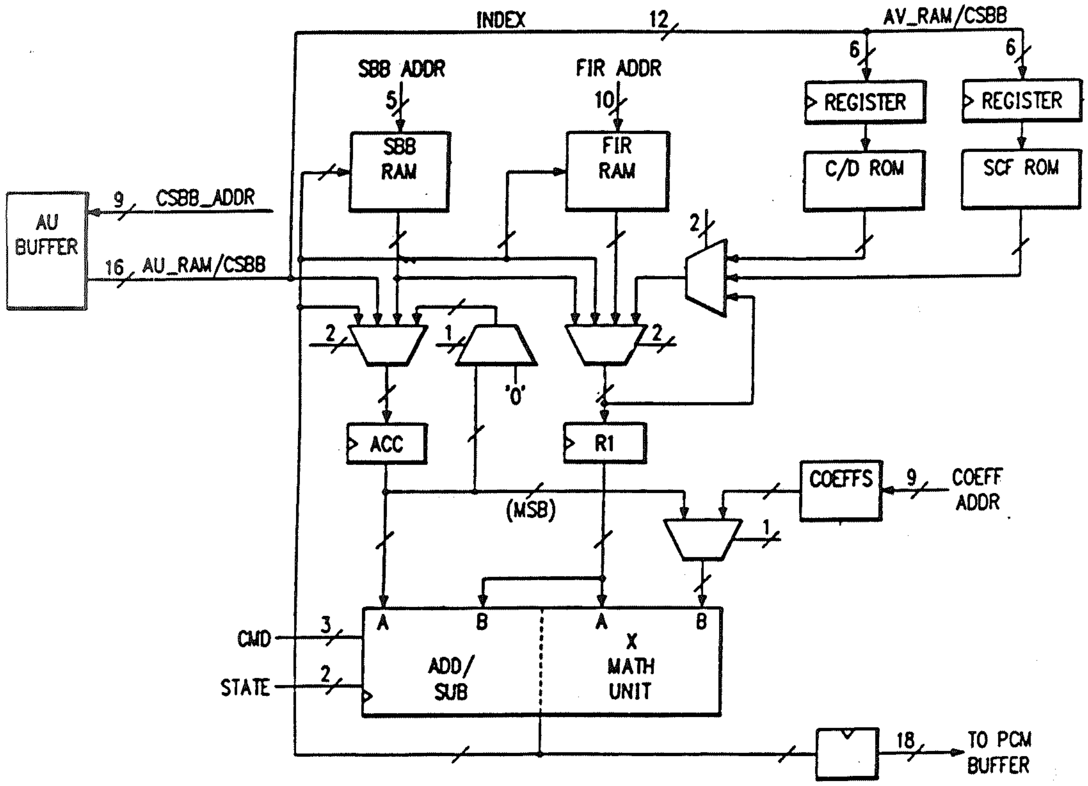


Figure 5: Arithmetic Unit Module Block Diagram

3.7 PCM Output Control:

The PCM module is responsible for outputting the reconstructed audio data. After the AU initially fills the PCM buffer, the PCM module outputs data when PLAY is asserted. Samples are shifted out one bit at a time, with samples alternating between the left and right channels. Data shifts out continuously until PLAY is de-asserted or a PCM buffer underflow occurs. In the event of an underflow, the PCM module outputs zeroes until the AU fills up one-quarter of the PCM buffer.

When the PCM module begins shifting a sample, it determines if there is a PTS or BOF (beginning of frame) associated with that sample. If so, the PCM initiates an interrupt to the user via the host interface and pulses the pin signals BOF and/or PTS.

The PCM module generates a bit clock (SCLK) and a sample clock (LRCLK) to allow direct connection to a D/A converter. Data leaves the chip on the falling edge of SCLK and should be latched by the external system on the rising edge of SCLK. An LRCLK value of 1 implies left channel data, and 0 implies right. When neither PLAY nor MUTE is asserted, the clocks do not run. However, when either PLAY or MUTE is asserted, the following equations describe SCLK and LRCLK:

$$\text{SCLK} = \text{PCMCLK} \ / \ \text{PCMDIV}$$
$$\text{LRCLK} = \text{SCLK} \ / \ (32 \mid 48)$$

Both PCMCLK and the PCMDIV register can be adjusted to set the value of SCLK. The PCM18 register determines how to generate LRCLK from SCLK. For eighteen bit samples (PCM18 = 1), 48 is used. For sixteen bit samples, 32 is used. If eighteen bit samples are selected, then the DIF register determines whether valid data shifts out at the beginning or the end of each LRCLK half period. Independent of the above options, the PCMORD register determines whether a sample's MSB or LSB shifts out first.

4 Design Methodology

A design methodology was chosen that would support a short design cycle time and support a specification that would be evolving and changing during the IC design process. A design methodology was also needed that would support a rapid retargetting of the design to a smaller die size, lower cost solution after the initial IC was proven production worthy. For these reasons and for improved productivity, VHDL, logic synthesis and a Texas Instruments ASIC standard cell 1.0 micron technology were chosen for implementing the design. The design was done completely in this environment and 100was synthesized. No schematic capture tools were used for full chip integration of the modules. The primary CAD tools used for the design were VHDL, Synopsys VHDL System Simulator, Synopsys Design Compiler for synthesis and the Zycad XP200 System Design Accelerator.

In addition to workstation simulation and hardware accelerators such as ZyCAD and IKOS, hardware emulation technology, Quickturn, was also employed in the design verification process. Although emulation technology is known for its pre-silicon functional verification capability, for this project emulation was mostly used for pre-silicon debug of an audio demonstration printed circuit board and as a post-silicon debugging tool. Using the emulator the design could be emulated at 500kHz which is about 80 thousand times faster than non-accelerated simulation. This allows functional debug of the IC without the observability problems of debugging with actual silicon and with better turn around time than simulation.

Microcode development was done in the Synopsys VHDL System Simulator environment using standard Synopsys tools, meta/micro assembler, linker and command macros. Much of the microcode VHDL simulations were run with parts of the IC deactivated in order to speed up simulation and debug time. Because of the very short design development schedule, exhaustive testing of the microcode through simulations of all the many different datastreams and possible error conditions was not possible prior to release of the design to wafer processing. A Patch RAM module was designed into the IC to provide silicon based microcode debugging and modifications to the microcode if needed.

The Patch RAM module is composed of 2 breakpoint registers, a control unit and contains enough RAM for 64 instructions. Two 32 word patch segments may be applied to the Patch RAM along with their corresponding breakpoints. When the program counter matches the value of either breakpoint register then the internal microcode ROM is disabled and instructions are then fetched from the corresponding patch segment. Program control can jump back to the microcode ROM at any time. There are no differences between instructions in the Patch RAM and instructions in the microcode ROM.

The Patch RAM is accessed through a single register address and the instructions are loaded via the host bus interface. The Patch RAM is a module that will be removed from future cost reduced implementations.

5 Conclusion

The design methodology chosen proved to be successful. First pass functional devices were achieved with a very short design schedule that was complicated by changing specifications and requirements. The design methodology is also showing its value in the reliable manner in which cost reduced versions and derivative versions of the device can be produced.

The architectural decision of having a hardwired arithmetic unit separate form the programmable processor has also proved to be a good decision. The performance of the device is primarily determined by the specific implementations of algorithms and the datapath widths in the Arithmetic Unit. Characterization results show that the noise level is −101.1dB at 16 bits and −113.1dB at 18 bits when tested with a swept sine wave, 10Hz − 20kHz, 1 volt full scale −20dB. Figures 6 and 7 show the THD and Amplitude responses versus frequency. Figure 8 shows the temporal distortion of a 11kHz tone burst. The decoder IC qualifies as a high precision MPEG audio decoder.

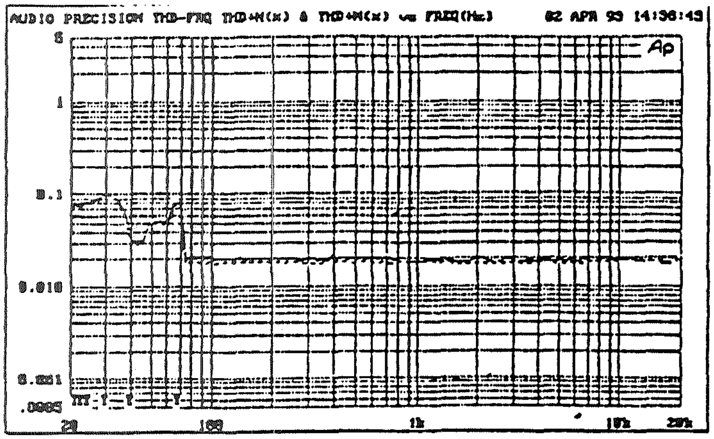


Figure 6: Total Harmonic Distortion of AV110

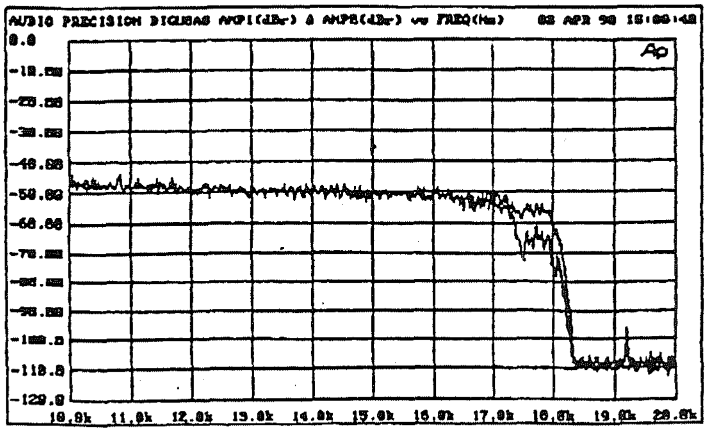
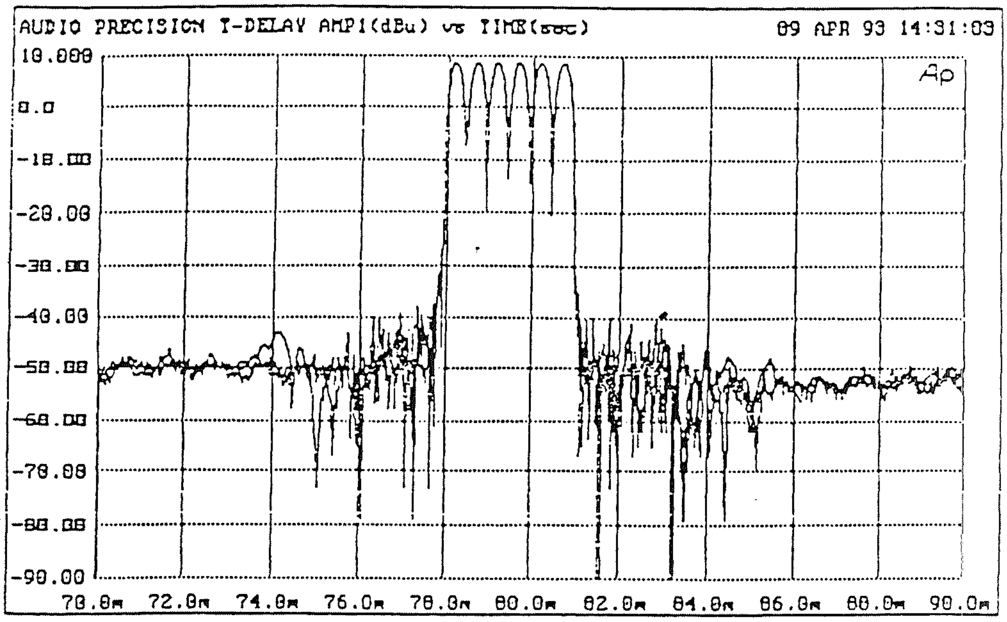


Figure 7: Frequency Response of AV110



coder processing delay measured using tone burst envelope
one channel per trace

Figure 8: Temporal Distortion of 1kHz Tone Burst

6 Acknowledgements

We would like to express our gratitude to Dr. Peter Schreiner of Scientific Atlanta at Atlanta for the frequency domain data and extensive testing of the device. We would also like to thank Doug Lankford of Thomson Consumer Electronics for his contributions to the development and evaluation of the prototypes.

References

[1] ISO Standard "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5MBit/s," CD 11172-3, July, 1992.

[2] E. Zwicker, U. T. Zwicker, "Audio Engineering and Psychoacoustics: Matching Signals to the Final Receiver, the Human Auditory System," J. Audio Eng. Soc., Vol. 39, No. 3, March, 1991.

Session 4
Featured Presentations II

Chairman: Nasir Ahmed

Correct CMOS IC Defect Models for Quality Testing

Jerry M. Soden
Electronic Quality & Reliability
Center
Failure Analysis Dept. 2275
Sandia National Laboratories
Albuquerque NM 87185
505 - 845 - 8575

Charles F. Hawkins
EECE Department
The University of New Mexico
Albuquerque NM 87131
505 - 844 - 7187

Abstract - **Leading edge, high reliability and low escape CMOS IC test practices have now virtually removed the stuck-at fault model and replaced it with more defect-orientated models. Quiescent power supply current testing (I_{DDQ}) combined with strategic use of high speed test patterns is the recommended approach to zero defect and high relaibility testing goals. This paper reviews the reasons for the change in CMOS IC test practices and outlines an improved CMOS IC test methodology.**

1 Introduction

Quality test practices for high reliability and even commercial CMOS ICs have changed dramatically in the past ten years [1,2]. Conscientious design-for-test (DFT) practices using scan designs and near 100% stuck-at fault (SAF) testing in the 1980's have been followed by reports of unacceptably high defect levels at the board level of assembly and for delivered products [3-7]. Careful studies on the nature of CMOS IC defect electrical behavior and test evaluations by several different manufacturers support the conclusion that the SAF model is inherently inadequate for CMOS ICs [3-12]. In some cases, scan-based SAF test practices have led manufacturers to higher, not lower, defect levels when compared to traditional at-speed functional testing. We discuss why this occurs and what future direction is necessary regarding defect models.

A joint Sandia National Laboratories and University of New Mexico team has taken data over the past ten years showing that nearly 95% of CMOS IC defects cause significant elevation of the quiescent power supply current (I_{DDQ}) of the device under test (DUT). In addition, many of these same types of defects only weaken the signal voltage levels, so defective ICs may still appear to function in the test environment, but fail later in customer use. Quiescent power supply current monitoring (I_{DDQ} testing) is the single most powerful technique to reduce defects in CMOS ICs [1,13]. I_{DDQ} testing measures the magnitude of the power supply current during the quiescent period of certain test vectors, with defect detection levels set by a pass/fail limit for the magnitude of I_{DDQ}.

Recent data from NCR, Storagetek, Hewlett Packard, Mitsubishi, VLSI Technology, AT&T and National Semiconductor show that I_{DDQ} testing may reduce defect levels in the range of 10-100 times below that achieved when SAF or functional testing alone was used [3-9]. In addition, studies by Ford Microelectronics and Sandia Labs show that a significant reliability risk exists when using ICs that function but have elevated I_{DDQ} values [14-16]. These conclusive data from independent sources provide a solid foundation for new

CMOS IC test methodologies. Realistic defect models exist for CMOS ICs and a test plan based upon these models is presented in this paper.

We discuss the following concerns.

- Why has SAF testing led to high manufacturing defect levels?
- Voltage testing and current testing - Why two?
- Defects & Faults - What is the Difference?
- What defect models are correct for the electronic properties of CMOS ICs?
- A CMOS IC high-rel, low escape test plan

2 Why has SAF testing led to high manufacturing defect levels?

The SAF model fails as a primary test metric because it does not accurately represent the properties of the dominant CMOS IC defects. A SAF describes defect behavior that permanently holds a circuit node at a logic "0" or "1" (V_{SS} or V_{DD}). A SAF is not sensitive to clock frequency, power supply voltage, temperature, or anything else. The majority of real CMOS defects do not behave like this and, as a consequence, SAF test sets allow a high test escape level for common defects. If this is true, why did SAF testing become a dominant test?

We must examine the roots of the SAF model to understand why it became a dominant, but flawed, concept for testing CMOS ICs. The SAF concept originated in the short, creative paper by Richard Eldred in 1959 [17]. Computer companies, particularly IBM and AT&T, further developed the idea in the 1960-70 era, defining such SAF concepts as fault coverage measurement, automatic test pattern generation (ATPG), controllability/observability, fault equivalence, and testability analysis [18,19]. The prevalent technologies of that era were bipolar RTL, DTL, TTL, and ECL. No reports are found that describe the effectiveness of the SAF model in achieving low defect levels. One study by an IBM failure analysis group reported that TTL failures caused either SAF failure modes or delay errors not detectable by SAF test patterns [20].

The growing acceptance of the SAF model in the 1980's can be attributed to (1) the stature of the computer companies who promoted it, (2) the easily grasped concepts of total controllability and observability enabled by circuit scan techniques, and (3) the relative ease with which test vectors could be generated by computers using Boolean algebra algorithms. It was easy to accept the concept that 100% node voltage control and observation would solve the difficult test problems of the LSI era.

It is important to note that in the 1980's, many of the companies espousing 100% SAF testing did not achieve it for all of their products. Certain divisions of a company might practice SAF testing while others might use functional vectors. We are not aware of data published at that time by any company comparing the effectiveness of SAF testing versus functional testing in terms of real board or fielded product defect levels. The SAF model became the entrenched test methodology in the bipolar era. When nMOS, and then CMOS, technologies replaced bipolar as mainstream IC technologies, the established SAF test practices were *assumed* to produce low defect levels. A stuck-at fault grading experiment performed by Motorola in 1983 on 22,506 nMOS ICs showed that the failure percentage increased as the SAF coverage increased [21,22]. This has sometimes been cited as supporting SAF effectiveness in achieving low defect levels for MOS IC testing. In fact, this is not so. An erroneous assumption is that the 99.8% SAF coverage of the experiment produced near zero defects. The experiment did not measure detection of all defects but instead measured only those ICs detected as failing the SAF test set for a particular test clock frequency. Bridges, stuck-opens, delay and other defect coverage were not measured and a population of these defects would still exist in the sample of ICs that had passed the high SAF coverage test. William Bruce, a co-author of that work, concurred and also stated that the experiment was not to show zero defects, but to test a prevailing assumption that failing ICs would be detected early in the test set (or "When ICs fail, do they do it catastrophically?") [23]. As for CMOS ICs, we are also not aware of any comparative studies during the 1980's that examined the relation between SAF testing and defect levels measured from assembly and field failure rates.

Despite these observations, SAF model test practices continue today with considerable effort often expended to achieve 100% SAF coverage. Additionally, SAF and yield information are used to predict manufacturing defect levels. It is worth commenting on why this latter practice does not predict defect level and is wasteful.

If a single defect spectrum is assumed, then a relation between defect level (DL), yield (Y), and defect test coverage (T) is [19]

$$DL = 1 - Y^{(1-T)} \tag{1}$$

The family of curves for a range of Y are plotted in Figure 1. These curves are used by some to predict defect level from SAF test coverage and supplier yield information. Practitioners use SAF coverage (not defect coverage) as the measure for T.

In practice, the $DL = 1 - Y^{1-T)}$ model cannot be verified from real manufacturing measurements and is therefore virtually worthless. Our observations regarding the use of the CMOS SAF model for (1) include the following.

1. SAF model testing assumes a pure spectrum of defects that cause true, static, SAF behavior, but real world defects causing SAF behavior are in the minority. Therefore, a mismatch exists in the model/defect relation. You don't know what defect types you have detected when you measure your tester failures and, significantly, *the large fraction of nontarget defects detected by the SAF vectors are not accounted for in the "T" of the equation.* Defects producing SAF behavior may represent 5-20% of all defects, therefore what good is the "T" value if SAFs are so poorly represented in the real CMOS world? You don't know what you're predicting, so why bother. Huisman made similar observations when he reviewed Hewlett Packard data [24].

   Sandia Labs failure analysis has found that only signal nodes bridged to power supply nodes and a certain form of logic gate open circuit defect are capable of true SAF behavior. These two defect types are typically in the minority so that SAF tests detect

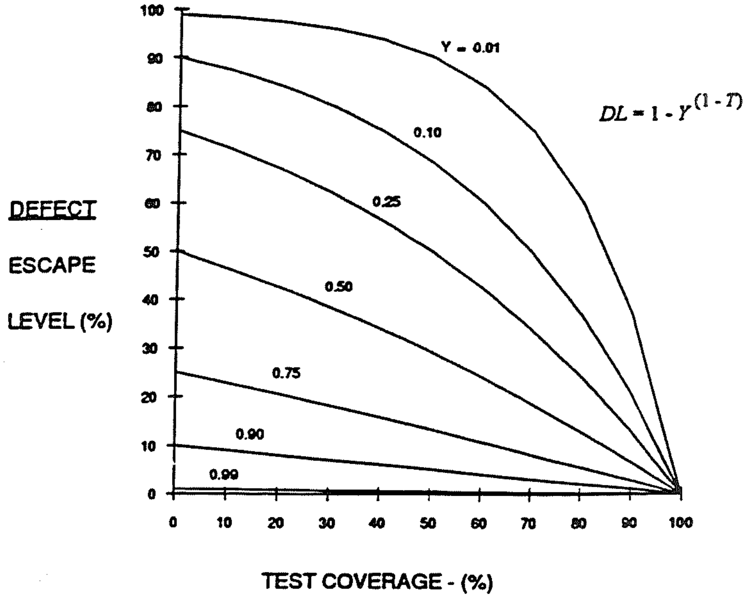


Figure 1: Plot of DL versus test coverage for a range of yield (Y).

proportionately more nontarget defect (fault) types than defects causing SAFs. By chance, SAF test sets pick up nontarget defects including some bridges, some stuck-opens, and some delay faults.

2. The confusing mix of target and nontarget defects described above is one source of SAF model error; another is listed by Huisman [24] - the clustering assumption. Theoreticians have added a constant n_o SAF faults per die, to model equation (1) to account for inaccuracies in matching (1) to test data [25]. The constant n_o is conceptually related to defect clustering, but, as applied to (1), it is related to either SAF equivalence or SAF clustering, a nebulous concept for CMOS ICs. Defect clustering does make test detection more efficient, but real die with small percentages of SAF type defects can't be made to fit a defect level model expression simply by assuming that SAF equivalence accounts for differences. While many have strong faith in this image of the n_o parameter, we don't, preferring to think of it as a fudge factor that forces SAF model test data to fit equation (1). If defects causing SAFs are basically not present in real ICs, why would we assume that the SAF equivalence concept is trustworthy?

3. Another IC test measurement situation shows the difficulty of using the $DL = 1 - Y^{(1-T)}$ model with or without n_o. Tester clock frequency has strong influence on real defect levels since many CMOS defects are clock frequency sensitive. One might simply halve the clock frequency for the SAF test set and see an apparent, but misleading, increase in the yield. The DL versus SAF coverage will then acquire a range of curves as a function of clock frequency. So, while customers may demand a yield value (Y) from a supplier, their results are significantly distorted by the application clock frequency.

4. Defect density and die size affect the parts per million (PPM) defect level. Gayle presented manufacturing data showing that the defect level rose for the larger die sizes within a small range of SAF coverage [4]. Intuitively, these data show that, for a given

defect density, a larger die at a given test coverage will expose more untested sites than a smaller die. This is further evidence that SAF testing does not model the real manufacturing world.

The $DL = 1 - Y^{(1-T)}$ model prediction for CMOS ICs is not much better than intuition. The model states that if your fab has defects and every crevice of circuitry isn't examined, then a probability exists that an escape will occur. The model quantifies this for a "pure" defect (fault) spectrum. Simplicity has been its greatest selling point. The message that we must examine every defect possibility is a contribution to test quality, but to use the model in real CMOS environments, test coverage must be quantified as defect coverage not fault coverage and that is very difficult to do. *There is no one magic fault (defect) model that detects all defects - zero defect testing requires several defect model tests* [5,26].

The challenges are how to adapt to new testing approaches and how to replace the SAF culture in the face of strong acceptance by many (but not all) manufacturers. We will discuss the technical issues, leaving cultural change to each manufacturer. Recent data comparing SAF and current monitoring (I_{DDQ}) on board level defect levels indicate that SAF model testing will rapidly be replaced by other methods [3-9].

3 Voltage testing and current testing - Why two?

The established criteria for logic test sensing is voltage measurement of primary outputs. Voltage measurement determines the Boolean functionality used with SAF, at-speed, or more general functional vectors. Current sensing uses power supply (V_{DD} or V_{SS}) currents as a pass/fail criteria. There are two methods for using power supply current used for defect detection; I_{DDQ} and i_{DDT} measurements. I_{DDQ} testing has been used since the mid-1970's and its use has grown rapidly since 1990 [1]. Transient power supply current (i_{DDT}) measurement is a promising test for defects, but it has not been evaluated for production testing [27,28].

Current sensing, especially I_{DDQ}, is the most sensitive test for the majority of CMOS defects. I_{DDQ} testing should be used as early as possible in the test process such as at wafer probe (die sort), at the package level to detect EOS/ESD failures, and at the post burn-in level. There are a few types of defects that I_{DDQ} testing does not detect. These include certain types of open circuit defects, subtle changes in contact integrity or interconnect material, and slight shifts in transistor parameters. Their effects, if any, on functionality may be dependent on frequency, V_{DD} and temperature. Voltage testing should be used to detect those types of defects that cause delay but don't elevate I_{DDQ}. Delay defect coverage is difficult to quantify and tests usually are called "at-speed testing" or "delay fault (defect) testing" [29]. *Both* current and voltage sensing tests are required as complementary tools that match particular CMOS defects and their responses.

SAF testing is inefficient and should only be considered for certain low occurrence, open circuit defects. However, SAF testing may not be used at all if there is confidence in the at-speed or delay defect test sets. SAFs are a subset of delay faults since SAFs can be interpreted as faults that cause infinite rise or fall time.

4 Defects & Faults - What Is the Difference?

A defect is a physical imperfection in the circuit caused by incorrect design, fabrication, handling, or a combination of these and other manufacturing processes. A defect model is a behavioral description of a specific defect or class of defects it models real properties. Defect models are required for developing accurate automatic test pattern generation. A fault, on the other hand, is a *hypothesis* of general behavior pattern that a defective circuit will exhibit; a fault is not real. It is more constructive to ask: what is your defect model; it removes the vagueness associated with most faults. We emphasize the terms *defect model* and *defect detection*, (rather than fault model and fault detection).

5 What defect models are correct for the electronic properties of CMOS ICs?

This section summarizes data taken primarily at Sandia Labs on CMOS electronic defect properties and then separates these properties into behavioral groups. Defects can be categorized into shorts, opens, and certain parametric shifts that typically cause delay. Defect models are defined here by their properties that allow test detection. A priority list of defect properties for CMOS IC testing would take the following categories into account

1. Bridging - Both resistive and zero ohm
   (a) Transistor Level
   (b) Logic Gate Level
   (c) Power Buses
2. Opens - Both narrow and wide open defects
   (a) Transistor Drain/Source
   (b) Transistor Gate (Floating Gate)
   (c) Transistor-to-Transistor node
   (d) Logic Gate
   (e) Transmission gate opens
   (f) Power Buses
3. Parametric Delay - $\Delta V_T, R_{via}, \Delta Tox, \Delta(W/L), ...$

5.1 Bridging Defects:

The effects of CMOS bridging defects on electrical behavior are dependent on where they are located in the circuitry. Bridging defect locations include

1. transistor level,

2. logic gate signal level, or power bus V_{DD} to V_{SS}. Bridging can occur in combinational or sequential logic elements, producing different effects. Transistor level bridges between gate (G), drain (D), source (S), and bulk (B) include gate oxide shorts (GB, GD, GS), punchthrough (DS), and leaky *pn* junctions (DB, SB). Intertransistor node defect bridges also occur. All shorts or bridging defects have an associated resistance whose value impacts the electrical response. Logic level bridges occur between logic gate signal nodes or between signal nodes and power buses. These bridges have a similarity to transistor bridges in that signal node voltages are weakened as the bridge resistance decreases. Bridges occurring between power buses include metal-metal shorts and soft *pn* junctions between wells to substrate. Combinational logic and power bus defect bridges are 100% detectable with I_{DDQ} testing while sequential logic (flip flops) may require both current and voltage sensing for full detection of all bridge defect possibilities [30,31].

Figure 2(a) shows a 2NAND gate buffered by inverters in which transfer curves for a transistor gate-drain bridging defect were simulated over a range of resistance from 500 Ω to $2k\Omega$. Figure 2(b) shows these curves with functionality achieved for defect resistance above $1k\Omega$. Figure 3 shows logic gate bridge node voltages and functionality above bridge resistance values measured on test circuits for different logic inputs. In both cases functionality is maintained for defect resistance above a certain value (e.g., $1k\Omega$ for transistor bridge and 150Ω for the logic defect bridges). This characteristic is consistent for transistor level and logic gate bridges. Above some defect resistance value, functionality is maintained. Sandia simulation of a 2 micron standard cell family (CMOSN) found the upper resistance value for functionality ranged from 150Ω to less than $2k\Omega$ for a variety of standard cell configurations. Figures reported metal bridge resistance in a test chip and found that 69% of the defect resistance clustered below 500Ω, 26% clustered around 500Ω, and 2.8% of the defects had resistance between 500Ω and $19.1k\Omega$ [32]. The table in Figure 3 shows the significant I_{DDQ} elevation associated with bridge defects.

Certain bridge defects in sequential circuits such as master-slave flip flops, may have transistor drive strength contentions whose results are modified by the resistance of the bridge. It has been shown that some low resistance bridge defects in sequential circuits are not detectable by I_{DDQ} and they therefore require both I_{DDQ} and voltage sensing [30,31]. Many transistor bridge defects, such as gate oxide shorts, punchthrough, and drain-bulk or source-bulk connections, have nonlinear I-V characteristics. These defects can weaken logic drive voltages, causing delay and elevation of I_{DDQ}.

We conclude that CMOS bridging defect models must account for these variations of defect resistance. Voltage-based bridge testing with Boolean vectors uses a zero ohm defect assumption and therefore is inadequate for CMOS testing. A correct CMOS bridge defect model includes the quiescent current (I_{DDQ}) sensitivity property for detection.

5.2 Open circuit defects

Open circuit defects are unintentional electrical discontinuities in the circuit that can cause diverse and, in some cases, unpredictable electronic behavior. These defects include open

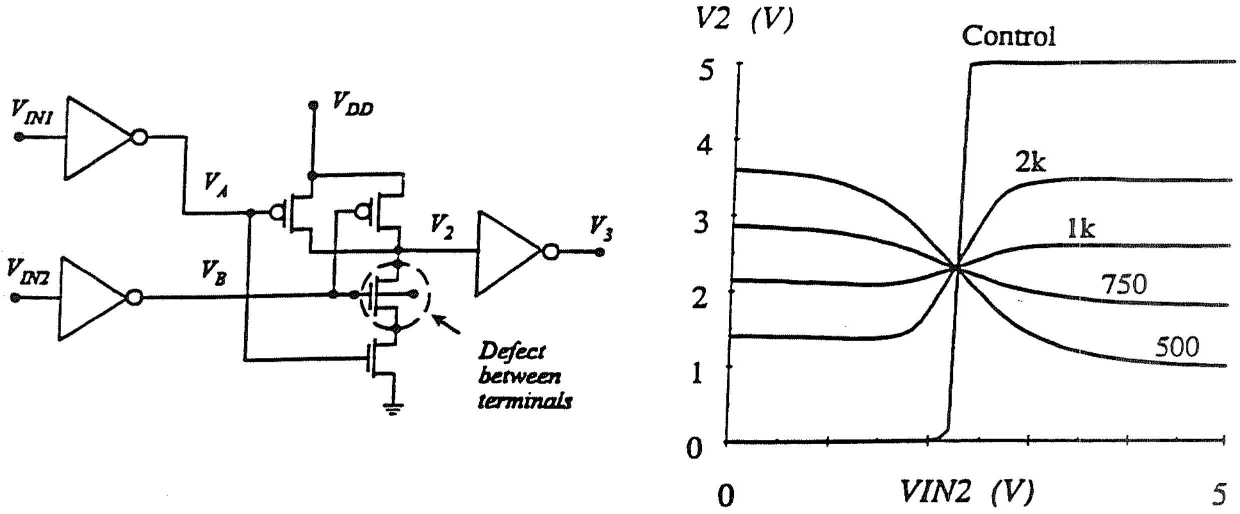
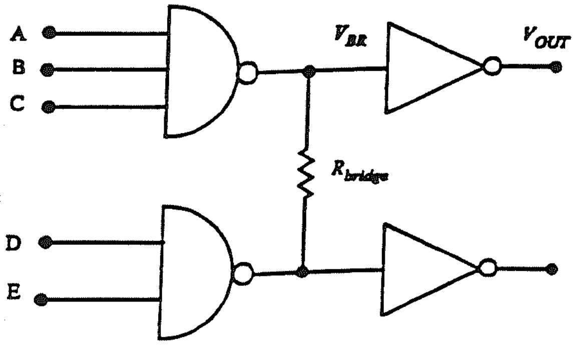


Figure 2: (a) 2NAND with gate-drain ohmic bridging defect.
(b) Transfer curve for several bridge defect resistances



ABC and *DE* Inputs					$R_{bridge} = 0\ \Omega$			$R_{bridge} > 0\ \Omega$
A	B	C	D	E	VBR	VOUT	IDDQ	Functional
(V)	(V)	(V)	(V)	(V)	(V)	(V)	(mA)	for R_{bridge} >
0	0	0	5	5	3.95	0.01	2.74	950 Ω
0	0	5	5	5	3.36	0.12	3.07	725 Ω
0	5	5	5	5	2.03	4.02	3.58	150 Ω
5	5	5	0	5	3.38	0.12	2.02	1250 Ω
5	5	5	0	0	4.24	0	1.64	1750 Ω

Figure 3: Logic gate defect bridge affects on functionality

contacts (missing via metal or unetched via oxide), metallization opens (patterning, improper etching, stress voiding, or electromigration), opens in diffusion or polysilicon (mask or fabrication errors), or power bus opens.

This section summarizes data from a range of open circuit defect structures and defines five open circuit defect models to account for all open circuit electronic failures. Data taken from Sandia production and test ICs show that each defect model exists in real silicon circuits and that defect properties are consistent with each defect model. Open circuit defects cause different effects that depend primarily upon circuit defect location, defect size, local electrical structure, and process variables.

Figures 4(a,b) show one effect of *defect location.* Figure 4(a) illustrates a large open circuit in an n-channel transistor gate. Large open circuit defects in a transistor gate interconnection cause the polysilicon structure to electrically float and acquire a static bias voltage that is a function of the surrounding electrical structure. The logic gate output voltage is then affected by the signal drive to the complementary p-channel transistor and by the bias state of the defective transistor. An open defect circuit location is also important if the defect lies in the transistor gate, drain, or source interconnect path.

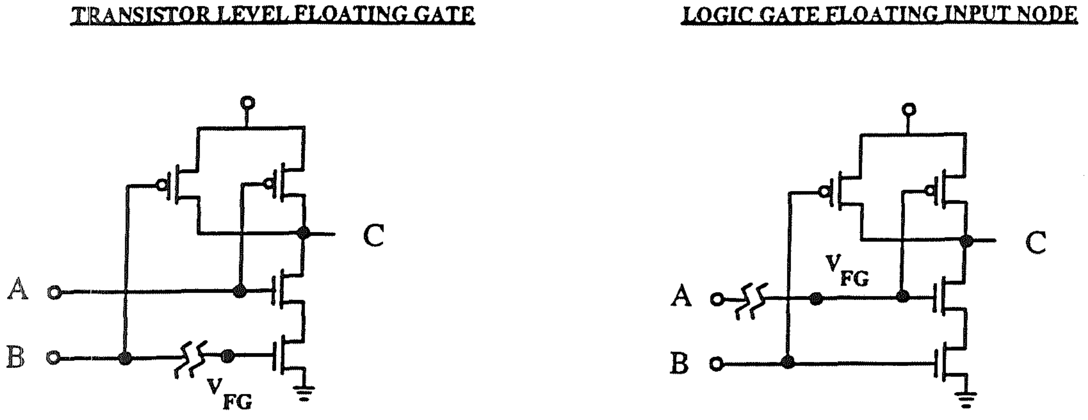

TRANSISTOR LEVEL FLOATING GATE
A
B
C
V_FG
LOGIC GATE FLOATING INPUT NODE
A
B
C
V_FG

Figure 4: (a) Open gate defect to a single transistor (b) Open gate defect to a logic gate.

Figure 4(b) shows a large logic gate defect that affects both complementary transistors. This defect location (and size) causes different responses than an open defect to a single transistor. A large open defect in Figure 4(b) allows the input node to float and the floating node voltage may put both transistors in a conduction state or it can force one transistor on and the other off.

Figures 5(a,b) illustrate the effects of *defect size.* Narrow open circuit defects can have signal coupling across the "open" regions by a tunneling mechanism [33]. The electrical responses of large and small open defects are quite different. Figure 5(a) shows a small open defect on a logic gate in which tunneling current (J) occurs across the narrow (< 1000 Å) defect cleft. This causes a slower transfer of charge that increases the rise and fall times of the input node. Figure 5(b) illustrates a large open defect in which the logic gate input node is decoupled from the signal .

However, the floating gate node acquires a DC bias voltage that is function of the *local electrical structure*, which affects the response of the defective circuit. Floating node voltages

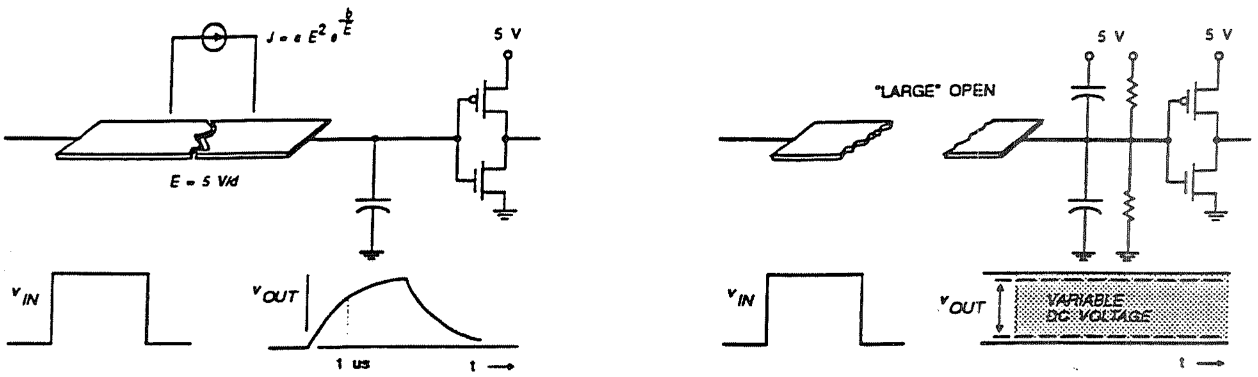


Figure 5: (a) Small open defect in logic gate input (b) Large open defect in logic gate input

have been observed in failure analysis to drift to either power supply voltage or the node may reach an intermediate voltage level that can turn -on both *p*- and *n*-channel transistors.

The effects of *process variables* on open circuit defects are not well understood. Identical mask-induced open defects were compared from two different test die fabrication sites and processes. In some situations, the electronic effects were significantly different. We emphasize that, while the response of certain open circuit defects is not always predictable, the possible responses are bounded. Therefore, test strategies for open circuit defects can be defined that account for all possibilities. The five open circuit defect models defined later in this report bound the behavior of CMOS open defects.

Summary of Open Circuit Defect Properties: Table I summarizes the diverse electronic properties of CMOS open circuit defects described above as: (1) large or small defect, (2) logic gate versus transistor defect, (3) defect in transistor drain, source, or gate line, and (4) local electrical structure and its affect on steady state floating node voltage. Table I shows defect behavior and the sensitivity of this behavior to three test methods: I_{DDQ}, i_{DDT}, and Boolean. Code number refers to situations in which the severity of the defect or local electrical topology affect detection. For example, a small, defect-induced degradation in rise or fall time may not initially be detectable, but if the degradation increases with time, then that defect may be detectable.

Open Circuit Defect Models: II reduces the properties described in Table I to a list of five defect models. Each of the five models is behaviorally unique and requires a separate test strategy for defect detection. Defect models #2 and #3 describe two behavior patterns for large opens in a single transistor. Local electrical structure and process variables allow (and data show) two possible defect model outcomes. The last two columns indicate the best test methods to achieve defect detection. It is presently difficult to attain 100% detection for open defect models #1 and #2.

Parametric Delay Defects: Certain defects cause signal delay without elevation of I_{DDQ}. These defects are not classified as opens or shorts. The electrical effects of "parametric defects" are often sensitive to temperature, operating frequency, and power supply voltage. Boolean test methods are difficult to implement since there are many defect sites to examine, the timing effect on voltage levels can be small, and a 2-vector pair is often required for

Defect	**Defect Behavior**		**Test Method**		
	Node		I_{DDQ}	i_{DDT}	Boolean
Small Open			*Code	*Code	*Code
Transistor	G,D,S	Increased rise or fall times; delayed i_{DDT}	2	2	2
Logic Gate	Input	Increased rise of fall times; delayed i_{DDT}	2	2	2
Large Open					
Transisitor	G	Floating node; transistor-off	2	x	1
	G	Floating node; transistor-on	1	x	2
	D,S	Transistor-off	2	x	1
Logic Gate	Input	Floating node; input node acquires a voltage that turns on transistor pairs	1	x	2
	Input	Floating node; input node acquires a voltage whose value lies at less than threshold for one of the transistors	3	x	1

*Code:

1. Detection assured for proper vector application.
2. Detection only for certain conditions.
3. Detection is not accomplished.
4. x (Data not known).

I_{DDQ} - Refers to measurement of quiescent power supply current.

i_{DDT} - Refers to measurement of transient power supply current.

Boolean - Refers to measurement of a functional voltage at a primary output node; this can be a large empirically derived test, a delay defect test, a stuck-at fault test set, or an at-speed upgraded vector test set.

Table 1: Open circuit defect properties

Defect Model	Defect Model Behavior	Defect Type	Test Method	100% Test
1	Delay - Increased rise time, fall time	Small opens at any transistor node, Small opens at any input to logic gate	Boolean i_{DDT} I_{DDQ}	No
2	Transistor-off – Floating transistor node (memory effect)	Large opens at transistor gate, drain and source nodes	I_{DDQ}, i_{DDT}, Boolean	No
3	Transistor on – Floating transistor node	Large opens at transistor gate node	I_{DDQ}, Boolean	Yes No
4	Transistor pair on – Floating logic gate input node larger than both transistor threshold voltages	Large opens to logic input gate terminal	I_{DDQ}, Boolean	Yes No
5	Transistor pair-off – Floating logic gate input node less than either transistor threshold voltages	Large opens to logic input gate terminal	Boolean I_{DDQ}	Yes No

Table 2: Open circuit defect models.

defect detection. These defects, such as a change in transistor threshold (ΔV_T), increase in via resistance (R_{via}), local change in thin oxide thickness (ΔTox), or nicks in the transistor geometry that alter width to length ratio ($\Delta(W/L)$) remain difficult defects to detect. They are an important class of defects to pursue since test escapes and reliability failures are probable. Thinned transistor oxide regions can lead to increased local charge tunneling, buildup of fixed charge, and subsequent gate oxide rupture. A defect that locally reduces the transistor length can increase electric field intensity and may initiate short channel carrier injection into the thin oxide.

Much information has been gathered and analyzed over the past ten years on the electronic properties of CMOS defects. This allows improved test methodologies. We know which defects we can quantify and those that we can't. Bridge defects are a common IC o occurrence and are highly detectable if the circuit is a true CMOS design with a fully complementary, fully static implementation. I_{DDQ} is an efficient, relatively short test method that guarantees 100% detection of bridge defects for all combinational logic and must be used with voltage testing for 100% detection of bridges in sequential circuits. I_{DDQ} has major advantages over Boolean bridge test methods because the Boolean technique is weak for detecting resistive bridge defects, but both are required to detect all bridge possibilities in sequential circuits [30,31].

Open circuit defects are more complex in their electronic behavior. The detection of certain open defects can be quantified, such as the transistor-on and transistor pair-on defect models. The transistor-off defect can cause combinational logic to behave as sequential logic in CMOS ICs (the CMOS "memory" effect). In theory, it can be detected with 2-vector sequences of Boolean vectors. In practice, this has not been achieved for large ICs because of numerical intractability. I_{DDQ} testing is useful for transistor-off defects, but is a nontarget activity. I_{DDQ} in practice has detected transistor-off defects when design conditions encourage high-Z states to turn on load transistor pairs or an input node floats to a voltage that causes load transistors to turn on. Transistor pair- off defects are 100combinational logic transistor pair-off defects if the transistor in the series pullup or pulldown path floats to the on-state and will not detect the defect if that series transistor is off. I_{DDQ} testing will not detect transistor pair-off defects in inverters. Delay defects caused by opens are difficult to detect. Boolean methods become intractable for the 2-vector requirement and I_{DDQ} tests rely on chance timing properties that elevate I_{DDQ} at the moment of measurement. i_{DDT} testing remains a challenge to be proven for delay defect test applications. The detection of parametric delay defects also cannot be quantified since the possible defect sites are quite large and the signal delay properties are difficult to detect.

CMOS ICs require both current and voltage test patterns for high defect coverage. We offer the following test method sequence for CMOS ICs designed for fully static, fully complementary performance. This recommended procedure can be altered to optimize test strategies for wafer probe versus packaged parts. Test vector generation requirements are not described.

A CMOS IC test strategy:

1. Perform I_{DDQ} testing using vectors that detect 100% of the transistor and logic gate bridges. This also detects 100% of the open defects that cause a transistor-on or transistor pair-on defect model behavior. Nontarget defects that are not quantifiable will also be detected by I_{DDQ} testing. These nontarget defects include transistor-off (memory) defects, transistor pair-off defects, open circuit delay defects, and parametric defects.

2. Perform Boolean testing for delay defects. These tests may include vectors generated from delay defect modeling or use a large functional test set applied at-speed. Delay defects not detected by the I_{DDQ} and Boolean tests are difficult to quantify for large ICs, but these approaches represent the best methods that we have available.

3. Run a powerup i_{DDT} test. Although this test is not quantifiable, it is sensitive to open and certain parametric defects.

These tests are intended to drive CMOS IC escape rates to the 10-100 PPM level. Further reduction in escape rates should be directed at parametric and open circuit delay defects. While test escape rates may be driven to low defect levels, manufacturers must continue to focus on reliability and induced (handling) failures whose rates may now be higher than test escapes.

6 Summary

This paper reviews our present knowledge regarding CMOS IC defect models and outlines a test procedure that addresses the varied defect properties. We discuss why the traditional stuck-at fault model test procedure is inadequate for high quality CMOS ICs.

Acknowledgment

We thank Rich Anderson and Ed Cole of Sandia Labs for their valuable comments and review of this article. This work was performed at Sandia National Laboratories and supported by the U.S. Department of Energy under contract DE-AC-04-76DP00789.

References

[1] J. M. Soden, C. F. Hawkins, R.K. Gulati and W. Mao,“ I_{DDQ} Testing: A Review,” J. of Electronic Test: Theory and Applications (JETTA),vol. 3, no. 4, pp. 291-303, Dec. 1992.

[2] J. M. Soden, R.K. Gulati, and C.F. Hawkins, “Current Monitoring for Efficient Detection of CMOS Defects and Faults,” Tutorial (1-day) at 1993 Int. Test Conf., Baltimore, 1993.

[3] R. Gayle, “The Cost of quality: Reduce ASIC Defects with I_{DDQ} At-Speed Testing and Increased Fault Coverage,” Int. Test Conf., Session 14, Oct. 1993.

[4] R. J. Perry, “I_{DDQ} Testing in CMOS Digital ASICs,” J. of Electronic Test: Theory and Applications (JETTA), vol. 3, no. 4, pp. 317-325, Dec. 1992.

[5] P.M. Maxwell and R.C. Aitken, ”I_{DDQ}, Testing as a Component of a Test Suite: The Need for Several Fault Coverage Metrics,” J. of Electronic Test: Theory and Applications (JETTA), vol. 3, no. 4, pp. 305-316, Dec. 1992.

[6] K. Sawada and S. Kayano, “An Evaluation of I_{DDQ} versus Conventional Testing for CMOS Sea-Of-Gates ICs,” Int. Test Conf., pp. 158-167, Sept. 1992.

[7] P. Wiscombe, “A Comparison of Stuck-At Coverage and I_{DDQ}, Testing on Defect Levels,” Int. Test Conf., Session 14, Oct. 1993.

[8] G. Schiessler, C. Spivak, and S. Davidson, “IDDQ Test Results on a Digital CMOS ASIC,” IEEE Custom Integ. Cir. Conf., pp. 26.4.1-.4, 1993.

[9] T. Storey, W. Maly, J. Andrews, and M. Miske, “Stuck Fault and Current Testing Comparison Using CMOS Chip Set,” Proc. Int. Test Conf., pp. 311-318., Oct. 1991.

[10] R. R. Fritzemeier, C. F. Hawkins, and J. M. Soden, “CMOS IC Fault Models, Physical Defect Coverage, and I_{DDQ} Testing, Proc. Custom Integrated Circuits Conf. , pp. 13.1.1-13.1.8, May 1991.

[11] J. M. Soden and C.F. Hawkins, “The Relation of I_{DDQ} testing to Increased CMOS IC Reliability,” European Symp. on Reliability and Failure Analysis, Swabisch-Gmund, Germany, Oct. 1992.

[12] J. M. Soden and C. F. Hawkins, “Electrical Properties and Detection Methods for CMOS IC Defects,” Proc. 1st European Test Conf., pp. 159-167, April 1989.

[13] C. F. Hawkins, J. M. Soden, R. R. Fritzemeier, and L. K. Horning, "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," IEEE Trans. Industrial Electronics, vol. 36, No. 2, pp. 211-218, May 1989.

[14] S.D. McEuen, "Reliability Benefits of I_{DDQ}, J. of Electronic Test: Theory and Applications (JETTA), vol. 3, no. 4, pp. 327-335, Dec. 1992

[15] C.F. Hawkins and J.M. Soden, "Reliability and electrical properties of gate oxide shorts in CMOS ICs," Proc. Int. Test Conf., pp. 443-451, Sept. 1986.

[16] J.M. Soden and C.F. Hawkins, "Reliability of CMOS ICs with gate oxide shorts," Semiconductor Int., pp. 240-245, May, 1987.

[17] R. Eldred, "Test Routines Based on Symbolic Logical Statements," J. Assoc. for Computing Mach. (ACM), vol. 6, pp. 33-36, 1959.

[18] D. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets," IEEE Trans. Electron. Computers, vol. EC-15, no. 1, pp. 66-73, Feb. 1966.

[19] T. Williams and N. Brown, "Defect Level as a Function of Fault Coverage," IEEE Trans. Computers, pp. 987-988, 1981.

[20] C. Bey, K. Arya, C. Radke, and K. Torku, "Do Stuck Fault Models Reflect Manufacturing Defects?," Int. Test Conf., pp. 35-42, 1982.

[21] R.A. Harrison, R.W. Holtzwarth, P.R. Motz, R.G. Daniels, J.S. Thomas, and W.H. Wiemann, "Logic Fault Verification of LSI: How it Benefits the User," WESCON-1980, pp. 34.1 - 34.4, 1980.

[22] R. Daniels and W. Bruce, "Built-In Self-Test Trends in Motorola Microprocessors," IEEE Design and Test of Computers, pp. 64-71, April 1985.

[23] Personal Communication with William Bruce, Dec. 1992.

[24] L. Huisman, "Fault Coverage and Yield Predictions: Do we Need More Than 100% Coverage," European Test Conf., pp. 180-187, Rotterdam 1993.

[25] V.D. Agrawal, S.C. Seth, and P. Agrawal, "Fault Coverage Requirements in Production Testing of LSI Circuits", IEEE J. Solid State Ckt., SC-17, pp. 57-61, 1982.

[26] J. M. Soden, R. R. Fritzemeier, and C. F. Hawkins, "Zero Defects or Zero Stuck-At Faults - CMOS IC Process Improvement with I_{DDQ}," Proc. Int. Test Conf., pp. 255-256., Sept. 1990.

[27] J.F. Frenzel and P.N. Marinos, "A Comparison of Methods for Supply Current Analysis", Custom Integ. Ckt. Conf., pp. 13.3.1-13.3.4, May 1991.

[28] J. Beasley, H. Ramamurthy, J. Ramirez-Angulo, and M. DeYong, "i_{DD} Pulse Response Testing of Digital and Analog CMOS Circuits," Int. Test Conf., Oct. 1993.

[29] A.K. Pramanick and S.M. Reddy, "On Multiple Path Propagating Tests for Path Delay Faults," Int. Test Conf., pp. 393-402, Oct. 1991.

[30] R. Rodriquez-Montanes, E.M.J.G. Bruls, and J. Figures, "Current vs. Logic Testability of Bridges in Scan Chains," European Test Conf., pp. 392-396, Rotterdam, April 1993.

[31] Personal Communication with Manoj Sachdev, Philips Labs, August 1993.

[32] R. Rodriquez-Montanes, E.M.J.G. Bruls, and J. Figures, "Bridging Defect Resistance Measurements in a CMOS Process," Int. Test Conf., pp. 892-899, Oct. 1992.

[33] C.L. Henderson, J. M. Soden, and C. F. Hawkins, "The Behavior and Testing Implications of CMOS IC Logic Gate Open Circuits," Int. Test Conf., pp. 302-310, Oct. 1991.

Complementary Heterojunction FET Technology for Space Application

George LaRue
Boeing Defense & Space Group
P.O. Box 3999
Seattle WA 98124-2499

Abstract - **A 32-bit serial integer multiplier was designed to investigate the yield and performance of Complementary Heterojunction FET (CHFET) technology. This is the largest reported CHFET logic circuit. The maximum operating frequency was 500 MHz. Very low power dissipation of 3 mW was obtained at 5 MHz operation. Single-event upset (SEU) characteristics of CHFET devices and latches were also measured and indicates the potential for SEU hard circuits for space and military applications.**

1 Introduction

Complementary HFET (CHFET) technology offers the promise of very low-power high-speed circuits. The largest CHFET circuits demonstrated previously have been static random access memories [1][2]. This paper addresses the yield, performance [3] and single-event upset (SEU) [4] of CHFET logic circuits. Logic gate test structures were laid out to determine the base delays of various gates and their sensitivities to fan-out and capacitive loading. These include dynamic logic, source-coupled FET logic (SCFL), and direct-coupled FET logic (DCFL). Test structures were also designed to evaluate radiation effects of devices and latches. A 32-bit serial multiplier was designed to determine the performance and yield of an LSI logic circuit. The serial multiplier architecture was chosen to investigate highly synchronous circuits at high clock rates rather than to optimize multiplication time. A 32 by 32 bit multiplication requires 34 clock cycles. Dynamic power dissipation is higher than with a parallel design an d more accurate modeling of total power dissipation for highly synchronous circuits is possible.

The multiplier and test circuits were fabricated with Honeywell's C-HIGFET process with delta-doped pseudomorphic InGaAs channels on GaAs wafers. Typical transconductances were 320 mS/mm and 60 mS/mm for n-type and p-type HFETs respectively. Threshold voltages for the standard delta-doped wafers were nominally 0.55 V for the n-type and -0.55 V for the p-type HFETs. On three wafers, no delta-doping was used which raised the p-channel threshold voltage to -0.35 V. The n-HFET threshold was adjusted by implanting silicon with doses ranging from $2 \times 10^{12}\ cm^{-2}$ to $2.4 \times 10^{12}\ cm^{-2}$.

Two wafers are studied for comparison in this paper. Wafer A was not delta-doped and the n-HFET was implanted with Si at a dose of $2.2 \times 1012 cm^{-2}$. It has n and p threshold voltages of 0.14 V and -0.34 V respectively. The gate turn-on voltages for the n-type and p-type FETs are 1.70 V and -1.52 V. The standard process was used for wafer B which has n and p thresholds voltages of 0.53 V and -0.56 V and gate turn-on voltages of 1.68 V and -1.82 V respectively.

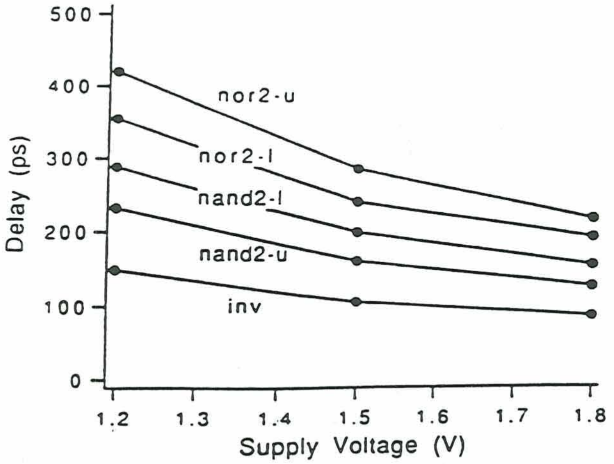


Figure 1: Gate delay vs. power supply for wafer A.

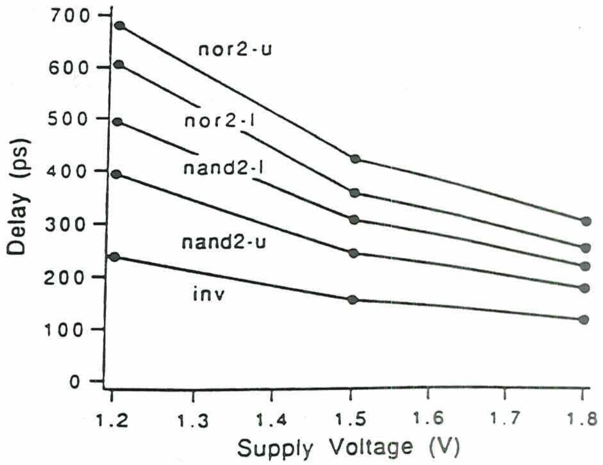


Figure 2: Gate delay vs. power supply for wafer B.

2 CHFET Performance

2.1 Static Complementary Logic

Several CHFET ring oscillators (ROs) were laid out and characterized to determine the performance of basic logic gates. Most of the ROs had seven stages. Ring oscillators with 23 stages were used in conjunction with 7 stage ROs to determine the static power dissipation and switching power of inverters. Inverter RO measurements were consistent with the results reported in [1]. The delays of inverters, 2-input NAND gates, and 2-input NOR gates are shown in Figures 1 and 2 as a function of supply voltage for wafers A and B. The width ratio of the p to the n channel devices was 2 for the inverters, 1 for the NAND gates and 4 for the NOR gates. The u and l denote which inputs of the NAND or NOR gates were used to form the ring, with nand2-u and nor2-l denoting a connection to the transistor with the source or drain connected to the output of the gate. The delays for wafer A, with lower threshold voltages, are lower and are not as sensitive to power supply voltage as those for wafer B. The effect of various loads on inverter delay is shown in Figures 3 and 4 for wafers A and B. Seven stage ring oscillators consisting of inverters driving a 100 fF load, 3 inverters, 3 NAND inputs, and 3 NOR inputs respectively were used for these measurements.

Delays for the 7 stage ROs were about 10% lower than the delays measured for corresponding 23 stage ROs. This systematic dependence of the gate delay on the number of RO stages should be taken into account when comparing these 7-stage RO results to those of competing technologies.

Figure 5 shows the delay and static power dissipation of inverters as a function of temperature for wafers A and B. The delay is fairly independent of temperature but the static power dissipation increases by a factor of 3 to 5.

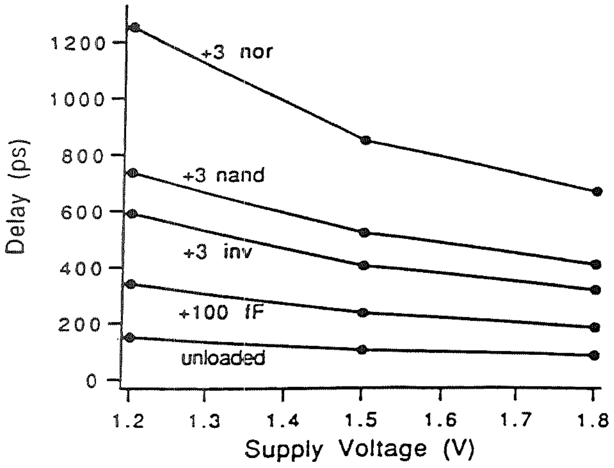


Figure 3: Loading effects on gate delay vs. power supply for wafer A.

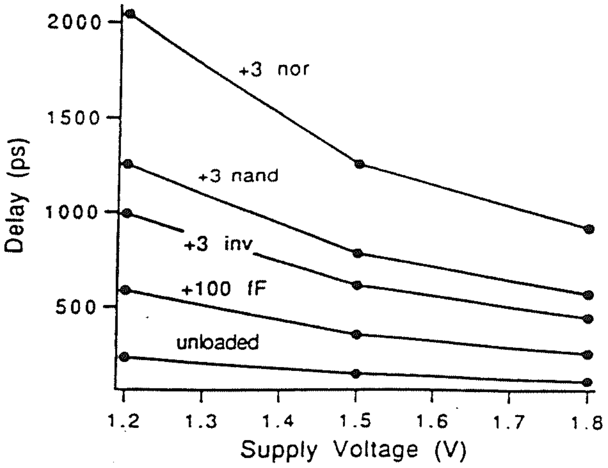


Figure 4: Loading effects on gate delay vs. power supply for wafer B.

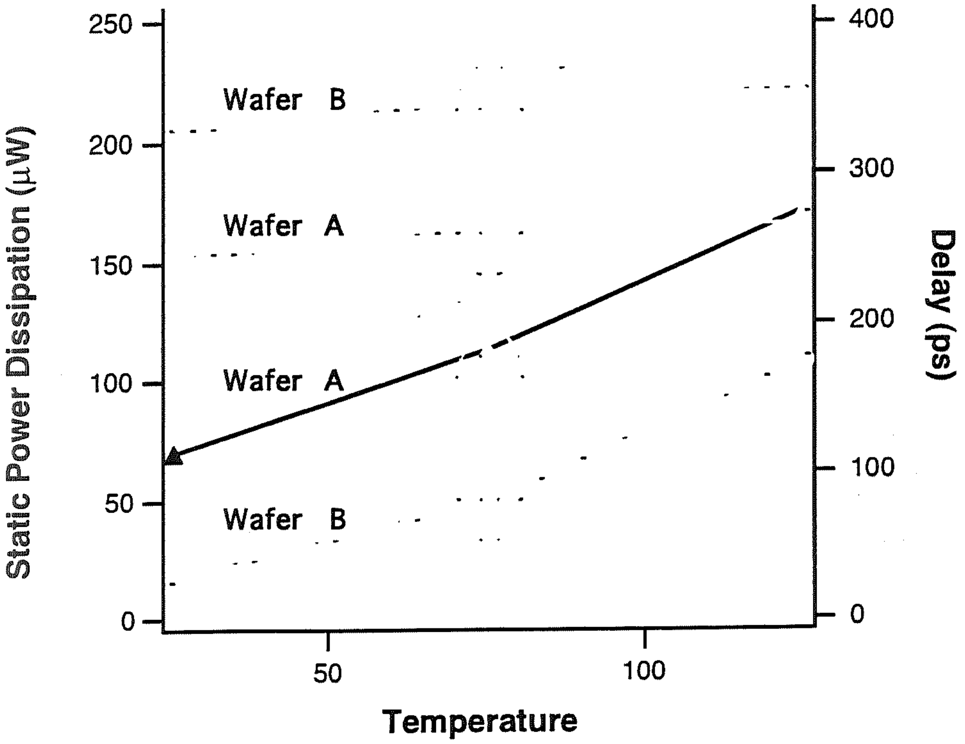


Figure 5: Delay and static power dissipation vs. temperature

2.2 SCFL, DCFL and Dynamic Logic

In addition to static complemetary logic, other types of logic were characterized. Figure 6 shows schematic diagrams for Direct Coupled FET Logic (DCFL) and Source Coupled FET Logic (SCFL) inverters. The delay versus power for DCFL, SCFL and static complementary logic are compared in Figure 7. Measurements on inverter ring oscillators and on loaded gates (3-input gate driving 3 other inputs and a 125 fF capacitor) are shown for each family. SCFL has much higher power but a lower loaded delay. DCFL and complementary logic have almost identical delay-power results. The higher static power dissipation of DCFL is offset by its lower dynamic power dissipation. Chips designed with DCFL will usually have higher power dissipation than those designed with complementary logic because most gate outputs do not toggle every cycle.

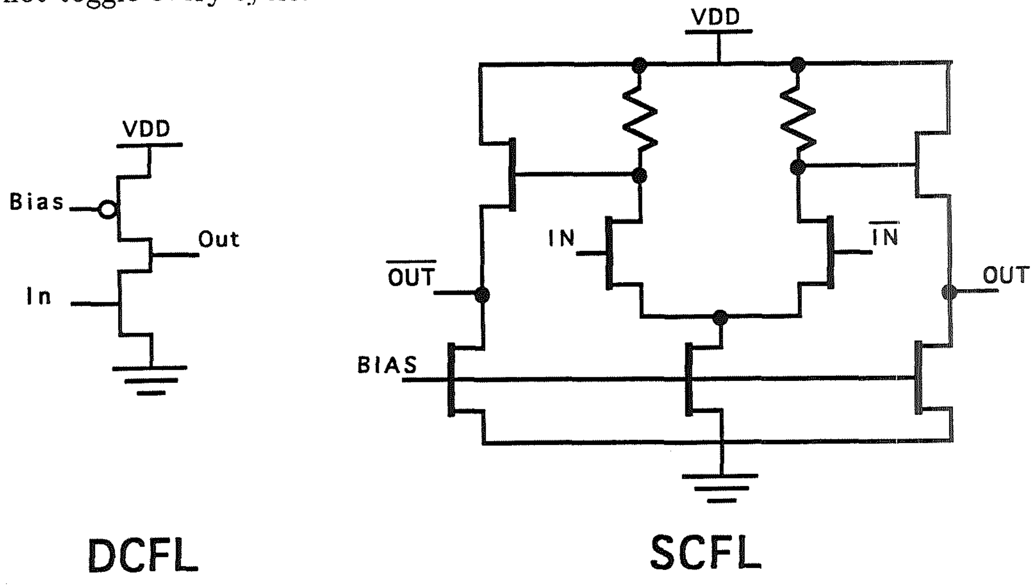


Figure 6: Schematic diagrams for DCFL and SCFL logic inverters.

The minimum operating frequency of the dynamic divider shown in Figure 8 was characterized as a function of supply voltage. Results are shown in Figure 9. With increasing supply voltage, gate leakage currents, which discharge the floating nodes, increase exponentially while the logic threshold voltages increase linearly. Together these effects account for the minimum frequency increasing exponentially with increasing power supply. Accurate measurements of the maximum frequency of operation were not possible because of transmission line termination problems but the divider operated beyond 250 MHz.

3 Multiplier Design

A block diagram of the serial multiplier is shown in Figure 10. Only static logic was used in the design. Differential clocks were used for improved performance. Differential clocks increase the dynamic power dissipation which helps distinguish the dynamic power dissipation of the multiplier from the static power dissipation.

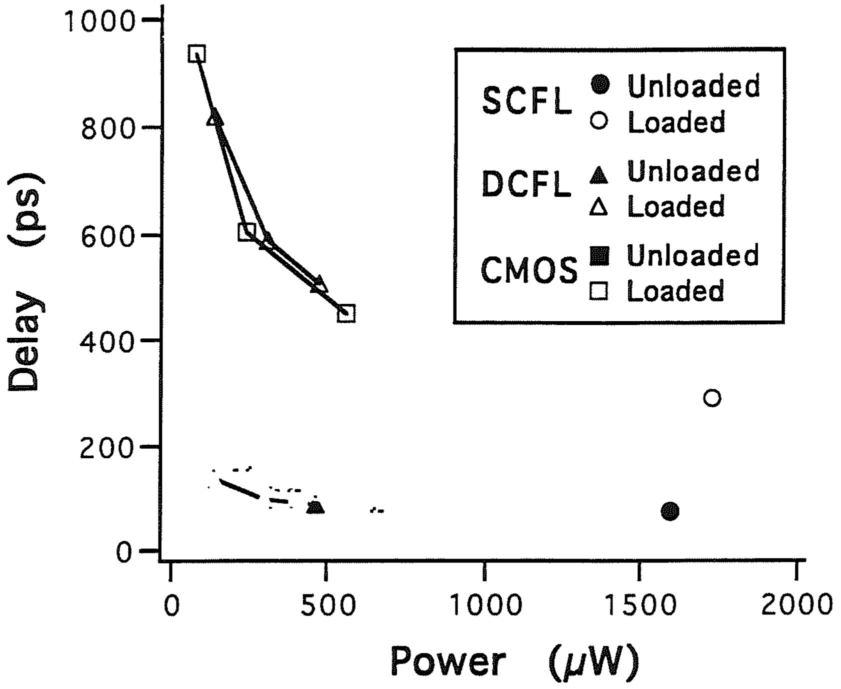


Figure 7: Loaded and unloaded ring oscillator delay vs. power for several CHFET logic families.

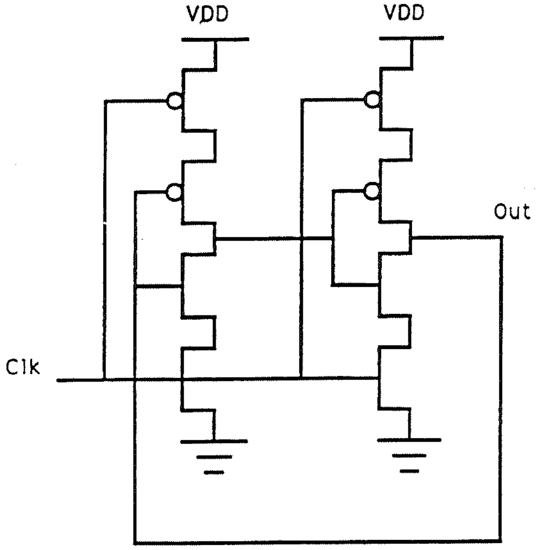


Figure 8: Schematic diagram of dynamic divider circuit.

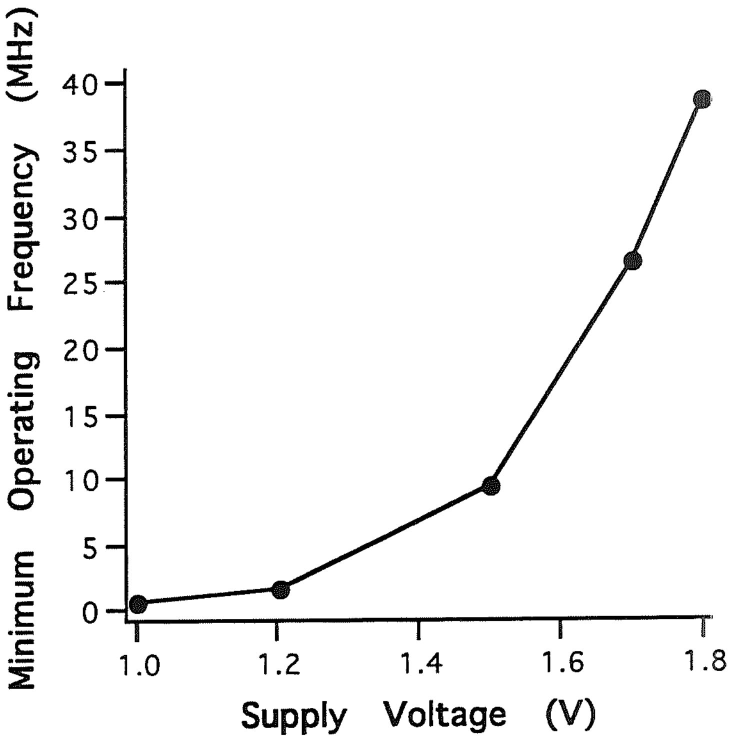


Figure 9: Minimum operating frequency of dynamic divider as a function of supply voltage.

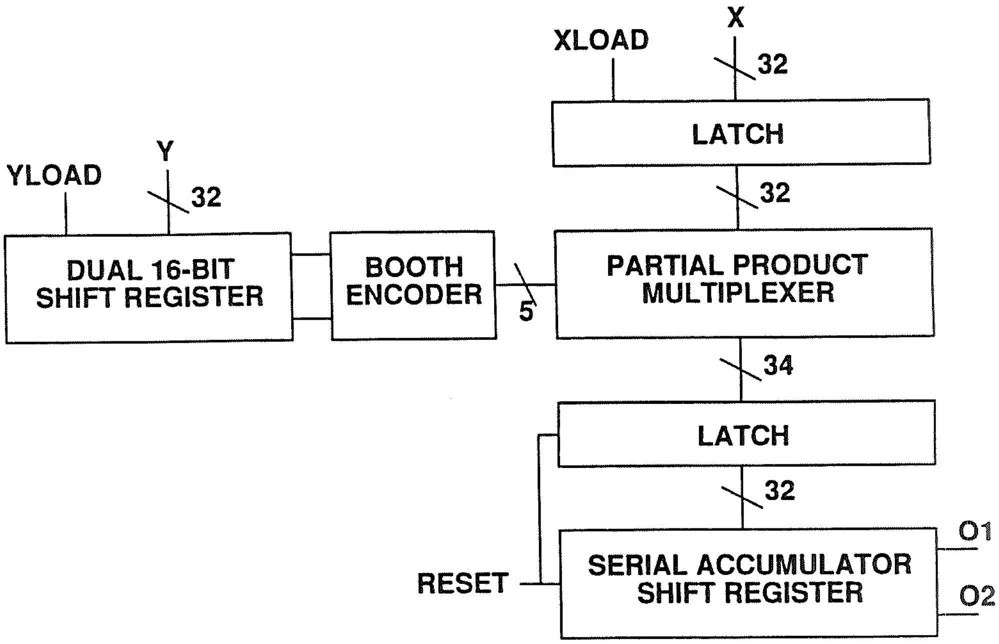

XLOAD
X
32
LATCH
32
YLOAD
Y
32
DUAL 16-BIT SHIFT REGISTER
BOOTH ENCODER
5
PARTIAL PRODUCT MULTIPLEXER
34
LATCH
32
RESET
SERIAL ACCUMULATOR SHIFT REGISTER
O1
O2

Figure 10: Block diagram of the serial multiplier

The Y multiplier inputs are latched into dual 16-bit shift registers with YLOAD. The X inputs are latched with XLOAD. After a RESET signal clears previous data, the Y inputs are shifted, two at a time, into the Booth encoder. The Booth encoder has five outputs which are used by the partial product multiplexer to output 0, X, -X, 2X, or -2X to the partial product latch. In each stage of the serial accumulator shift register, the 2-bit partial product is added to the previous result after it has been multiplied by 4 (shifted by two bits). The carry from each stage is latched and added during the next clock cycle so that no carry propagate circuits are needed. The result is output 2 bits at a time starting with the least significant bits.

Pass-transistors were used extensively in the dual shift register and the multiplexer. Figure 11 shows a schematic of one stage of the shift register. The load/shift multiplexer as well as the master and slave latches use pass transistors.

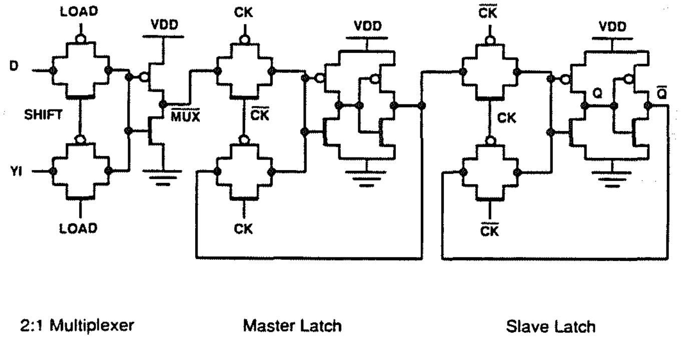


Figure 11: Implementation of the shift register using pass-transistor.

The accumulator shift register was designed using complex gates. Logic was incorporated into the shift register latches. Figure 12 shows the schematic of a latch with an imbedded 3-input NAND. Simulations indicated that these would provide higher performance than combinatorial logic followed by standard flip-flops even though four n-type FETs are placed in series. The longer delay from these complex gates is offset by fewer gates in the critical path.

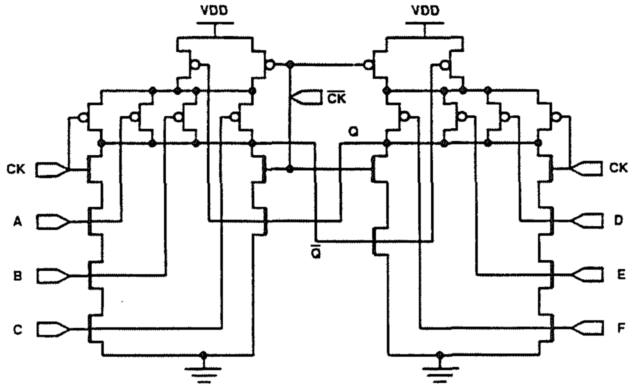


Figure 12: Complex logic gate used in the accumulator shift register.

A 32-bit by 32-bit result is obtained in 34 clock cycles. The 64-bit result requires 32 cycles, reset requires another cycle, and there is an additional cycle for a pipeline stage. In general, a 2N-bit by 2M-bit result can be obtained in N+M+2 clock cycles for M < 17 , where 2N is the number of bits for the Y input and 2M is the number of bits for X. After M+2 clock cycles, the partial products are all zero. The multiply time in a serial multiplier can be improved substantially by using a fast adder to add the two 32-bit numbers stored in the latches of the accumulator shift register after the partial products all become zero. A 32 by 32 multiply would then require only 18 cycles plus the fast adder delay.

Figure 13 shows a photomicrograph of the multiplier. The die size is 5.8 mm by 2.8 mm. Standard cells were placed and routed by hand. The design consists of about 3000 equivalent 2-input gates.

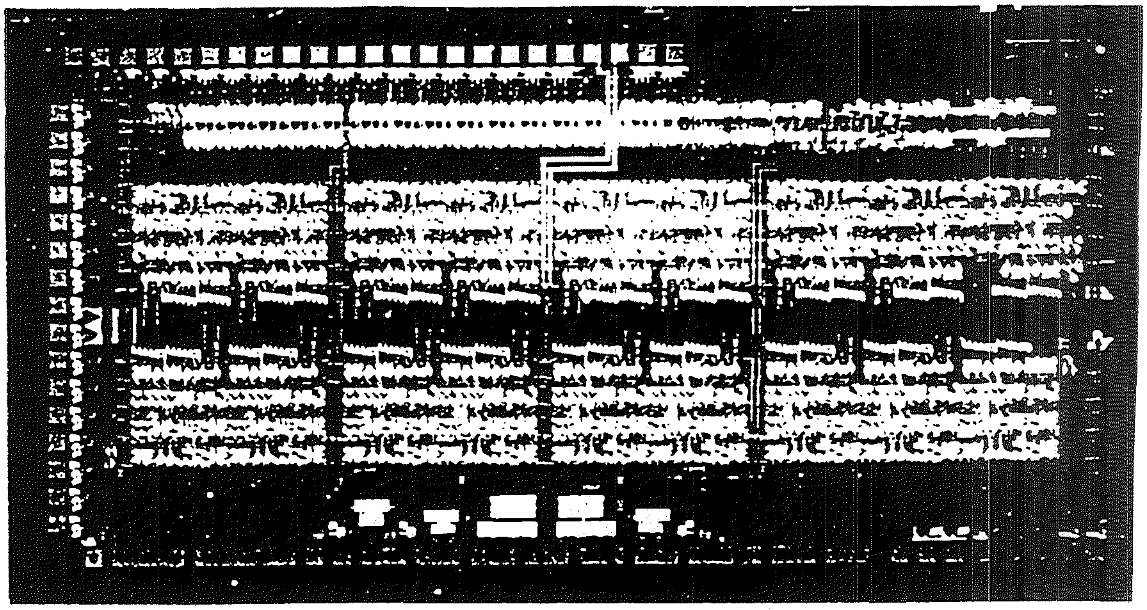


Figure 13: Photomicrograph of multiplier. Die size is 5.8 mm by 2.8 mm

4 Multiplier Results

Fully functional circuits were obtained with wafer yield averaging 20% over the four wafers characterized.

Figure 14 shows the total power dissipation as a function of the maximum operating frequency for wafers A and B. The maximum frequency is 500 MHz with a power dissipation of 2.5 W at a power supply voltage of 2.36 V. The static power dissipation, dominated by gate leakage, increases quadratically as a function of supply voltage while the maximum operating frequency increases linearly with supply voltage. Figure 15 shows the two serial output waveforms at 500 MHz operation.

Figure 16 shows the power dissipation of the multiplier as a function of clock frequency at supply voltages of 1.2 V, 1.5 V and 1.8 V for wafers A and B. The static power dissipation of wafer A is higher than for wafer B because of the higher gate leakage on that wafer. The dynamic power dissipation is linear with frequency for a given supply voltage. The performance of wafer A is better at the same supply and is consistent with the ring oscillator

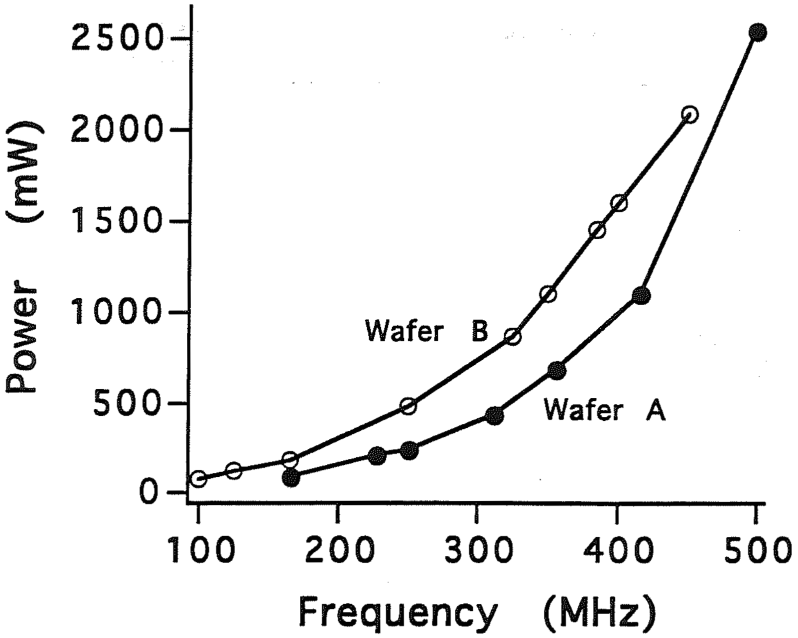


Figure 14: Total power of serial multiplier vs. operating frequency for wafers A and B.

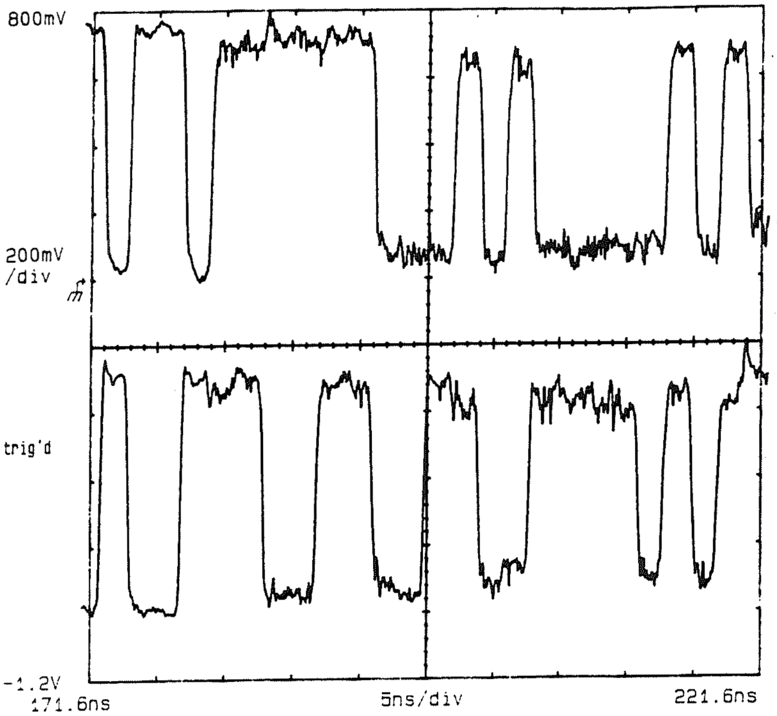


Figure 15: Oscilloscope traces of the two serial outputs at 500 MHz operation.

results. Because of its lower threshold voltages, the performance of wafer A relative to wafer B improves at lower supply voltages.

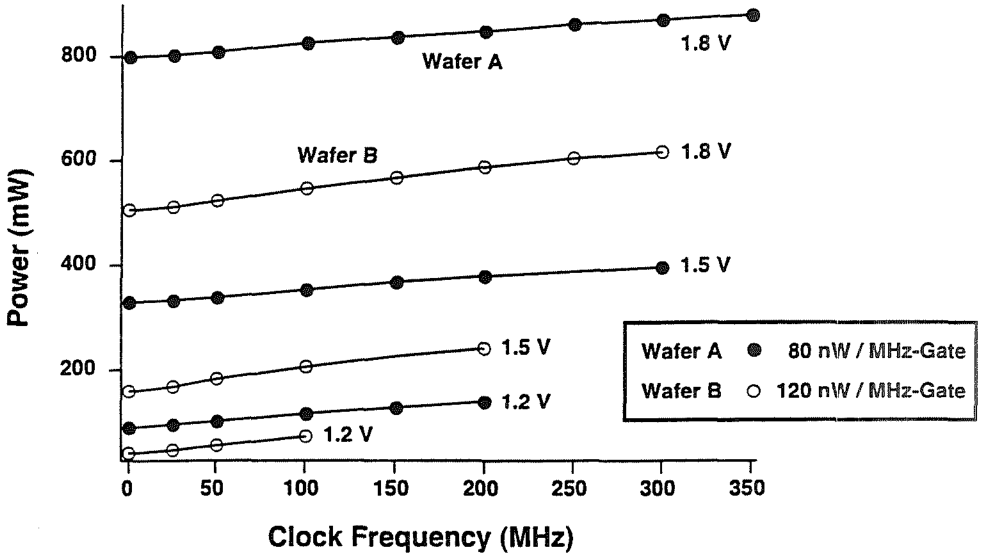


Figure 16: Power dissipation vs. clock frequency at three voltages.

The slope of the dynamic power dissipation as a function of frequency is almost independent of the supply voltage and is about 50% lower for wafer A. The dynamic power, even for a highly synchronous circuit such as this, is much smaller than the static power dissipation and illustrates the importance of reducing the gate leakage of the CHFET devices.

The multiplier could be operated with a power supply voltage as low as 0.73 V at a power dissipation of 2.2 mW on wafer A. Minimum power supply voltage on wafer B was about 0.9 V with a power dissipation of about 4 mW. These measurements were made at 5 MHz. At this clock rate, the total power dissipation is about 1μW per equivalent gate.

5 SEU Characterization

Space applications require radiation hard technologies. Special test structures were laid out to evaluate charge collection and SEU characteristics of CHFET technology. A comparison of SEU effects in MESFET, CHFET and InP is discussed by Hughlock et. al [4]. Figure 17 shows their SEU cross-section results versus linear energy transfer (LET) for the GaAs MESFET, CHFET and InP HFET latches. Charge collection effects on n-type and p-type devices showed an absence of excess charge collection as is seen in MESFET technology. The LET of the CHFET latches are about 6 MeVcm^2/mg versus only 1 MeVcm^2/mg for the MESFET latches. This difference is mainly due to the absence of excess charge collection mechanisms. The saturation cross-section of the GaAs MESFET latches is an order of magnitude higher that the channel area of the sensitive FETs. This is attributed to electric field lines at the MESFET gate edge creating a depletion region extending well beyond the active channel region and into the substrate, thus collecting charge from a much larger area. CHFET and InP HFET do not show any signs of this excess charge collection effect.

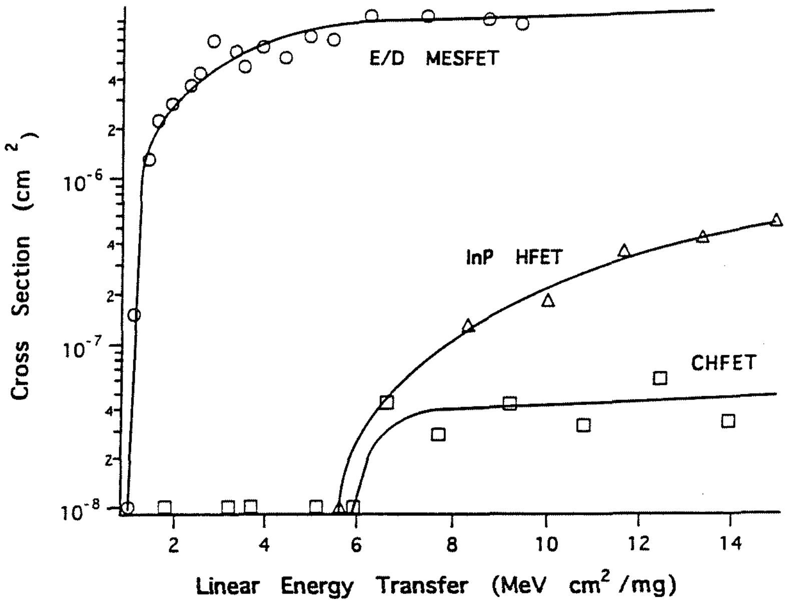


Figure 17: SEU cross-section vs. LET for MESFET, CHFET and InP HFET latches

CHFETs do not exhibit the gate upset mode found with MESFETs where charge is collected at the gate and can upset the latch. The use of cross-coupled resistors to harden latches will not work with E/D MESFET technology because of this gate collection mode. It appears that CHFET latches can be hardened with the cross-coupled resistor technique.

6 Conclusion

A 32-bit serial multiplier was successfully designed and fabricated using CHFET technology. An average wafer yield of 20% was obtained. A power dissipation of 3 mW (about 1 uW per equivalent gate) at 5 MHz operation was demonstrated which is among the lowest reported for any GaAs logic circuit. The maximum operating frequency was 500 MHz with power dissipation of 2.5 W at a supply voltage of 2.36 V.

Lower threshold voltages were obtained with implanted n-type transistors on non-delta doped wafers. This resulted in up to a 50the static power dissipation at the same operating frequency since lower supply voltages could be used.

CHFET technology is a good candidate for space applications. It has a low delay-power product and is harder to SEU that unhardened silicon technology.

Acknowledgements

The authors would like to thank Paul Shaffer and Tim Williams for their help with the design and layout, David Grider and Jim Nohava at Honeywell for their helpful discussions and assistance with wafer processing.

References

[1] D.E. Grider, I.R. Mactaggart, J.C. Nohava, J.J. Stronczer, P.P. Ruden, T.E. Nohava, D. Fulkerson, and D.E. Tetzlaff, "A 4 KBit Synchronous RAM Based upon Delta-Doped Complementary Heterostructure Insulated Gate FET Technology", GaAs IC Symposium, pp. 71-74, Oct. 1991.

[2] D.E. Grider, A.I. Akinwande, R. Mactaggart, P.P. Ruden, J.C. Nohava, T.E. Nohava, J.E. Breezley, P. Joslyn, and D. Tetzlaff, "Development of Static Random Access Memories Using C-HIGFET Technology", GaAs IC Symposium, pp. 143-146, Oct. 1990.

[3] G. S. LaRue, D. E. Grider, "Complementary Hfet 32-bit Serial Multiplier", GaAs IC Symposium, pp. 89-92, Oct. 1992.

[4] B. Hughlock, A. Johnson, T. Williams, and J. Harrang, "A Comparison of Charge Collection Effects Between GaAs MESFETs and III-V HFETs", IEEE Transaction on Nuclear Science, Vol. 39 No. 6, pp. 1642-1646, Dec. 1992.

NATURE'S CRUCIBLE: Manufacturing optical nonlinearities for high resolution, high sensitivity encoding in the compound eye of the fly, *Musca domestica.*

Mike Wilcox
Biomedical Research
Department of Anatomy
University of New Mexcio
Albuquerque NM

Abstract - **The number of pixels per unit area sampling an image determines Nyquist resolution. Therefore, the highest pixel density is the goal. Unfortunately, as reduction in pixel size approaches the wavelength of light, sensitivity is lost and noise increases. Animals face the same problems and have achieved novel solutions. Emulating these solutions offers potentially unlimited sensitivity with detector size approaching the diffraction limit. Once an image is "captured", cellular preprocessing of information allows extraction of high resolution information from the scene. Computer simulation of this system promises hyperacuity for machine vision.**

1 Trade-Offs with High Resolution and Sensitivity

The number of pixels in a detector array used to sample and encode an image determines the maximum resolution of that image. Therefore, the higher the density of receptors in the retinal array, the more detail can be resolved from the image. For real systems, this choice of pixel size and spacing is limited by diffraction principles. In machine vision, reduction in pixel size quickly reaches a sensitivity limitation, where excessively long integration times limit the ability to encode visual information in real time. Most high resolution silicon chips have limited pixel sized of 6-9 microns.

2 Dealing with Sensitivity in the Retina

Animals face the same problems of sensitivity in ambient light and yet they have found novel ways to overcome those problems. Implementation of their solutions could solve the sensitivity problem for machine vision as well. The highest density packing of elements in two dimensions incorporates hexagonal geometry. DeMonastero et al. [1] showed just such a packing for photoreceptors in the primate retina. This packing is typical of most animals whose eyes form complex images. The receptive elements (microvillar organelles inside the retinal photoreceptors) that house the visual pigment, have the same diameter of one micron. This is the minimum size for these phototransducing elements because they are, in fact, waveguides. The principle wavelength transmitted by the wave guide is the absorption peak of the visual pigment, rhodopsin (480nm). The waveguide description parameter V limits the diameter to a minimum of two times the transmitted wavelength [2]. On one hand, larger values of the parameter would provide more light gathering power but fine resolution gained by high density packing of detectors would be comprised. On the other hand, smaller values for size of this cellular organelle would violate the waveguide principle for total internal reflection and the structure could no longer contain the traveling wave. Rather than enlarge the waveguide diameter, many nocturnal animals use the length of a waveguide as a free parameter to increase retinal sensitivity, by increasing the probability that a photon will not be absorbed by the visual pigment, some nocturnal animals like deer and cats, have added a reflecting tapetum to the end of the waveguide to allow one more pass for unabsorbed photons through the semi-opaque matrix of the waveguide.

A good example of this type of optical system is found in the compound eye of the common housefly, Musca domestica, whose head is shown in Figure 1. These animals have compound eyes, that is, they are made up of many individual eyes, about 3,000 in Musca, each oriented toward a different point in space. In the focal plane of each corneal lenslet, there is a miniretina consisting of 8 photoreceptors. Each receptor cell has its own cylindrical organelle called a rhabdomere (see Figure 2), that houses the visual pigment and provides phototransduction for the cell. In Figure 2, a thin slice through the outer retina is shown. The lens images light onto the tips of the rhabdomeres. The membranous organelle is made up of microvilli and continues along the side of the cell, about 100 microns in length. The electron micrograph in Figure 3 is a cross section through one miniretina and shows the detector arrangement. Six peripheral cells (R1-6) surround one of the two tandemly arranged central cells (R7, R8). Tightly packed membranes in each rhabdomere provide a refractive index higher than the extracellular fluid bathing the cell and, therefore, formsan optically isolated waveguide. We can see the waveguide in an living animal and demonstrate its waveguide properties by simply neutralizing refraction of the corneal lens by placing a drop of water between the microscope objective and the fly's eye and illuminating. If a fluorescent dye is placed in the extracellular space of the animal's retina, fluorescent emission will be trapped by the waveguide and piped up the rhabdomere, into the microscope objective. A shown in Figure 4, the retinal array of photoreceptors is directly visualized in an intact animal.

In many nocturnal insects and deep sea creatures the size of their waveguides can reach incredible lengths in order to capture sparsely available photons, in an environment where survival is a fantastic achievement.

3 Implementation

Photodetectors based on waveguide principles is feasible for machine vision. Achieving both high sensitivity and high resolution is our goal. Our immediate problem for machine vision is to decrease the size of each pixel to approach to sensitivity is to increase the gain on the photodetector, but this approach is not immune to noise and, therefore, not practical. Another approach is to increase the surface area of the detector, while maintaining minimal cross sectional area for high density packing of detectors. Photocathodes with waveguide properties could be implemented in two ways, (1) using doped semi- opaque media to generate a photodetecting waveguide or (2) using a waveguide core of glass to transmit incident photons and building the photodetecting material onto the surface with a coating of aluminum on

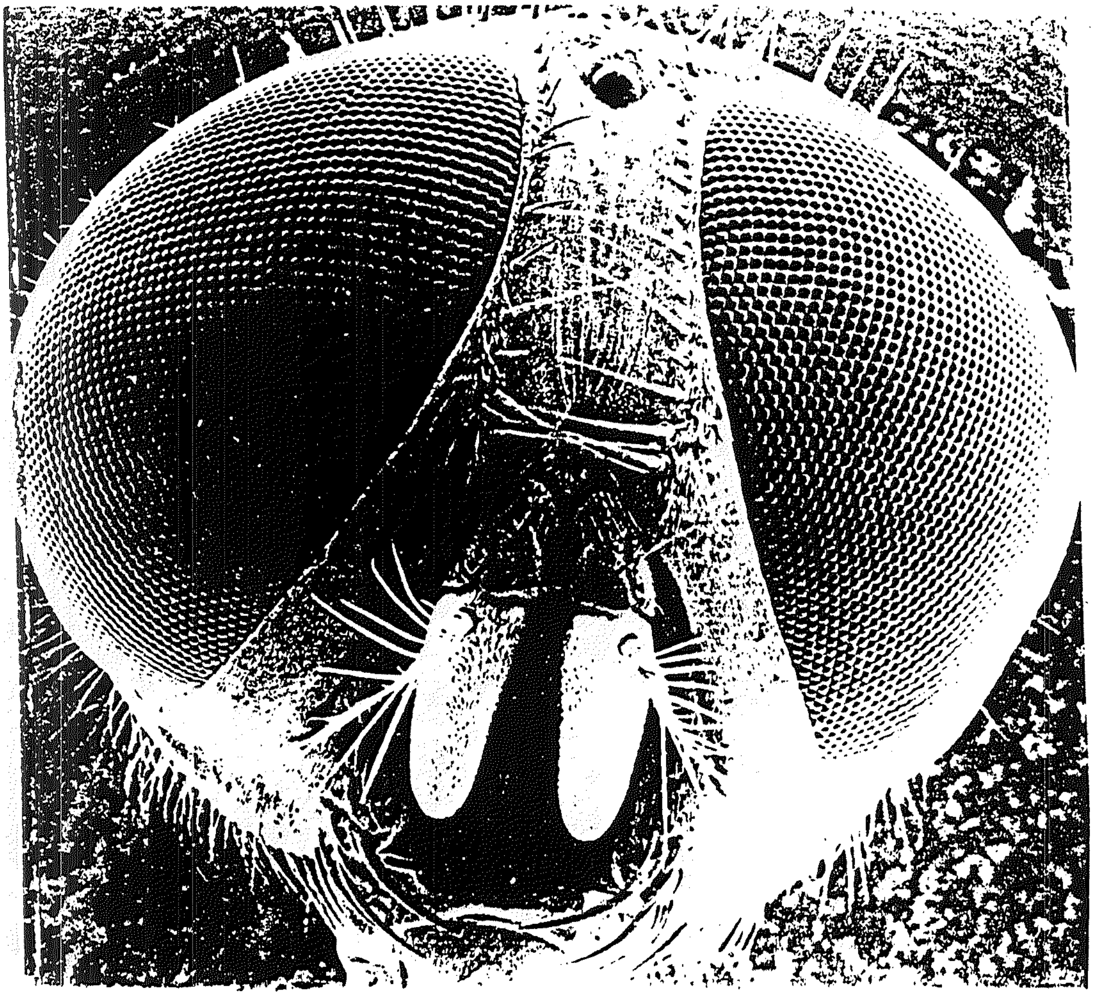


Figure 1:

Figure 2:

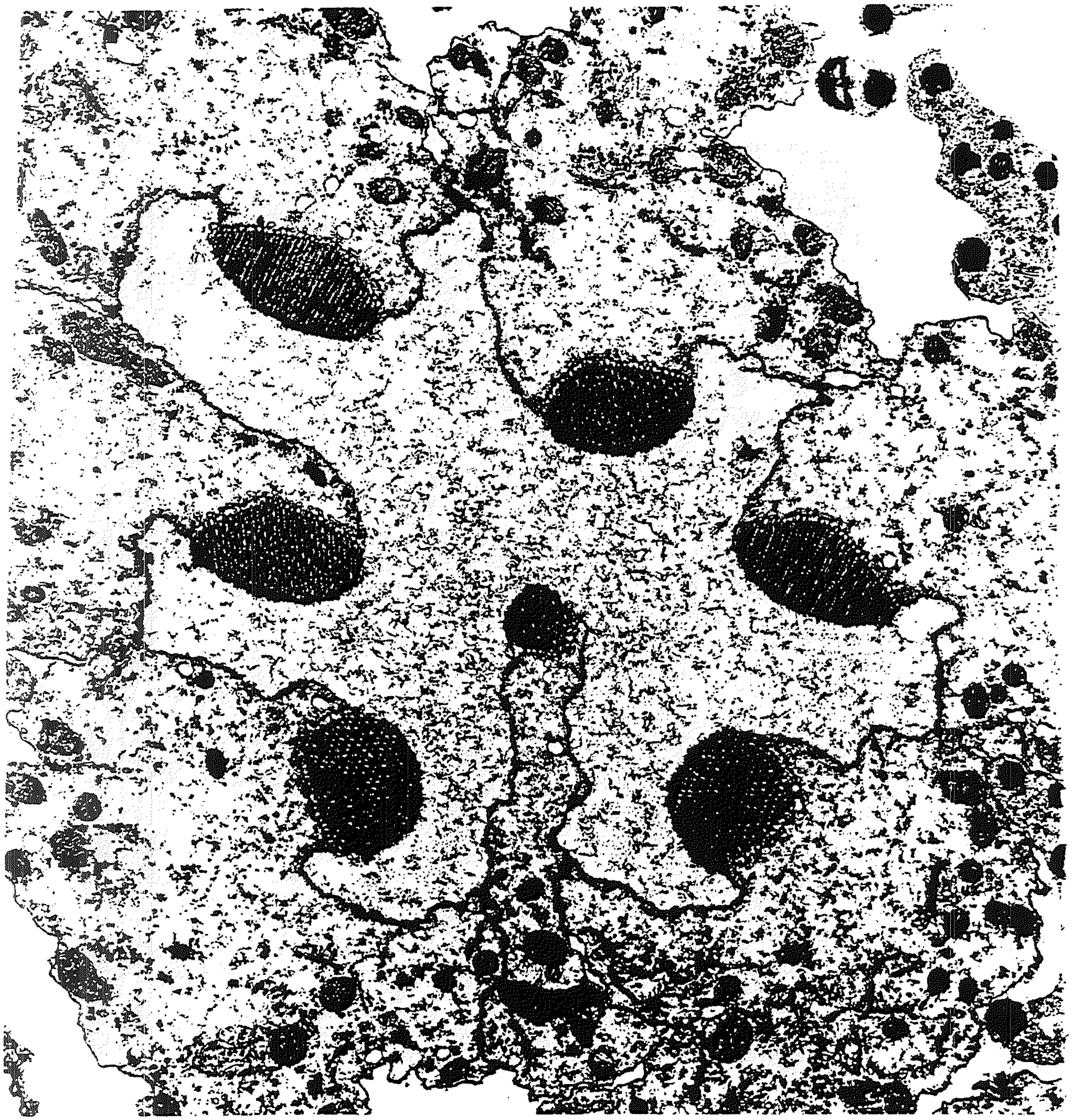


Figure 3:

4.3.6

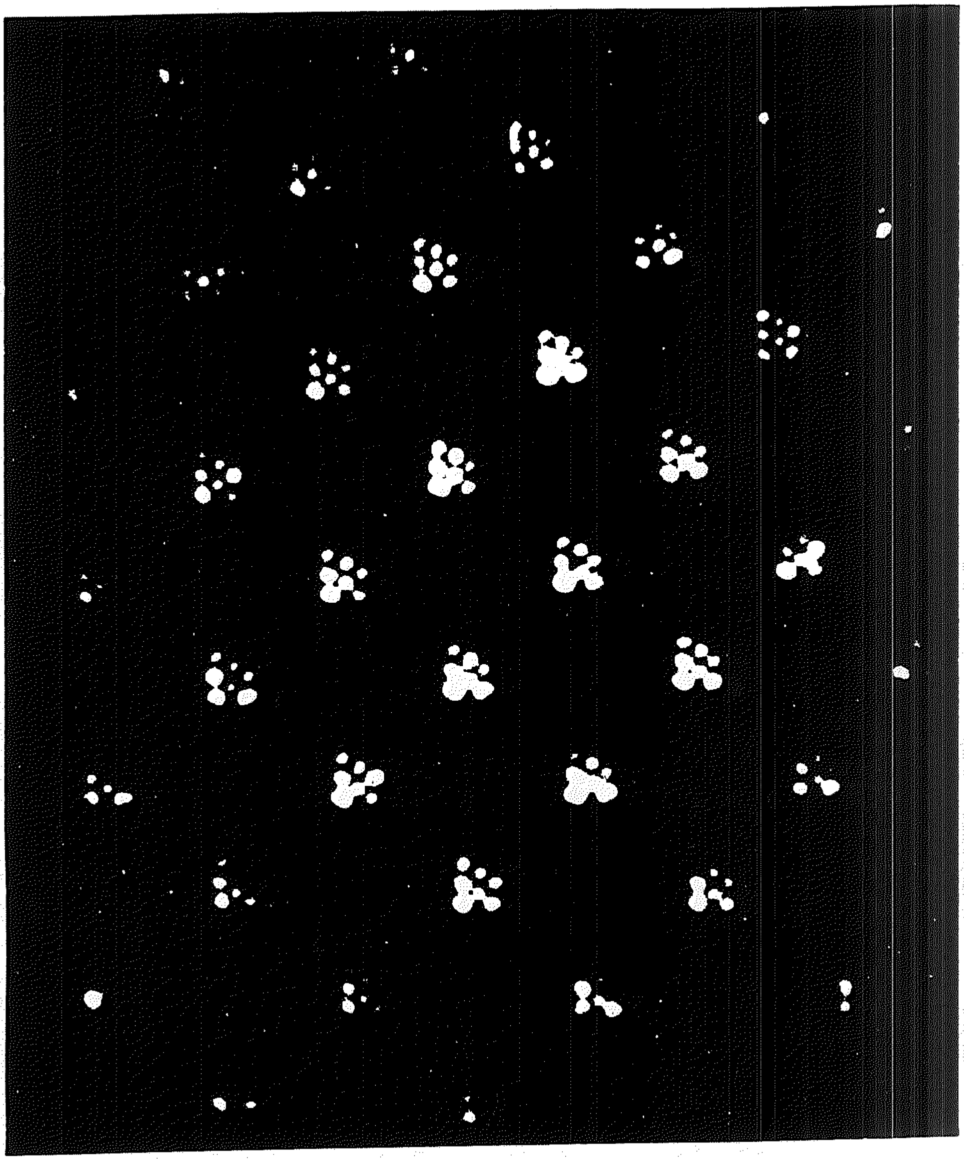


Figure 4:

the outside to provide three functions, (a) optical isolation from neighboring elements, (b) electrical tether for the cathode material to a CCD well and (c) internally directed reflection of nonabsorbed photons. The small diameter of each element allows the highest density packing. Another advantage of using a cylindrical geometry is that volume of the pixel in the matrix is minimized, while surface area would be determined by Schott noise generation from the cathode surface interfering with the signal generated by photon absorption in the photocathode. However, using thermal control, cathodes with enormous surface area could be realized and used for high sensitivity measurements in the dark reaches of space, where inexpensive real-time imaging and fast spectral analysis of images are still dreams.

4 Information Processing in the Retina

In resolving detail from an image, the optical system plays an important role. To form an optical image on a detector array, the classical engineering approaches to choose a lens system having an optical quality, or modulation transfer function, that matches the detector spacing. Snyder et al. [2] have shown that the dioptic apparatus in animal eyes has a modulation transfer function that is 10-100 times superior to granularity of the photoreceptor matrix. The interpretation is that preservation of the high frequency components of the optical image enables the retina to deal with aliasing of the input signal, where the matrix would misinterpret the input signal. Interestingly, animals whose eyes form high quality images, are able to interpret the image with real resolution 10-100 times better than predicted from the spacing of retinal photoreceptors. It must be more than chance that the magnitude of improvement in both optical quality and real resolution are the same. This ability is called hyperacuity. In spite of its almost universal implementation in animal visual systems, we know little of the actual neural computation the retina uses to encode and interpret signals.

Hints about information processing in the retina have come from comparative study of retinal structure of a number of different animals. Very early on, Cajal and Sanchez (1915) [3] and Zawarzin (1925) [4] pointed out the morphological parallels in retinal cell types of animals as evolutionarily far apart as molluscs, insects and vertebrates, which encompasses nearly every animal in existence. There is one-to-one homology in the scenario we learned in introductory level biology, that photoreceptors talk to bipolar cells which, in turn, talk to ganglion cells, the output of the retina. At the level of the bipolar cell outputs, there are elements called horizontal cells and at the level of the bipolar cell outputs, there are elements called amacrine cells both providing both providing lateral interactions within a plexus of synapses called the outer and inner plexiform layers, respectively. Electrophysiologically, the responses of these cells to similar stimuli are also homologous. If both structure and cellular responses are the similar, our interpretation then, is that the kind of information processing used is also similar. If animals so widely divergent as those mentioned use similar processing techniques, it suggests that there are a few good ways to process visual information and that everyone found out those techniques and held onto them with very little change over the eons in evolutions. Another point of view might be that if animals like flies have had billions of generations to get the genetic material right for construction of the best or most efficient way to process information, we ought to look to them for inspiration to solve similar problems in machine vision. Any way, we have an existence proof of the superior performance of animal visual systems. If we could emulate these processing principles, we could solve visual problems with the same parallel architecture.

Even though the general retinal circuitry is as described, our own retina has millions of photoreceptors and regionally complex circuitry that differs slightly from the center to the periphery of the retina. In the Neurocybernetics Lab at UNM, we have bypassed this complication by choosing to study the eye of an animal. Second, there is one-to-one projection of the axons from the miniretinas to the underlying structures that process the information. Moreover, the boundaries between elements is distinct. Third, the arrangement of the neural elements within the underlying layers is absolutely specified and the internal structure of each subunit is exactly the same as its neighbors, as suggested by Figure 5.

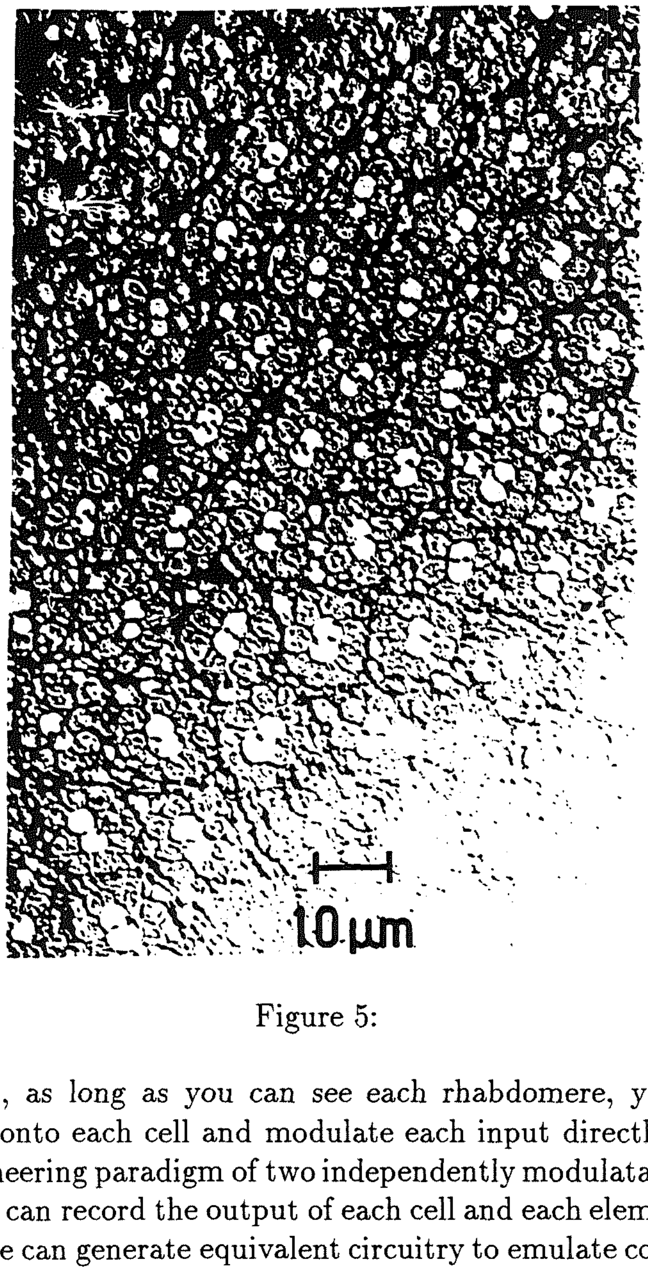


Figure 5:

Recalling Figure 4, as long as you can see each rhabdomere, you can also image an independent stimulus onto each cell and modulate each input directly. In other words, we have the classical engineering paradigm of two independently modulatable inputs and a single recorded output. If we can record the output of each cell and each element along the pathway of the visual system, we can generate equivalent circuitry to emulate communication between various cells in the retina, in response to the input of the photoreceptors.

We have made a neural network model of signal encoding at the first synapse [5].The circuit is depicted in Figure 6. The input layer to each processing elements formed by six photoreceptors (R1-6), one from each of six adjacent lenses. The optical arrangement of these six photoreceptors (open symbols under each of the hexagonal facet lenses of the compound eye) is such that they form an over- lapped sampling of the far field. The receptive field of each individual receptor has a Gaussian profile, subtending a solid angle of 2 degrees. The amount of overlap is 0.3 degrees [6].

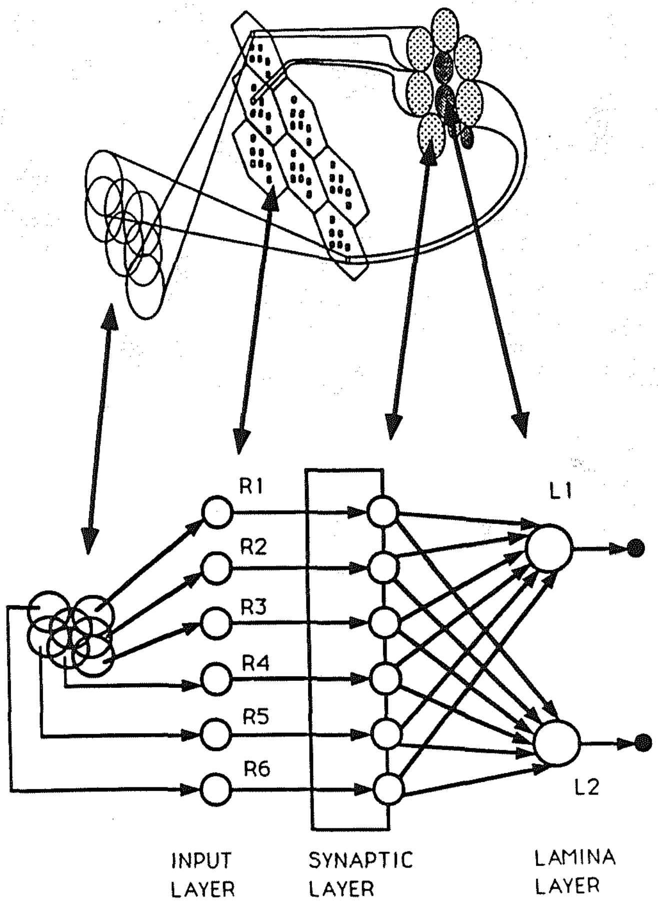


Figure 6:

The interaction of these six inputs at the first synapse shows how information is encoded by the system. during the development of the adult form, the axons of these six photoreceptors distribute themselves so that they arrive at the same processing element in the underlying plexiform layer called the lamina, where they make synaptic contact with two monopolar cells called L1 and L2 [7, 8]. The relative position of the photoreceptor axon terminals with respect to L1 and L2 is shown in the electron micrograph of Figure 7, showing the membranes of the receptor terminals and the dendrites of both monopolar cells. Each of these processing units, called a cartridge, is surrounded by a darker- staining, insulating process from a glial cell [9]. This cell process functionis to isolate the cartridge from its neighbors.

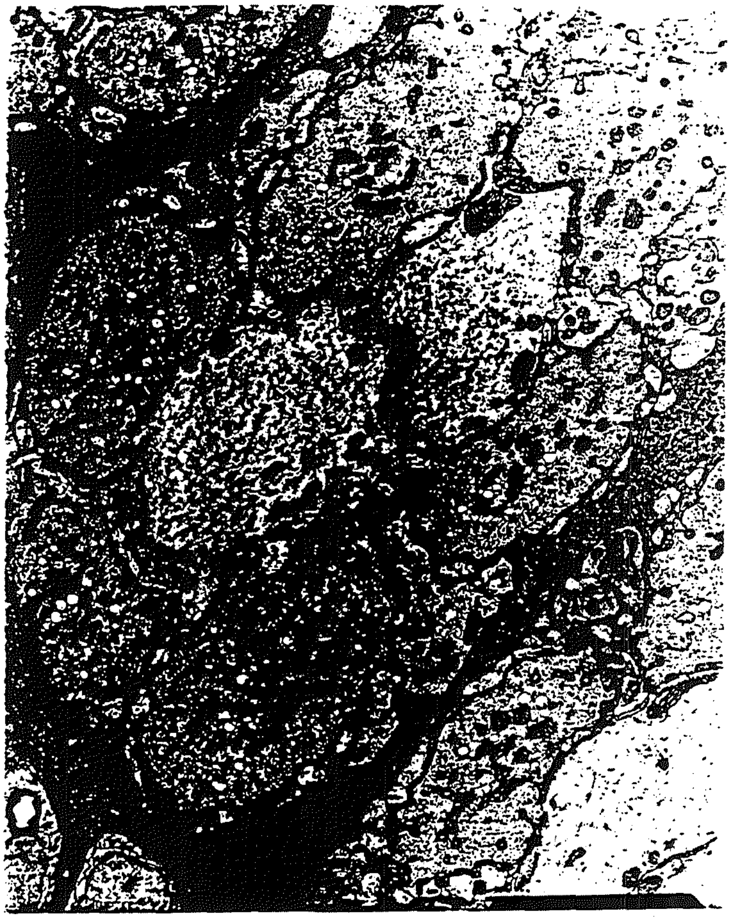


Figure 7:

Within the cartridge, photoreceptor synaptic terminals are electrically coupled to one another by gap junction proteins [10]. The function of these gap junctions enables the structure to optimize the photoreceptor signals at different levels of light adaptation. At low luminance, electrical coupling provides presynaptic pooling of signals in order to augment the signal, effectively summing the individual inputs to reduce the effects of noise. However, when luminance is optimal, such pooling would degrade acuity gained by the alignment of photoreceptors in the overlapped fashion depicted in Figure 6. At this level of light adaptation, current shunted from the terminal of the stimulated photoreceptor is drawn back to the cell body of the unstimulated photoreceptors. Photocurrent leaving the depolarized axon terminal bleeds into the extracellular space and depolarizes that restricted compartment. The extracellular voltage now opposes the potential inside the unstimulated terminals within the same cartridge. This arrangement induces current flux backward through axons of the unstimulated receptor cells. This effectively inhibits transmitter release from the unstimulated terminals. Inhibitory current flux is depicted in Figure 8. This current-shunting inhibition of transmitter release generates artificial contrast within the cartridge.

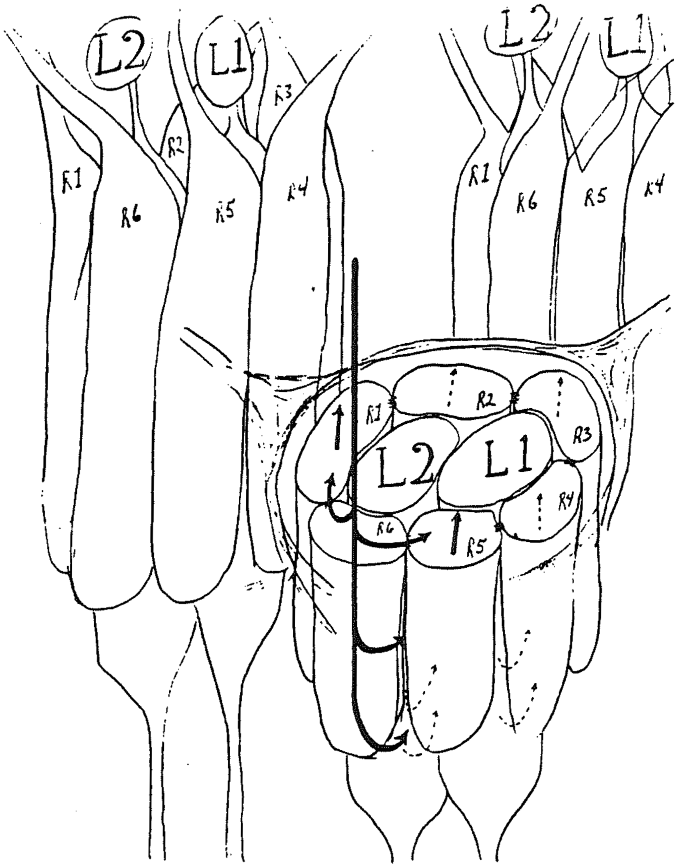


Figure 8:

We have modeled and simulated this interaction between neural profiles. The output generates an orientation coordinate and an incremental displacement response that approaches the improvement in resolution seen in the intact animal [11]. Figure 9 shows a comparison of six linear elements of digital camera and six overlapping Gaussian images of a fly's photoreceptive rhabdomeres. The fine line indicates the trajectory of a moving point source as it crosses the far-field radiation of the photodetectors. In the arrangement of linear, abutting picture elements of a digital camera, motion sensitivity occurs only at the boundary of two elements. The overlapping arrangement, in contrast, is exquisitely sensitive to the position of a point source, indicated by the trajectory of the point source target. In our model, the animal uses linear synaptic processing to exploit optical nonlinearities provided by the Gaussian profiles of the rhabdomeric waveguides, in order to extract high resolution information from the input image. The model does not yet incorporate feedback circuitry, known to exist at this level of processing. Such incorporation will make the circuitry perform even better. However, it is remarkable that such a simple circuit can achieve a log unit of improvement in resolving the position of a point source. These results are encouraging and we look forward to rapid progress in achieving hyperacuity in machine vision and implementation of these principles in hardware that could be mounted directly onto the photosensitive chip, providing preprocessing of the signal and high resolution encoding of images.

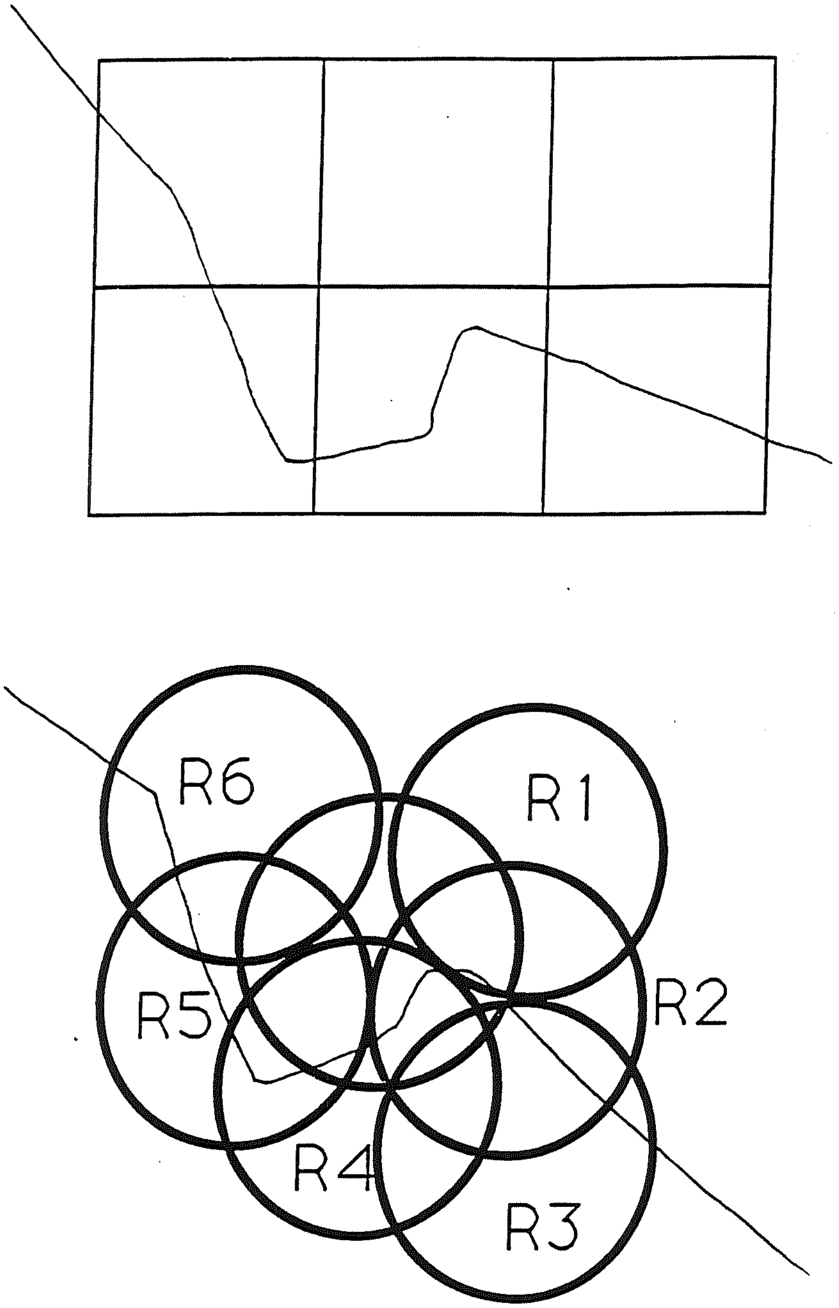


Figure 9:

References

[1] FM DeMonastereo, SJ Schein, EP McCrane Science 213, 1278-81 (1981).

[2] A Snyder, T Bossomaier, A Hughes Science 231, 499 (1985).

[3] SR Cajal, D Sanchez Trabaj. Lab. Investig. Biolog. de la Univer. Madrid 13, 1-168 (1915).

[4] A Zawarzin Ztrsch. fur wissensch. Zool. Leipz. 124, 118-212 (1925).

[5] JA Moya, MJ Wilcox, GW Donohoe Proc. World Congress on Neural Networks, Portland, OR, July 11-15, 1993, vol. 1, pp 3-10.

[6] B Pick Biol. Cybernetics 26, 215 (1977).

[7] V Braitenberg Exp. Brain Res. 3, 271 (1967).

[8] O Trujillo-Cenoz J. Ultrastruct. Res. 13, 1 (1965).

[9] R Sainte-Marie, SD Carlson J. Neurocytol. 12, 243 (1983).

[10] W Ribi Cell Tiss. Res. 195, 299 (1978).

[11] P. Coombe, M Srinivasan, R Guy J. Comp. Physiol. A 166, 23 (1989).

[12] GM Shepherd Nuerobiology, 2nd ed., Oxford Univ. Press, Inc. 1988.

Session 5
Mixed Signal

Chairman: Don Wiseman

Proton Radiation Testing of an 18 bit A/D Converter

Donald C. Thelen and Don R. Wiseman
NASA Space Engineering Research Center for VLSI System Design
University of New Mexico
Albuquerque, New Mexico 87106

Abstract - **An 18 bit first order incremental analog to digital converter (ADC) has been constructed in a 2μ commercial CMOS process. The ADC was exposed to energetic protons at the Harvard Cyclotron Laboratory to simulate latch-up and noise behavior in low earth orbits. Results of the test indicate no latch-up problems, and only a small loss in accuracy while operating in proton radiation with energies similar to the South Atlantic Anomaly.**

1 Introduction

In low earth orbits, satellites and the chips on board encounter protons which are trapped in the Van Allen belts. These protons can cause SCR latch-up [1, 2] and single event upsets in an unsuspecting chip thereby causing a system failure. Charge deposited by protons can also be seen as noise by analog circuits, resulting in a loss of accuracy [3, 4]. One way to test for such undesirable behavior without paying for a launch, is to expose chips to protons from a particle accelerator. This paper presents the results of such a test on an 18 bit analog to digital converter (ADC) at the Harvard Cyclotron Laboratory. This paper includes a description of the ADC and the test set up, then presents the results of the test, and some conclusions.

2 A/D Converter Circuit

We chose an incremental ADC [5, 6, 7] for space applications, because of its noise performance, and linearity. The incremental ADC is really a switched capacitor version of the charge balance ADC [8, 9]. Noise on the ADC input is integrated along with the desired signal, so only the average value of the noise, which is usually zero, corrupts the result of the conversion. Also, a bad comparison caused by noise on the comparator input is corrected by the next integration cycle. The operation of the first order incremental ADC, shown in Figure 1 is summarized here. The ADC starts in the reset mode, where the integrating capacitor $C2$ is discharged, and the digital counter on the output of the comparator is set to zero. The phase 1 switches are then closed, and V_{in} is sampled on $C1$. The phase 1 switches open, then the phase 2 and phase 4 switches close, transferring the charge $-V_{in}C1$ to the $C2$. The output of the integrator is now $V_{int} = V_{in}C1/C2$. Next the phase 2 switch opens, and the comparator is strobed. If $V_{in} > 0$, then the output of the comparator is a logic 0, and $+V_{ref}$ is applied to the input of the phase 3 switch. The phase 3 switch then closes (the phase 4 switch is also still closed) which transfers either $+V_{ref}C1$ or $-V_{ref}C1$ on $C2$, depending on the output of the comparator. If $V_{in} > 0$, the new integrator output is

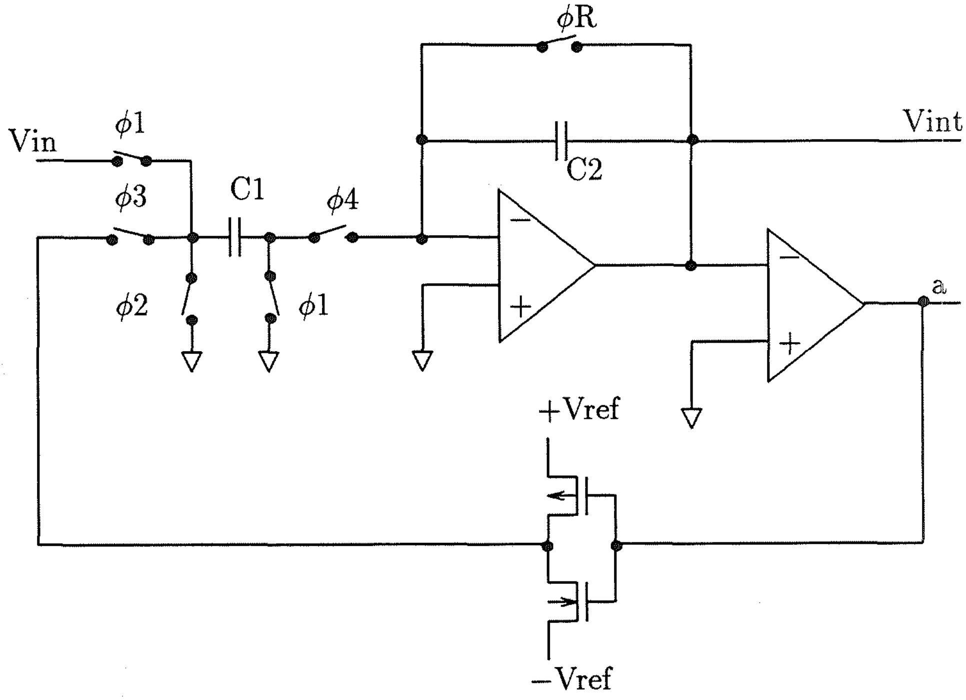


Figure 1: First Order Incremental ADC

$V_{int} = (V_{in} - V_{ref})C1/C2$, while if $V_{in} < 0$, $V_{int} = (V_{in} + V_{ref})C1/C2$. A counter increments if the output of the comparator is a zero. This cycle repeats n times with the same input voltage, until the output of the integrator is given by equation 1.

$$V_{int} = \alpha(nV_{in} - V_{ref}\sum_{i=1}^{n} a_i) \tag{1}$$

where α is the integrator gain $(C1/C2)$, and a_i are the individual comparator outputs which have the values $+1$, or -1. Solving for the ratio of the input voltage to the reference voltage yields:

$$\frac{V_{in}}{V_{ref}} = \frac{V_{int}}{n\alpha V_{ref}} + \frac{1}{n}\sum_{i=1}^{n} a_i \tag{2}$$

We can interpret equation 2 to be the average of the comparator outputs, plus a residue. To get n bits of resolution, 2^n integration/comparison cycles are required, which means this technique is as slow as an integrating ADC. With a clock rate of 1.67MHz, we get about six conversions per second with 18 bit accuracy.

The analog portion of an 18 bit incremental ADC was designed in a $2.0\mu m$ commercial process with an epitaxial layer. The chip was laid out with extensive substrate and well tie downs to make SCR latch-up difficult. The die were packaged in a standard 40 pin ceramic DIP with a taped on lid.

3 Radiation Test Setup

The device under test (DUT) was mounted without the lid in the proton beam perpendicular to the flow of protons, to maximize the flux. A Lucite acrylic resin wheel of variable thickness was rotated between the source of protons, and the DUT to spread the energy of the protons to a spectrum similar to that seen in the South Atlantic Anomaly (SAA) with $4mm$ aluminum shield [10]. The resulting protons had energies between 5 and 120MeV. The flux of the SAA is about $2.4 \times 10^3 protons/(cm^2 sec)$ with $2gm/cm^2$ shielding, which corresponds to about $1.73rad(Si)/hour$[11, 12]. For this experiment, the flux was adjusted to approximately 10x and 100x the SAA flux.

The analog to digital converter was supplied a constant input and reference voltage which were derived from the power supplies as shown in Figure 2. The comparator output was sent to an HP counter, which summed the total number of 0's. When the 0 count is divided by the number of clock cycles, the ratio should equal the ratio of input voltage to reference voltage. Extensive by-pass capacitors were added to the board, and careful shielding of wires and ground planes yielded a setup which varied only ±1 count from the center value. This corresponds to an average noise of only $\pm 7.6\mu V$. A digital pattern generator supplied the clock and reset pulses to the ADC. The personal computer, pattern generator, and counter were all linked together through an GPIB interface. Current was monitored to detect a latch-up condition. Electronic equipment had to be placed about 15 meters from the DUT, while the chip was being irradiated. Terminated twinax cables transmitted clock, reset, data, and power to and from the DUT.

4 Radiation Results

Detecting a failure in an analog circuit is not as simple to define as a change from a one to a zero or vise versa in a digital circuit [3, 4]. A proton, or the resultant particles from a nuclear reaction caused by a proton can cause a continuum of errors in an analog circuit. To measure the magnitude of these errors, we created histograms of the output codes from the analog to digital conversion. A baseline histogram of several analog to digital conversions was created just prior each proton exposure to allow comparison of expected ADC behavior with ADC behavior during proton irradiation. Three tests were run, two with a flux of 10 times the SAA flux, while the third was run at 100 times the SAA flux. The baseline and exposed histograms for a 20 second test with approximately 10 times the SAA flux are shown in Figures 3 and 4.

A 20 minute test was run with approximately the same flux, and the results are shown in Figures 5 and 6. The standard deviation and mean did not change significantly for either test at 10 times the SAA flux.

The flux was increase to approximately 100 times the SAA for 20 minutes, and the results are shown in Figures 7 and 8. The standard deviation increased slightly for this rate. We found no increase in power supply current while irradiating the DUT.

5.1.4

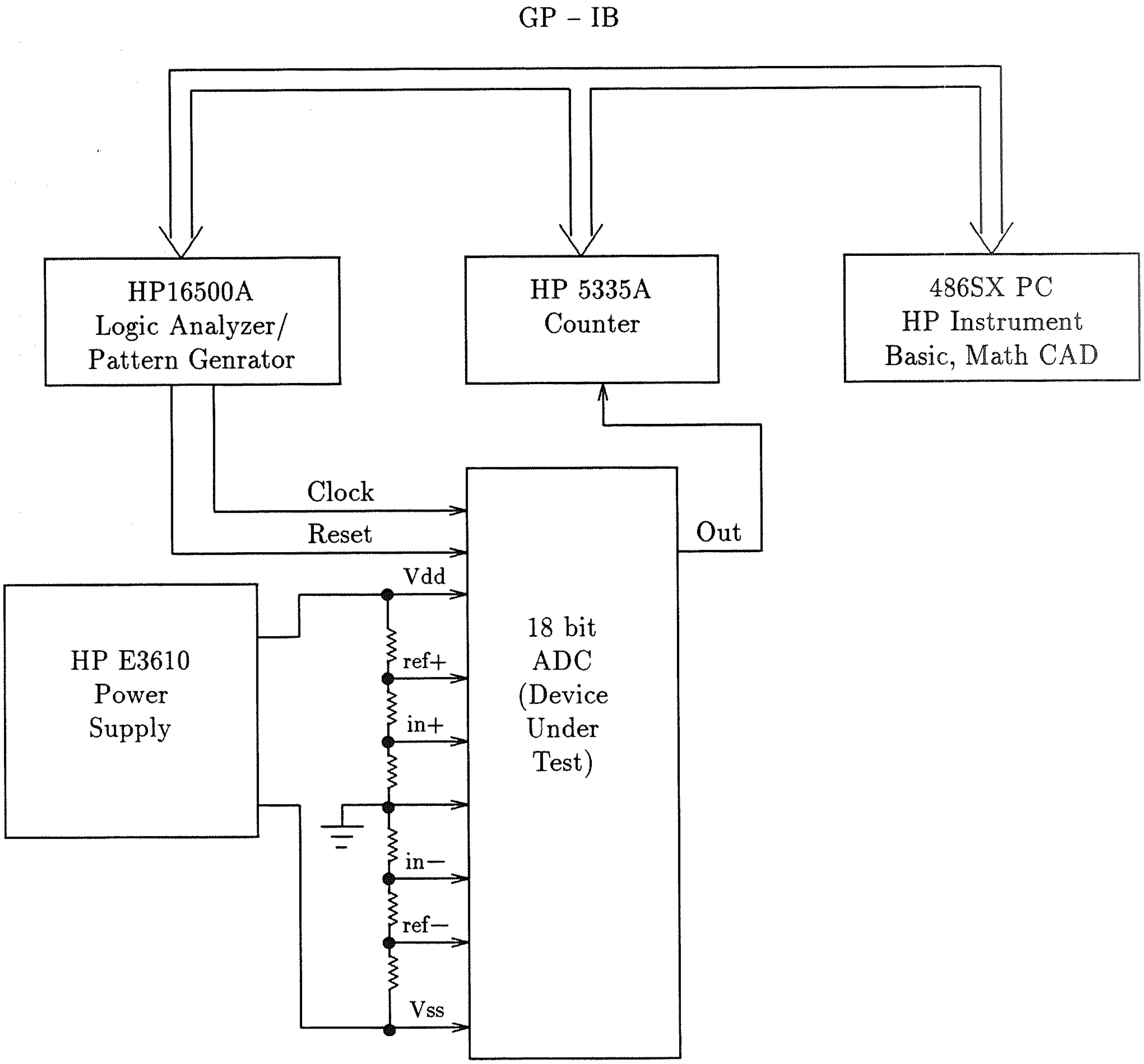
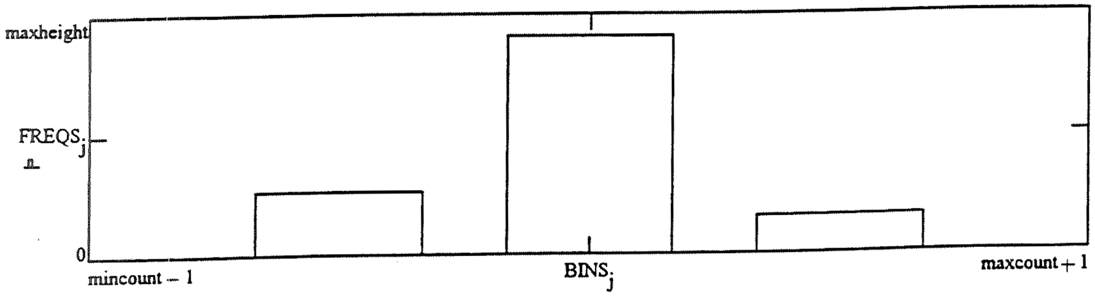


Figure 2: Test Set Up



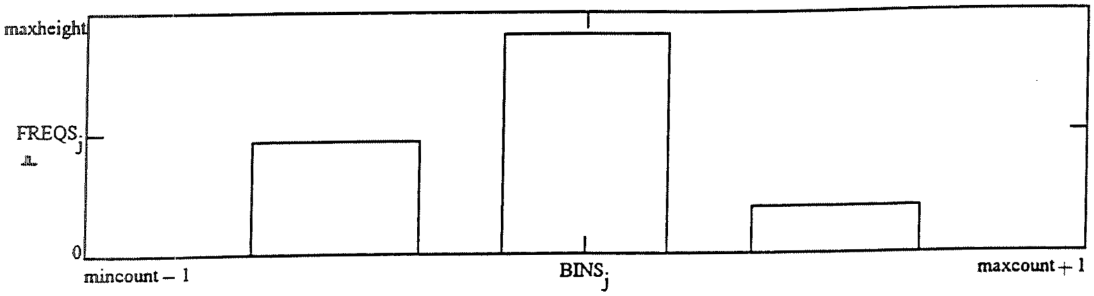
mean(counts) = 234374.92

stdev(counts) = 0.559994

var(counts) = 0.313594

$BINS_j$	$FREQS_j$
234374	10
234375	34
234376	6

Figure 3: 20 Second Baseline Histogram



mean(counts) = 234374.82

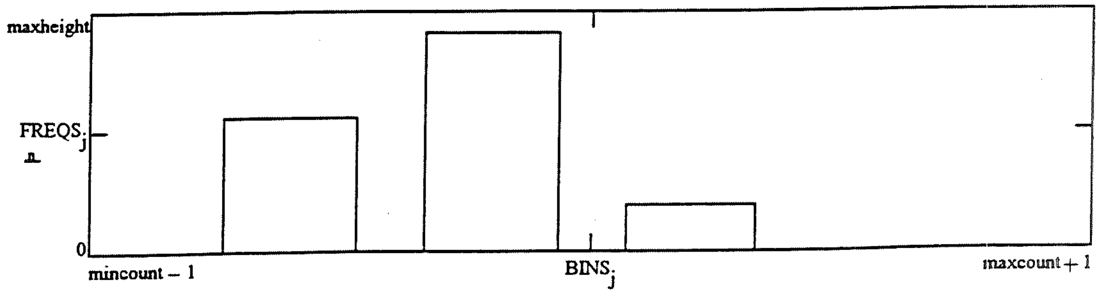
stdev(counts) = 0.622571

var(counts) = 0.387595

$BINS_j$	$FREQS_j$
234374	15
234375	29
234376	6

Figure 4: 20 Second Radiation Histogram, $Flux = 12 rad/hour$

5.1.6



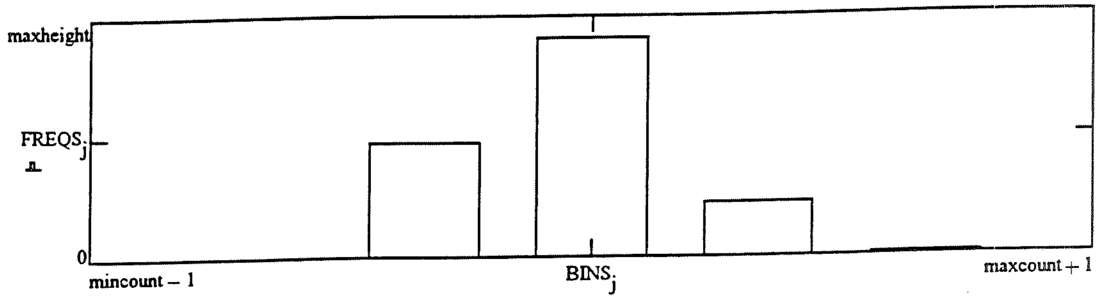
mean(counts) = 234375.784

stdev(counts) = 0.639796

var(counts) = 0.409339

$BINS_j$	$FREQS_j$
234375	335
234376	547
234377	117
234378	1

Figure 5: 20 Minute Baseline Histogram



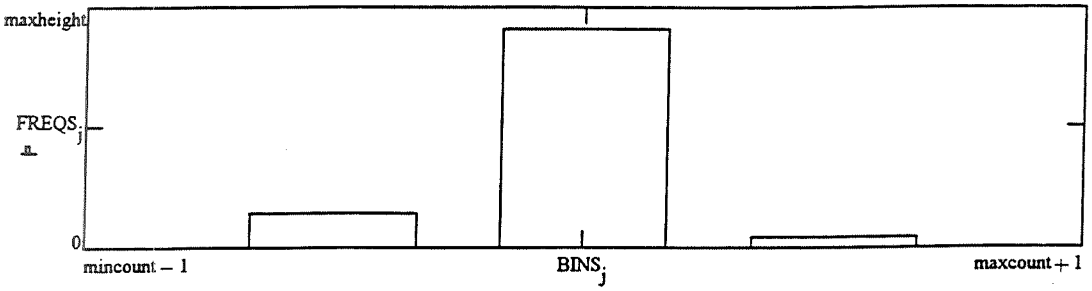
mean(counts) = 234375.8585

stdev(counts) = 0.665187

var(counts) = 0.442474

$BINS_j$	$FREQS_j$
234374	3
234375	582
234376	1123
234377	279
234378	13

Figure 6: 20 Minute Radiation Histogram, $Flux = 16.4 rad/hour$



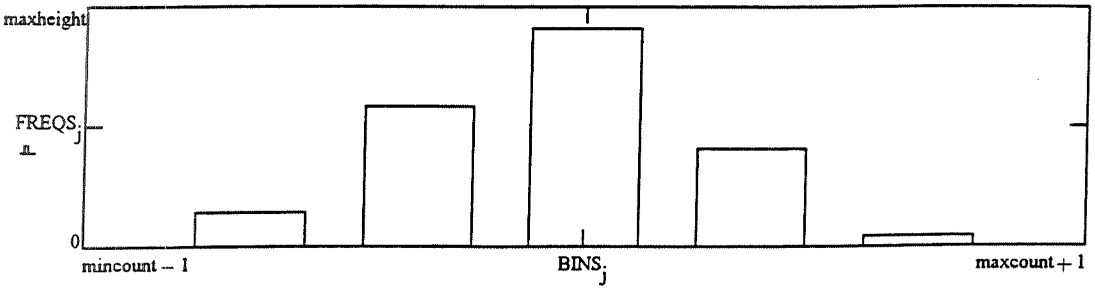
mean(counts) = 234371.91

stdev(counts) = 0.397366

var(counts) = 0.1579

$BINS_j$	$FREQS_j$
234371	64
234372	417
234373	19

Figure 7: 5 Minute Baseline Histogram



mean(counts) = 234373.8175

stdev(counts) = 0.887802

var(counts) = 0.788192

$BINS_j$	$FREQS_j$
234372	135
234373	561
234374	876
234375	390
234376	38

Figure 8: 20 Minute Radiation Histogram, $Flux = 187 rad/hour$

5 Conclusions

The ADC performed with virtually no change in output noise with a flux of ten times the peak proton flux in the SAA. At 100 times the SAA flux, the standard deviation of the output code increased slightly, but the performance was still very acceptable for many applications. The guard bars appeared to prevent latch-up effectively in a commercial epitaxial process.

6 Acknowledgements

The authors would like to thank Greg Zweigle for his valuable help in creating low noise circuit boards. This research was made possible by the NASA Space Engineering Research program which is funding the development.

References

[1] D. Nichols, J. Coss, R. Watson, H. Schwartz, R. Pease, "An Observation of Proton-Induced Latch-up," IEEE Transactions on Nuclear Science, Vol. 39, pp. 1654-1656, December 1992.

[2] L. Adams et al, "A Verified Proton Induced Latch-Up in Space" IEEE Transactions on Nuclear Science, Vol. 39, pp. 1804-1808, December 1992.

[3] T. Turflinger, M. Davey, "Understanding Single Event Phenomena in Complex Analog and Digital Integrated Circuits," IEEE Transactions on Nuclear Science, Vol. 37, pp. 1832-1838, December 1990.

[4] T. Turflinger, M. Davey, "Transient Radiation Test Techniques for High-Speed Analog-to-Digital Converters," IEEE Transactions on Nuclear Science, Vol. 36, pp. 2356-2361, December 1989.

[5] D. Thelen "An 18 Bit 50 kHz ADC for Low Earth Orbit," 4th NASA Symposium on VLSI Design, pp 2.5.1-2.5.11, October 1992.

[6] J. Robert, P. Deval, "A Second-Order High-Resolution Incremental A/D Converter with Offset and Charge Injection Compensation," IEEE J. of Solid-State Circuits, Vol. SC-23, pp. 736-741, June 1988.

[7] J. Robert, G. Temes, V. Valencic, R. Dessoulavy, P. Deval, "A 16-bit Low-Voltage CMOS A/D Converter," IEEE J. of Solid-State Circuits, Vol. SC-22, pp. 157-163, April 1987.

[8] B. Gordon, "Linear Electronic Analog/Digital Conversion Architectures, Their Origins, Parameters, Limitations, and Applications," IEEE Transactions on Circuits and Systems, Vol. CAS-25, pp. 391-418, July 1978.

[9] G. Landsburg, "A Charge-Balancing Monolithic A/D Converter," IEEE J. of Solid-State Circuits, Vol. SC-12, pp. 662-673, December 1977.

[10] J. Suter, J. Cloeren, J. Norton, D. Kusnierkiewicz, A. Koehler, "Simulation of Low-Earth-Orbit Radiation Environments with a 5 to 120 MeV Proton Cyclotron Beam Using a Proton Beam Modulator," IEEE Transactions on Nuclear Science, Vol. NS-34, pp. 1070-1075, August 1987

[11] Private Conversation with Walter Fowler, NASA GSFC, Sept. 29, 1993.

[12] E. Stassinopoulos, J. Raymond, "The Space Radiation Environment for Electronics," Proceedings of the IEEE, Vol. 76, pp. 1423-1442, November 1988

Use of Switched Capacitor Filters to Implement the Discrete Wavelet Transform

Kraig E. Kaiser and James N. Peterson
NASA Space Engineering Research Center for VLSI Design
University of Idaho
Moscow ID 83843

Abstract - **This paper analyzes the use of IIR switched capacitor filters to implement the discrete wavelet transform and the inverse transform, using Quadrature Mirror Filters (QMF) which have the necessary symmetry for reconstruction of the data. This is done by examining the sensitivity of the QMF transforms to the manufacturing variance in the desired capacitances. The performance is evaluated at the outputs of the separate filter stages and the error in the reconstruction of the inverse transform is compared with the desired results.**

1 Introduction

The wavelet transform is a powerful new tool in the area of signal analysis and decomposition. There are many applications where wavelet signal analysis is very beneficial. Some of these areas are computer vision, image compression, and speech recognition [3, 4, 5, 6]. Application-specific VLSI implementations of the wavelet transform could be very useful in areas such as these.

In this paper we examine using switched capacitor filters to implement the wavelet transform. First we shall take a look at the wavelet transform and how it relates to the Fourier transform. Next we show what types of filters are required to successfully implement the wavelet transform and the inverse transform. Then we proceed with conversion to switched capacitor filters and analyze how well the switched capacitor filters are able to perform the desired transforms when capacitance inaccuracies are taken into account.

2 Wavelets

The wavelet transform is essentially a multiresolution, or multiscale tool for signal analysis that includes both time and frequency information. It allows looking at more than just the frequency content of the signal, as is done commonly with the Fourier transform. By analyzing a signal using a sliding scale between time and frequency resolutions, the wavelet transform can readily identify transients and discontinuities in time as well as frequency content. But before discussing how the wavelet transform works, we should first briefly review its predecessor, the Fourier transform, which is fundamental in frequency analysis methods.

2.1 The Fourier Transform

The Fourier transform analyzes signals under the assumption that the original time-domain function is periodic in nature. It uses complex sinusoids which have an infinite duration as the basis functions. The Fourier transform of a continuous time signal $x(t)$ is:

$$X_F(\omega) = \int_{-\infty}^{\infty} e^{-j\omega t} x(t) dt \tag{1}$$

A common difficulty with the Fourier transform is that, because the basis functions have an infinite duration, any information about transients in the signal is spread out over the whole frequency axis and is lost. This problem has been previously addressed by introducing a windowed complex sinusoid as the basis function [1, 2]. This leads to the doubly indexed windowed Fourier transform or the short-time Fourier transform (STFT):

$$X_F(\omega, \tau) = \int_{-\infty}^{\infty} e^{-j\omega t} w(t - \tau) x(t) dt \tag{2}$$

where $\omega()$ is an appropriate window function. In other words, $X_{WF}(\omega, \tau)$ is the Fourier transform of $x(t)$ windowed with $w()$ shifted by τ. The STFT essentially uses modulated versions of the window function as the basis functions.

The major advantage of the STFT is that if a signal has most of its energy in a given time and frequency interval, then its STFT will be localized to that region, and will be close to zero in the time and frequency intervals where the signal has little energy. Yet the STFT is still limited because it uses the same window for all frequencies. This means that the resolution of the analysis is the same for all locations in the time-frequency plane.

2.2 The Wavelet Transform

However, with the wavelet transform we are able to vary the window function in the time-frequency plane. This makes it possible to trade resolution in time for resolution in frequency. Using a short duration basis function allows isolation of discontinuities in signals by giving increased time resolution, while a longer-duration basis function enables greater frequency resolution. The wavelet transform has high frequency basis functions with short time duration and low frequency basis functions that are of longer time duration.

The basis functions are obtained by translating and either dilating or contracting the prototype wavelet $\Psi^{a,b}(t)$ which is defined as:

$$\Psi^{a,b}(t) = \frac{1}{\sqrt{a}} \Psi\left(\frac{t-b}{a}\right) \tag{3}$$

where a is a positive real number and b is a real number, and $\Psi()$ is the chosen wavelet function above. By increasing or decreasing a, the wavelet is expanded or contracted respectively, and by varying b it is shifted in time. The continuous wavelet transform is:

$$X_W(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} \Psi\left(\frac{t-b}{a}\right) x(t) dt \tag{4}$$

2.3 The Discrete Wavelet Transform

In order to obtain the discrete form of the wavelet transform we set $a = a_0^m$ and $b = na_0^m b_0$, where $a_0 > 1, b_0 \neq 0$, and m and n are integers. Substituting into (3) and (4) we obtain the following equations for the discrete wavelet and the discrete wavelet transform [11].

$$h_{mn}(t) = a_0^{-m/2} \cdot h(a_0^{-m}t - nb_0) \tag{5}$$

$$X_W(m,n) = a_0^{-m/2} \int_{-\infty}^{\infty} h(a_0^{-m}t - nb_0)x(t)dt \tag{6}$$

The discrete wavelet transform can be implemented using digital filters [7]. Special types of filters are required in order to be able to reconstruct the signal and perform the inverse transform. These filters are known as Quadrature Mirror Filter (QMF) pairs. A QMF pair consists of two filters which have amplitude responses that are mirror images about a certain frequency, and have phase responses that differ by $\pi/2$ radians at all frequencies.

When performing the wavelet transform the dividing point for the QMF pair is at one fourth the sampling rate. The outputs of the QMF pair are decimated by a factor of two (every other sample is discarded) which yields two signals with half the time resolution. The low-pass filter output is fed into another identical QMF filter pair. This operation is then repeated in a tree structure as is shown in Figure 1. This results in a group of signals which divide the spectrum of the original signal into octave bands. Each of these successive bands have reduced time resolution and increased frequency resolution.

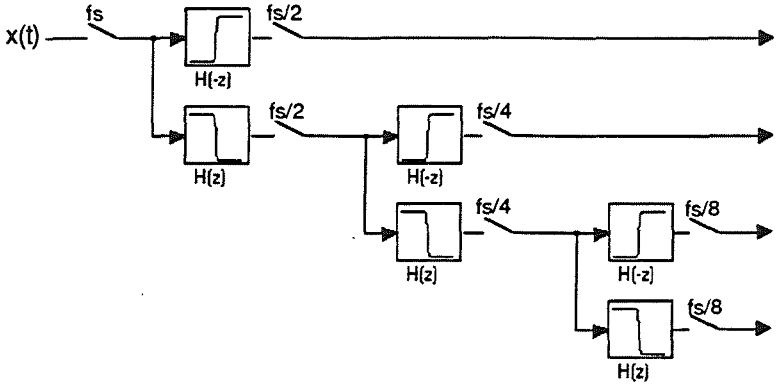


Figure 1: QMF tree structure for wavelet transform.

The inverse wavelet transform is performed using a similar tree structure, shown in Figure 2. Prior to each filter the input signal is up-sampled by two. This is done by inserting zeros between each of the data points. The signals are then passed through the filters again to remove the aliasing that is added by the up-sampling.

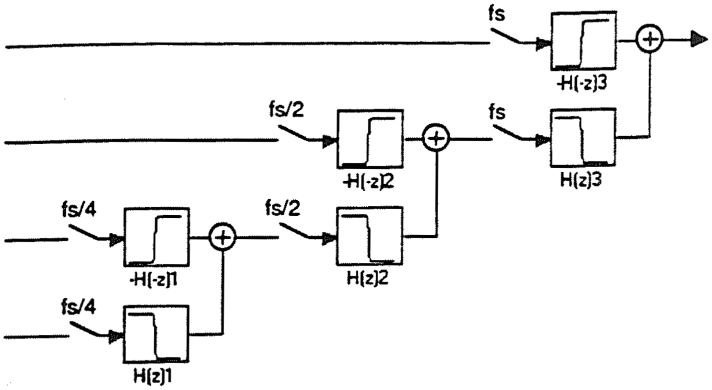


Figure 2: QMF tree structure for inverse wavelet transform.

3 Switched Capacitor Implementation

3.1 Low Q Biquadratic Conversion

The QMF filters were designed using the method presented in [10] and then converted to switched capacitor (SC) form by using the method presented in [8]. The conversion method uses biquadratic sections of SC circuitry to implement second order blocks of the filter equations. The circuit shown in Figure 3, known as a low Q biquadratic (LQBQ) section, is used to implement an equation of this form:

$$H(z) = -\frac{a_2 z^2 + a_1 z + a_0}{b_2 z^2 + b_1 z + 1} \tag{7}$$

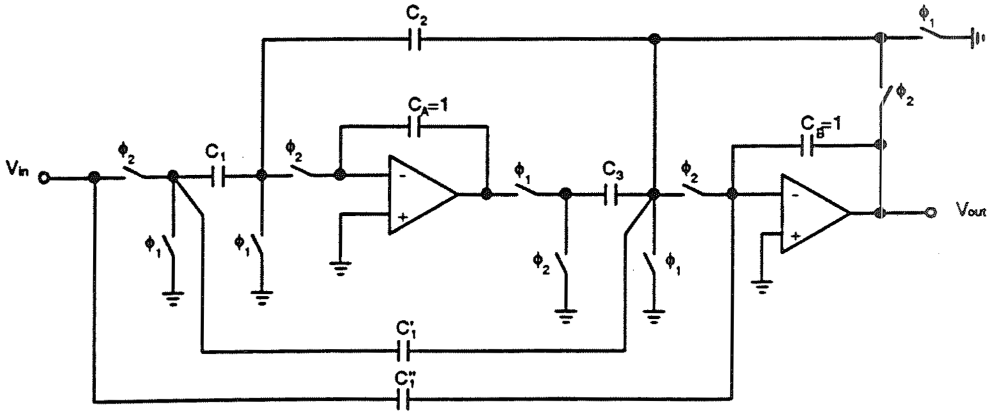


Figure 3: Low Q Biquadratic Switched Capacitor Circuit.

Analysis of the circuit in Figure 3 results in the following exact z-domain transfer function for the circuit:

$$H(z) = \frac{(C_1' + C_1'')z^2 + (C_1C_3 - C_1' - 2C_1'')z + C_1''}{(1 + C_4)z^2 + (C_2C_3 - C_4 - 2)z + 1} \tag{8}$$

By matching the coefficients in (7) and (8) it is possible to calculate the precise values for the capacitances that will result in an exact realization of the $H(z)$ given in (7).

3.2 Capacitance Ratio Error

The various C's in (8) represent ratios to the unit capacitances C_A and C_B. When these circuits are actually manufactured there will be some error (ε) in the ratio of the capacitances. This error is typically on the order of ±0.1 percent for the ratio [9].

For this paper we are primarily interested in how this error affects the performance of the QMF filters and their use in the discrete wavelet transform. The effect of the ratio error on the filter coefficients can be studied by looking at the following equations which relate the capacitance ratios to the coefficients.

$$a_0 = C'' \tag{9}$$

$$a_1 = C_1 C_3 - C_1' - 2C_1'' \tag{10}$$

$$a_2 = C_1' + C_1'' \tag{11}$$

$$b_1 = C_2 C_3 - C_4 - 2 \tag{12}$$

$$b_2 = C_4 + 1 \tag{13}$$

Now by varying each of the C values (each might vary independently of the others), the effect of the manufacturing tolerance variations on the filter coefficients can be determined. A complete study of this phenomena could address possible statistical correlations among the capacitance ratio variances, as well as the final filter characteristics from a probabilistic framework. However, for the analysis presented herein, we will use a worst-case methodology.

3.3 Worst Case Analysis

If we apply the ratio error to each of the capacitance ratios in (9) - (13), we can evaluate the worst case effect the error will have on each of the coefficients. The following equations were obtained by examining the worst case effect of the ratio error on the coefficients. They were derived by assuming that the errors would combine in the largest possible fashion, resulting in the greatest effect on the coefficients. Using these equations we can calculate the maximum effect for each of the coefficients in equation (7).

$$a_0' = a_0(1 \pm \varepsilon) \tag{14}$$

$$a_1' = a_1(1 \pm \varepsilon)^2 \pm \varepsilon(1 \pm \varepsilon)(a_0 + a_2) \tag{15}$$

$$a_2' = a_2(1 \pm \varepsilon) \tag{16}$$

$$b_1' = b_1(1 \pm \varepsilon)^2 \pm \varepsilon(1 \pm \varepsilon)(b_2 + 1) \tag{17}$$

$$b_2' = b_2(1 \pm \varepsilon) \pm \varepsilon \tag{18}$$

In order to see how the changes in the coefficients affects the performance of the filters we need to decide how to apply the changes. Each of the coefficients can either be increased or decreased according to equations (14) - (18). There are many different ways of altering the coefficients which effect the performance at different frequencies. For our worst case analysis we are only going to examine two possible changes for the filter coefficients. This does not necessarily result in worst case variations over all frequencies of the filter frequency response magnitude and phase, but this does serve to illustrate the impact of the manufacturing error tolerances.

$\varepsilon = +0.001$	Increase all of the numerator coefficients and decrease all of the denominator coefficients.
$\varepsilon = -0.001$	Decrease all of the numerator coefficients and increase all of the denominator coefficients.

There are now four different possibilities for applying the error to the QMF pairs. A positive or negative error can be applied to either the low-pass (ε_L) or high-pass (ε_H) filter.

4 Performance Analysis

4.1 Combined Filter Response

To analyze the performance of the filters we first considered how the error applied to the filter coefficients affected the response of the combined filters. The low-pass filter is $H(z)$, the high pass filter is $H(-z)$, and the response we examined is simply $H(z) + H(-z)$. The combined transfer function response was calculated for all four of the possibilities being evaluated. Figure 4 shows the two responses where the low-pass error (ε_L) is the same as the high-pass error (ε_H), and Figure 5 shows the responses where $\varepsilon_H = -\varepsilon_L$.

The desired frequency response for the complete decomposition/reconstruction system would be an all-pass filter having unity gain over the complete range of frequencies. Figure res 4 and 5 show the inability of the original QMF filter set to achieve this (the solid line in each figure), and also the changes in the frequency response due to the variations in the capacitance ratios. This shows that the ratio errors of 0.1% can create up to 2-3% change in the magnitude response characteristic of the complete wavelet decomposition and reconstruction signal processing.

4.2 Single Level Reconstruction

The next method used to analyze the performance is to look at how well the filters are able to reconstruct the input signal after splitting it into two bands. The system used to test this is shown in Figure 6. It is very difficult to obtain a frequency response for a system such as this, due to the non-linearities introduced by down-sampling and then up-sampling the signals. It is also difficult to evaluate the performance for an input containing multiple frequencies

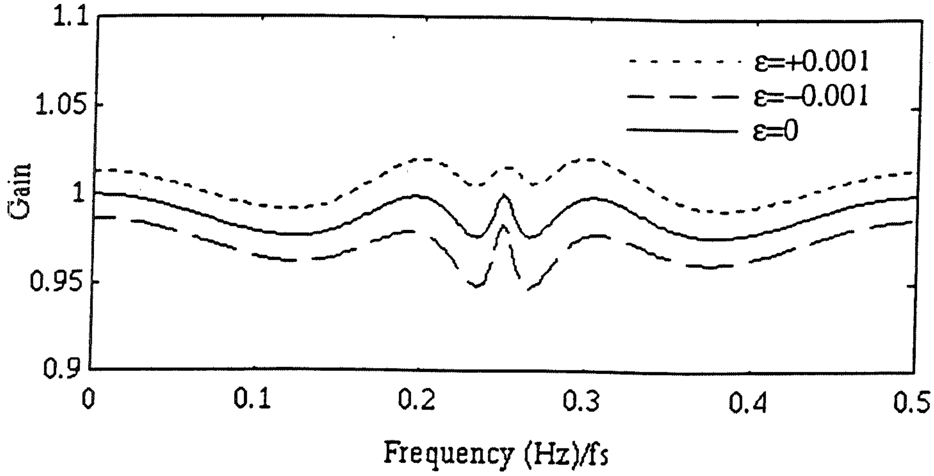


Figure 4: Frequency response curves for $\varepsilon_L = \varepsilon_H = \varepsilon$.

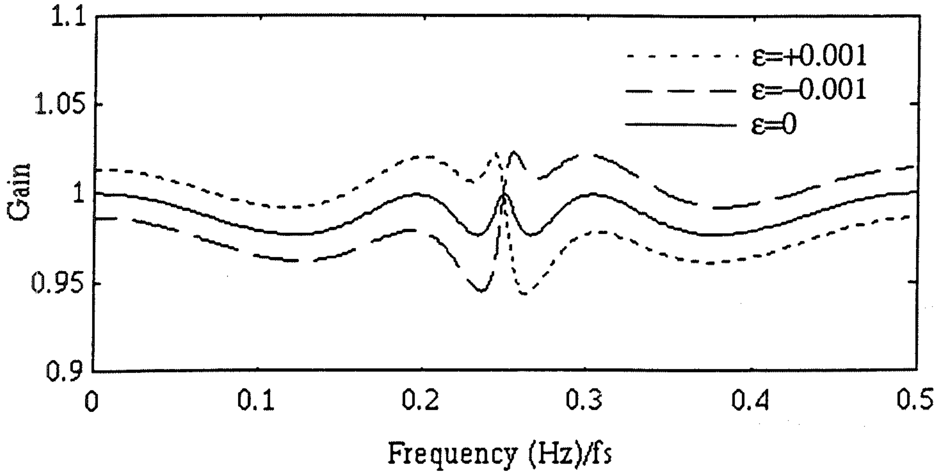


Figure 5: Frequency response curves for $\varepsilon_L = \varepsilon$ and $\varepsilon_H = \varepsilon$.

because the different frequencies will be have different phase distortions. However, it is quite easy to analyze the performance at a particular frequency by simply comparing the system output to the input.

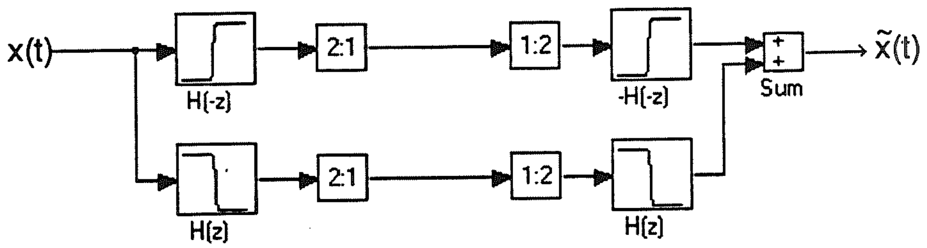


Figure 6: Single level reconstruction system.

Our analysis looked at the performance at three frequencies: $f_S/4$, $f_S/8$ and $3f_S/8$. These frequencies are positioned in the transition band, the middle of the low band, and the middle of the high band respectively. For an index to measure the performance, we used the sum of the squared differences between the input and the output for one hundred sample points.

Errors		**Frequencies**		
ε_L	ε_H	$f_S/8$	$f_S/4$	$3f_S/8$
0.000	0.000	0.8930	2.239E-4	0.8930
0.001	0.001	0.8226	0.2771	0.8135
0.001	-0.001	0.8226	0.6491	1.0620
-0.001	-0.001	1.0457	0.2696	1.0620
-0.001	0.001	1.0457	0.6532	0.8134

Table 1: Analysis Results for Single Level Reconstruction

From the results listed in Table 1 you can see that the performance is affected differently at each of the different frequencies. At $f_S/4$ the worst performance is when ε_L is negative and ε_H is positive, but this is also where the best performance at $3f_S/8$ is obtained. It can also be seen that the performance was increased for some of the error combinations.

4.3 Two Level Reconstruction

For further analysis the performance of the QMF filters was tested using a system where the input signal is passed through two levels of QMF filtering. In the test system shown in Figure 7, the signal is split into three different bands and then reconstructed. The performance in this system was measured the same way as the performance for the single level reconstruction, by looking at the sum of the squared differences between the input and the output. This system was also evaluated at the same three frequencies.

The results of the analysis of this system for each of the four error combinations and for the nominal case are listed in Table 2. It can be seen from these results that the amount of error in the output increased from what it was with only one level of decomposition. This

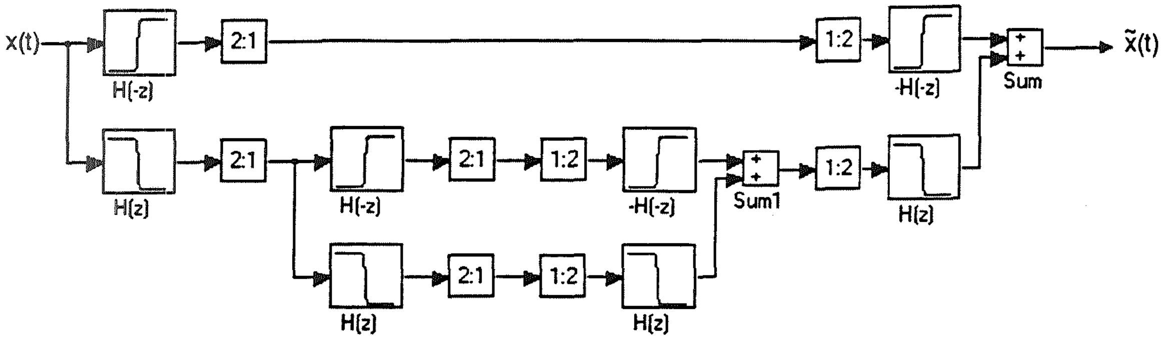


Figure 7: Two level reconstruction system.

Table 2: Analysis Results for Two Level Reconstruction

Errors		**Frequencies**		
ε_L	ε_H	$f_S/8$	$f_S/4$	$3f_S/8$
0.000	0.000	1.5026	0.0032	0.8940
0.001	0.001	1.0094	0.4058	0.8145
0.001	-0.001	1.4995	0.4635	1.0632
-0.001	-0.001	2.8572	0.3874	1.0630
-0.001	0.001	1.5259	0.5654	0.8143

increase, however, is not uniquely due to the error in the coefficients. The amount of error in the original filter's output at $f_S/8$ has also greatly increased. These results also show that the effect of the error is quite different at the different frequencies.

5 Conclusions

Results demonstrate that switched-capacitor filters can successfully implement the discrete wavelet transform using QMF pairs. These results, obtained by analyzing the performance of the filter systems at certain frequencies and by examining the combined transfer function for the QMF pairs, show that the error in the capacitance ratio can have an effect on the transform performance. But the effect is fairly small, only slightly more error is introduced than what is already contributed by the imperfections in the original filters. A more thorough analysis of the effects of the ratio error could be performed by studying the performance across the whole frequency range and by considering different means of applying the error to the coefficients.

References

[1] James A. Cadzow, *Foundations of Digital Signal Processing and Data Analysis*, Macmillan Publishing Company, 1987, pp. 104-105.

[2] Alan V. Oppenheim and Ronald W. Shafer, *Discrete-Time Signal Processing*, Prentice-Hall Publishers, 1989, pp. 713-730.

[3] R. A. DeVore, B. Jawerth, and B. J. Lucier, "Image Compression Through Wavelet Transform Coding," IEEE Trans. on Information Theory, Vol. 38, pp. 719-746, March 1992.

[4] D. M. Healy and J. B. Weaver, "Two Applications of Wavelet Transforms in Magnetic Resonance Imaging," IEEE Trans. on Information Theory, Vol. 38, pp.840-859, March 1992.

[5] S. Mallat, "Multifrequency channel decompositions of images and wavelet models," IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. 37, pp. 2091-2110, Dec. 1989.

[6] X. Yang, K. Wang, and S. A. Shamma, "Auditory Representations of Acoustic Signals", IEEE Trans. on Information Theory, Vol. 38, pp. 824-839, March 1992.

[7] M. Vetterli and C. Herley, "Wavelets and Filter Banks: Theory and Design," IEEE Trans. on Signal Processing, Vol. 40, pp. 2207-2232, Sept. 1992.

[8] Roubik Gregorian and Gabor C. Temes, *Analog MOS Integrated Circuits for Signal Processing*, John Wiley & Sons, Inc., 1986, pp. 280-296.

[9] Roubik Gregorian and Gabor C. Temes, *Analog Integrated Circuits for Signal Processing*, John Wiley & Sons, Inc., 1986, pg. 5.

[10] Paul C. Millar, "Recursive Quadrature Mirror Filters - Criteria Specification and Design Method", IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-33, pp. 413-420. April 1985.

[11] Ali N. Akansu and Richard A. Haddad, Multiresolution Signal Decomposition, Academic Press, Inc., 1992.

Silicon Retina for Optical Tracking Systems

K. Strohbehn and R.E. Jenkins
Space Department
Johns Hopkins University
Applied Physics Laboratory
Johns Hopkins Rd.
Laurel MD 20723
301-953-6000, 301-953-6670 (FAX)
aleph0@aplcomm.jhuapl.edu

X. Sun and A.G. Andreou
Department of Electrical and
Computer Engineering
Johns Hopkins University
3400 North Charles Street
Baltimore MD 21218-2686.
410-516-7076
andreou@olympus.ece.jhu.edu

Abstract - **There are a host of position sensors, such as quadcells and CCDs, which are candidates for detecting optical position errors and providing error signals for a mirror positioning loop. We are developing a novel, very high bandwidth, biologically inspired position sensor for optical position tracking systems. We present recent test results and design issues for the use of biologically inspired silicon retinas for spaceborne optical position tracking systems.**

1 Introduction

The use of optical point to point communication links is likely to be increasingly prevalent in both spacecraft communication links as well as terrestrial communications. In space the advantages of optical communications are the high bit rates achievable for a given power with a highly directional antenna that is both light weight and compact. This type of consideration is especially important for the low cost microsats which will undoubtedly proliferate. Initial experiments have demonstrated a one megabit per second optical laser link between a ground station in Japan and a geostationary satellite [1]. The GOPEX laser communications demonstration with, ironically, the Galileo spacecraft demonstrated the ability to open loop detect a ground station laser at a distance of up to six million kilometers (i.e. the ground station pointed at the spacecraft accurately enough to put the laser spot in the field of view of the camera).

Although the GOPEX demonstration was able to point the groundstation laser at the Galileo spacecraft with enough accuracy to keep it in the field of view of a CCD camera for it to be detected, for optical communications links with satellites or earth ground stations, it is necessary to track the laser beam and keep it in the detector field of view. For this reason the laser beam is often amplitude modulated (chopped) at a frequency of some tens of kHz to serve as a beacon that is discernable from other bright point sources, such as stars. The laser spot must be positioned on the data demodulating detector with great stability to maximize the signal to noise ratio and minimize the amount of broadcast power required for a given data rate. This is an especially important consideration with deep space communications.

There are a host of position sensors, such a quadcells and CCDs, which are candidates for detecting a laser beacon and providing error signals for a mirror positioning loop. Most of the current generation of sensors tend to place serious bandwidth constraints on the mirror position loops. Sensors based on quad cells and lateral effect diodes are inherently slow because of very large junction capacitance, whereas CCD arrays are constrained by read out delay. We are developing a novel, very high bandwidth, biologically inspired position sensor for optical laser communication beacon tracking systems. The development of this sensor is analogous to providing "beacon sensitive" eyes for autonomous microsats and ground stations. Moreover, it is worth considering that laser beacon tracking is applicable to a host of other applications which could utilize laser beacons such as corner-cube tracking, navigation and even power transmission or laser propulsion of deep space probes.

Figure 1 shows the architecture of a sensor chip which we refer to as a video computation (VC) chip. A focal plane array detector is constructed which replicates each pixel signal to form instantaneous x and y projection signals and a scanned out video signal. The original VC chip employed the pixel shown in the expanded view in Figure 1. This pixel consisted of a phototransistor and a simple current mirror as shown. This sensor was developed as a sunspot (or pore) tracker for stabilizing solar images acquired with a solar vector magnetograph [4], and is in routine use at the National Solar Observatory at Sacramento Peak in Sunspot,NM.

We have found that the tracker works well on spots near the center of the solar disk, but several improvements are desirable. First, position sensitivity is a function of both the spot contrast and the spot size in the field of view. Even a high contrast spot that is small relative to the field of view produces a weak position signal. This is undesirable because the effective loop gain of the tracking system becomes a function of spot size, contrast, position on the solar disk, and overall illumination level. Thus, to obtain optimum performance, it is necessary to readjust the loop gain for each new spot to be tracked. For very small or low contrast spots, it may be difficult or impossible to achieve useful tracking bandwidths because the detector is not very sensitive in those cases. Simply increasing the gain of the servo amplifier is only a partial solution, limited by the available gain, and the introduction of noise into the servo loop. Second, illumination gradients near the solar limb produce DC offsets in the position error signal which depend on the spot being tracked. For large high-contrast spots this effect is not a significant problem, but gradients that are large relative to the position error signal can cause unacceptable decreases in the system dynamic range.

2 Retinal Pixels

Some improvements in these two areas can be obtained by replacing the simple focal plane used in the VC architecture of Figure 1 with pixels constructed of PNP phototransistors that are interconnected in a fashion which is analogous to the outer plexiform of biological retinas as shown in Figure 2.

This focal plane can be shown to provide near optimal edge detection and noise smoothing, and removes brightness gradients from the incident image. In addition the output levels are normalized independent of the incident light level. As in Figure 1 simple current mirrors replicate the individual pixel responses to form x and y projections of the image for continuous time centroid position computation, as well as a scanned out video signal. The result is a photodetector array which has similar sensitivity to a rapidly scanned CCD, but does not require a read out for computing the instantaneous image position. The VC retina is scanned out only as a matter of convenience for system set up, or if other image computations are to be performed. For tracking loops the elimination of the readout step results

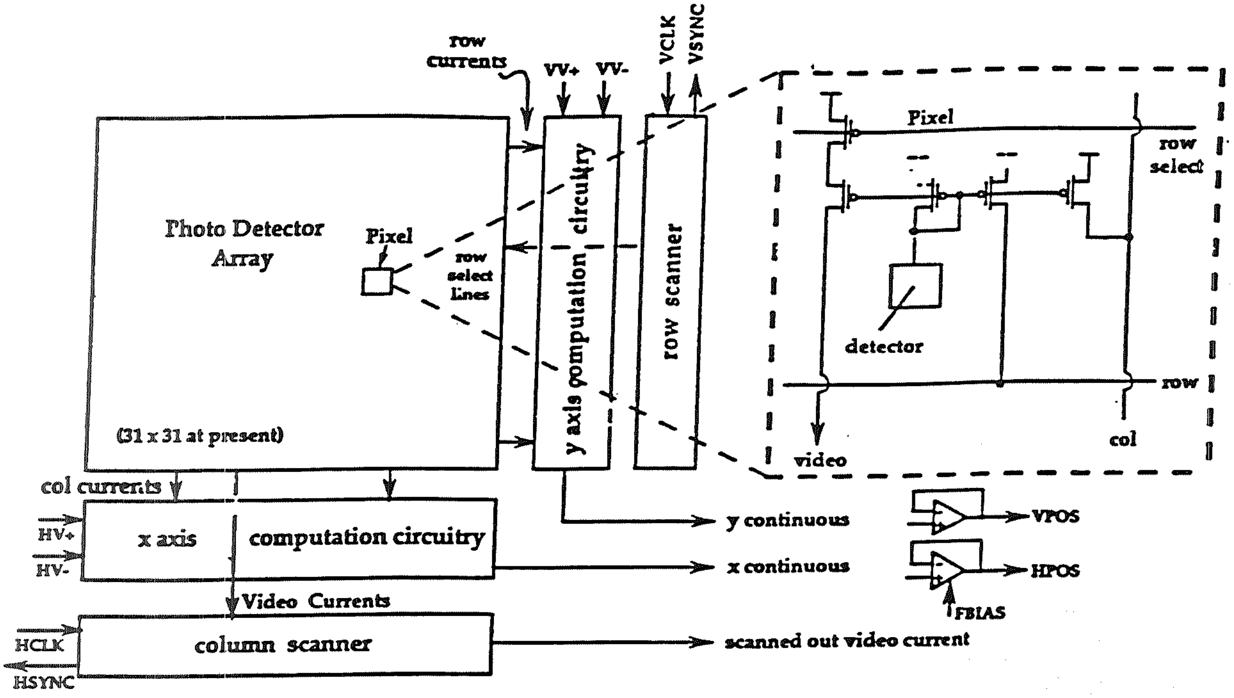


Figure 1. Video Computation Chip Computing Organization. A detector array which may consist of simple phototransistors or any electronic "retina" which provides continuous time x and y signals and a scanned video signal. Global x and y computations, such as image centroid, are performed at the periphery.

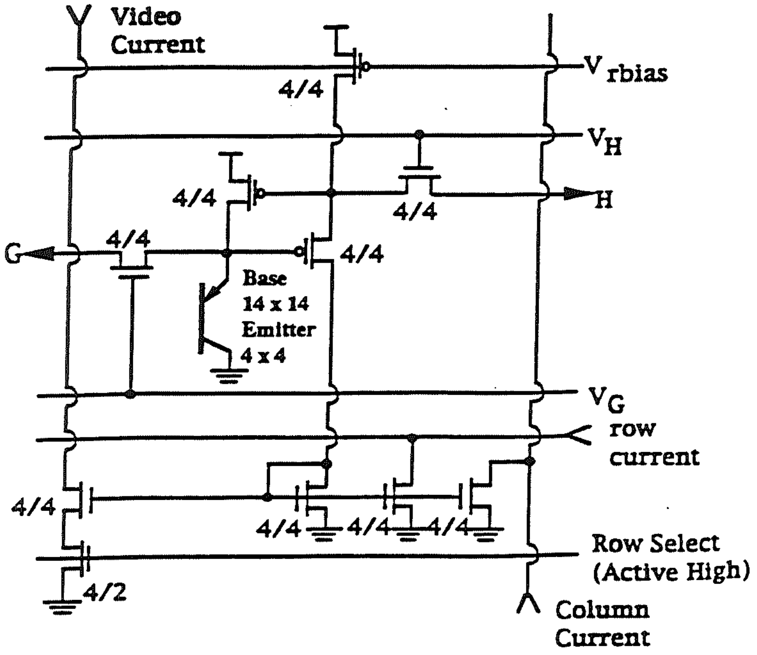


Fig 2. VC Pixel Schematics

in greatly improved detector bandwidth, greatly reduced complexity, and much lower power consumption.

To appreciate how this focal plane accomplishes these image processing tasks, suppose that the image brightness $B(x,y)$ is given by

$$B(x,y) = I(x,y)s(x,y) \tag{1}$$

where $I(x,y)$ is the lowpass illumination which results in the overall image brightness level and gradients, and $s(x,y)$ is the highpass signal image consisting of the interesting edges in the image. A homomorphic filter performs a linear highpass spacial filter operation on the logarithm of $B(x,y)$ which passes only $\log(s(x,y))$, and then outputs the exponential of the filtered result

$$\begin{aligned} f(x,y) &= \exp(hpf[\log B(x,y)]) \\ &= \exp(hpf[\log I(x,y) + \log s(x,y)]) \\ &= \exp(\log s(x,y)) = s(x,y) \end{aligned} \tag{2}$$

where $hpf[\,]$ denotes a highpass spacial filtering operation. The focal plane retina shown in Figure 2 performs this homomorphic operation very compactly, and normalizes the output levels independent of illumination.

Referring to Figure 2, the logarithm of the input current is generated naturally as the gate-source voltage of an MOS transistor operated in subthreshold. The interaction of the two, single-transistor, diffusive spreading networks results in a nonlinear "Mexican-Hat" highpass spacial filtering operation, $hpf[\,]$, like the one found in biological retinas, The relative degree of smoothing versus edge sensitivity can be controlled by the control voltages VG and VH. The output currents are exponential functions of the highpass filtered spreading network node voltages G if the devices are biased in subthreshold. This silicon retina is discussed in more detail in [6].

We designed and fabricated a 50 by 50 pixel chip with the VC architecture of Figure 1 and the retinal focal plane array just described on a 4.6 by 6.8 mm die. We refer to this chip as the RVC chip. The focal plane was a rectilinear array (as opposed to hexagonal) tiled with a 77 micron by 77 micron unit pixel. A schematic of the unit pixel is shown in Figure 2, and a MAGIC layout of the pixel is shown in Figure 3. Figure 4 is a die photograph of the RVC chip. The architecture of Figure 1 is evident in Figure 4 where the 50 by 50 array occupies the central portion of the active area, and centroid position computation and video scan circuitry is located at the periphery of the focal plane array. The video scanning circuitry is similar to that described in [5], except that each video column is amplified prior to horizontal scanning. The large devices at the lower right are the video output drivers.

3 Test Results

We received 12 packaged parts from MOSIS and of these two parts appeared to have damaged spreading networks which was possibly caused by improper handling (The VG and VH bias inputs are unprotected, because they must be set higher than Vdd). Initially we tested the RVC chips for retinal response, and for position sensitivity.

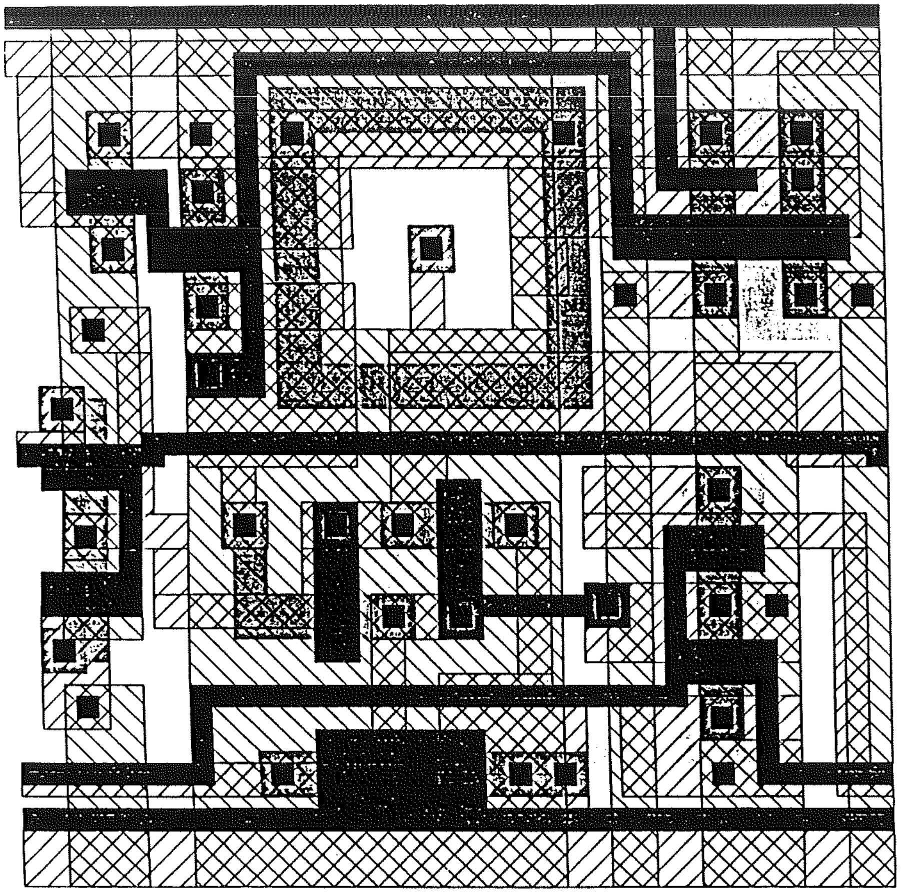


Figure 3. Retina Pixel Layout.

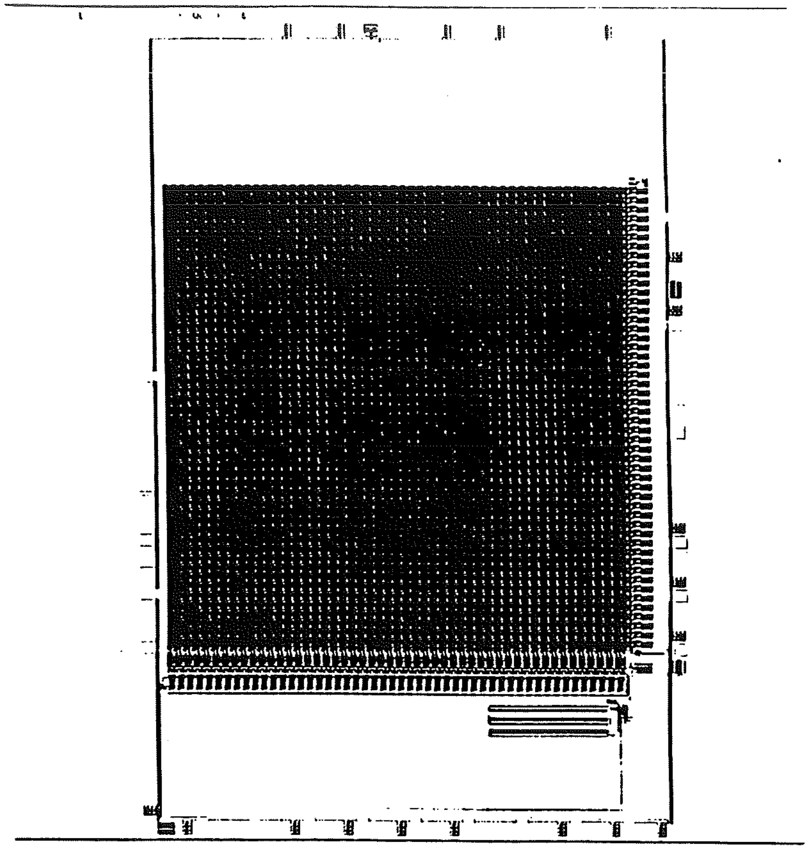


Figure 4. RVC Silicon Retina Chip.

4 Retinal Response

Figures 5 and 6 are typical video outputs from the RVC chip in which the Mexican-Hat edge response is evident. Figure 5 shows the response to a black spot on a white background which is intended to simulate a sunspot image., and Figure 6 shows the response to several white spots on a black background which is intended to simulate solar granulation. Both Figures 5 and 6 were intentionally acquired in the presence of a severe illumination gradient introduced with a desk lamp. The response to uniform lighting was virtually identical as expected. The average output level changed by about 4 percent over at least 6 orders of magnitude (from lens cap on, to a bright penlight at full aperture). We concluded that 10 of the test parts function as retinas. The space constants of the two spreading networks were adjustable with potentiometers so that an appropriate balance between smoothing and edge detection could be selected for a given scene.

5 Position Response

The RVC chip centroid circuitry was identical to the VC chip design which was based on [7]. Unfortunately, this was a design error, because the retinal focal plane is insensitive to the image centroid. This is intuitively obvious because an image centroid is a low frequency property of the image, and the retinal highpass response is insensitive to lowpass illumination properties such as centroids. This can also be shown mathematically by considering a function f(x,y) with a constant mean response such as produced by the retina in equation 2.

$$\sum_{x,z} f(x,y) = Irb = \text{the retinal bias current} \tag{3}$$

Since the centroid of a constant is zero by definition of our coordinates, we can take $Irb = 0$ for analysis purposes. The x centroid is given by

$$x_c = \sum_x x \sum_y f(x,y) \tag{4}$$

If we translate the distribution f(x,y) by say p in the x direction we get by change of variables

$$\sum_y x f(x+p,y) = \sum_y u f(u,y) - p \sum_y f(u,y) = \sum_y x f(x,y) \tag{5}$$

so that the centroid is invariant to a translation of f(x,y) as long as f(x,y) is included in the field of view [2]. As predicted the RVC chips were very insensitive to spot position. Therefore, we digitized the video output of the RVC chips with a framegrabber to evaluate four alternative position sensing schemes.

The retinal response to an image centroid can be recovered by simply taking the absolute value of the retinal response f(x,y) relative to the bias level Irb.

$$x_c = \sum_x x \sum_y |f(x,y) - Irb| \tag{6}$$

and

$$y_c = \sum_y y \sum_x |f(x,y) - Irb|$$

which we refer to as an absolute value centroid (AVC).

Another method of computing position is to minimize the sum of absolute differences between the instantaneous projection of f(x,y) and a reference projection r(x,y).

$$\begin{aligned} x_c &= \sum_x [|\sum_y f(x+1,y) - r(x,y)| - |\sum_y f(x-1,y) - r(x,y)|] \\ y_c &= \sum_y [|\sum_x f(x,y+1) - r(x,y)| - |\sum_y f(x,y-1) - r(x,y)|] \end{aligned} \tag{7}$$

This scheme is similar to block motion algorithms used in video compression, and has properties similar to correlation, but is easier to compute. We refer to the scheme described by equation 7 as Absolute Difference Correlation (ADCor).

In equations 6 and 7 the summations are over the whole focal plane array, as in the VC chip, and we refer to this as full projection. Another variation is to only use the central cross of pixels so that edges don't get washed out. We call this cross projection and equations 6 and 7 become

$$\begin{aligned} x_c &= \sum_x x|f(x,0) - Irb| \\ y_c &= \sum_y y|f(0,y) - Irb| \end{aligned} \tag{8}$$

and

$$\begin{aligned} x_c &= \sum_x |f(x+1,0) - r(x,0)| - |f(x-1,0) - r(x,0)| \\ y_c &= \sum_y |f(0,y+1) - r(0,y)| - |f(0,y-1) - r(0,y)| \end{aligned} \tag{9}$$

for cross projection where the central cross is defined by x or y = 0.

We evaluated the four schemes defined by equations 6 through 9 for two types of imagery. In the first case we want to improve sunspot tracking in the presence of illumination gradients so we used imagery similar to Figure 5. In the second case we want to explore schemes for analog correlation tracking of solar granulation so we used imagery similar to Figure 6.

Figure 7 displays the position response for various centroiding schemes with both full and cross projection for sunspot images. Cross projection is clearly superior to full projection for centroiding with a retina. Cross projection results in much higher sensitivity and lower noise than full projection. This is because the edges don't get washed out in cross projection, and there is less edge noise included in the computation.

We found that centroiding was very insensitive to granulation image positions, unless the image scale is changed so that only one granule dominates the field of view. The danger

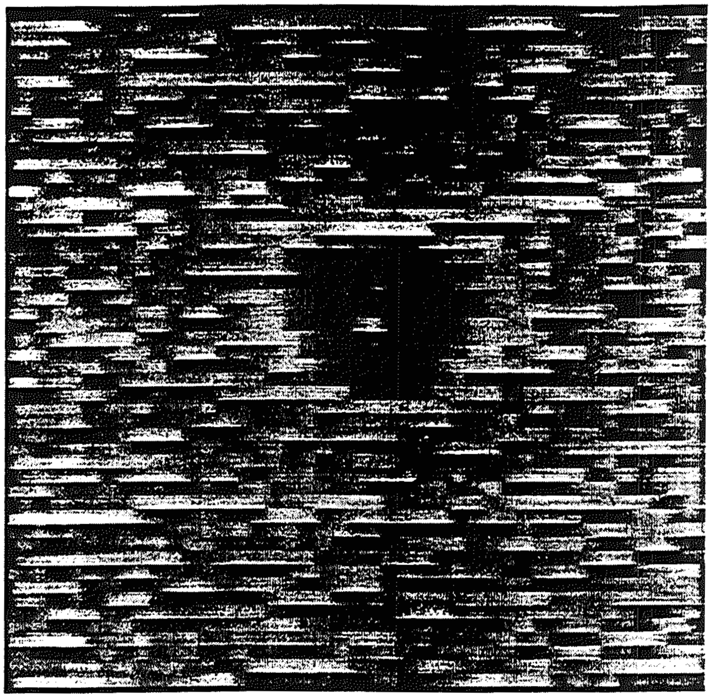


Figure 5. Retinal response to a dark spot on light background.

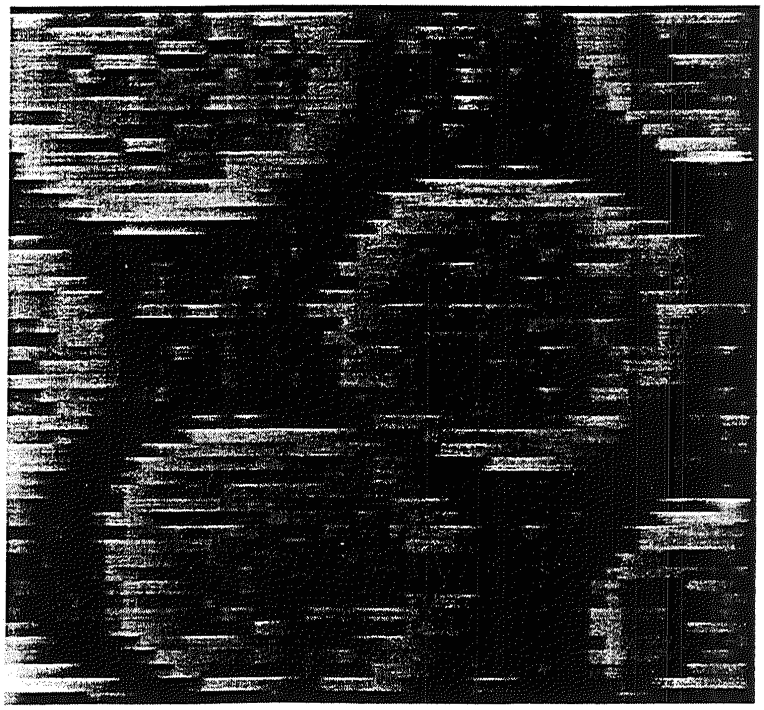


Figure 6. Retinal response to light blobs in dark background.

in this case is that noise in the servo loop might cause a jump that results in a shift from tracking one granule to another adjacent granule.

Figure 8 compares the AVC cross projection response with a best fit straight line response. Observe that there is some non-monotonic response due to retina mismatch effects. This is undesirable because it produces a jitter in the tracking servo that we term non-uniformity noise. Circuit simulation indicates that the current conveyor transistors in Figure 2 and the PNP phototransistor contribute equally to this mismatch. The contribution of the spreading transistors is much less than the conveyor or detector transistors because of averaging in the spreading networks.

If we bias the spreading networks off, then we can measure the mismatch of the retina bias transistor and the output mirrors. Figure 9 shows the effect of turning the spreading networks off. The bias/mirror mismatch is 1.7 percent which indicates very good matching of these small 4 by 4 micron devices. Measurements on the VC chip indicate a mismatch of around 7 percent for the phototransistors. Since the current conveyor transistors are probably matched similarly to the mirrors, we conclude that the retina mismatch shown in Figure 9 is dominated by the detector mismatch. The detector variation is enhanced by the highpass homomorphic filtering of the retina, and this defines the lowest contrast edges that can be reliably detected by this type of retina.

We found that cross projection is superior to full projection for absolute difference correlation also. Again, this is because the edges are not washed out. Correlation is more sensitive to granulation position than to spot position because there are more signal edges in granulation. Thus, centroiding is more appropriate to use for spot tracking, and correlation is more appropriate to use for granulation tracking.

Figure 10 compares the absolute difference correlation response with a best fit straight line response. We expect the s-curve type roll off around plus or minus one pixel of shift, because in this case we are only comparing shifts of plus or minus one pixel. We also see that the correlation response is monotonic for small shifts. For larger pixel shifts (not shown in Figure 10) the correlation response exhibits a non-monotonic response similar to the centroid response. Restricting the track loop to small shifts results in lower noise; however, there is a danger that the loop can lose lock briefly due to residual noise, and then relock at a different track point. The servo loop must be very tight to prevent this.

The VC chip has demonstrated bandwidths of 40 kHz at the illumination levels found in our vector magnetograph [2]. The bias current for the position sensing circuitry in the VC chip is dependent on the illumination level, and this limits the bandwidth in the VC chip. The RVC chip has another source of delay which can further limit its bandwidth. The dynamic response of the spreading networks is determined by the bias settings VG and VH. These can produce a range of bandwidths from tenths of Hz up to bandwidths similar to the VC chip. We were unable to measure the upper bandwidth limit on the RVC chip since the position sensing circuitry was so insensitive.

6 Discussion

In most of the applications mentioned in the introduction illumination gradients are not an issue. We conclude from this test data that in these applications it is preferable to employ

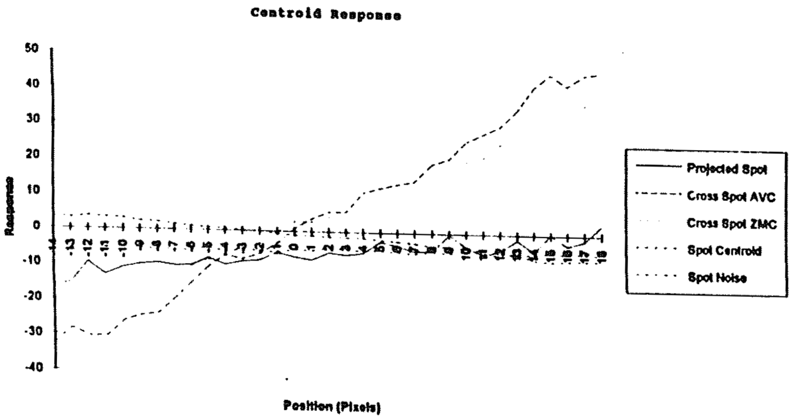


Figure 7. Comparison of spot centroiding schemes.

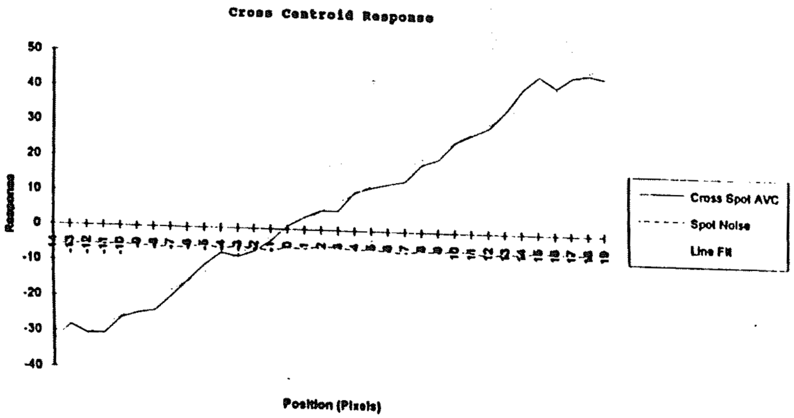


Figure 8. Cross projected centroid response.

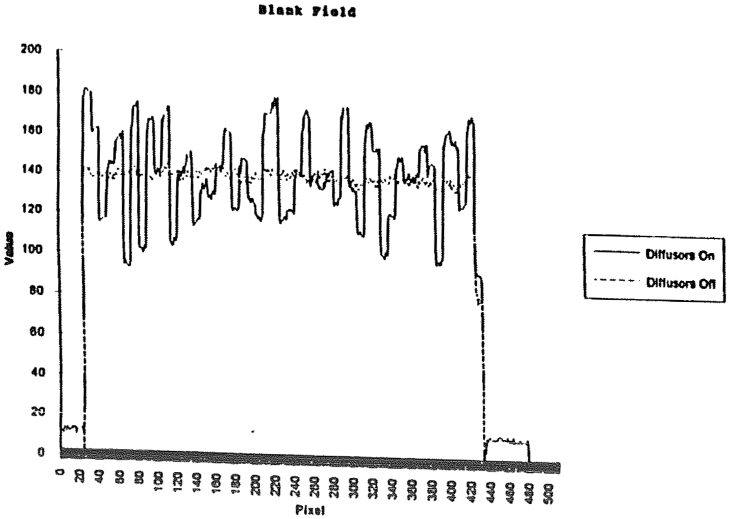


Figure 9. Flat field response.

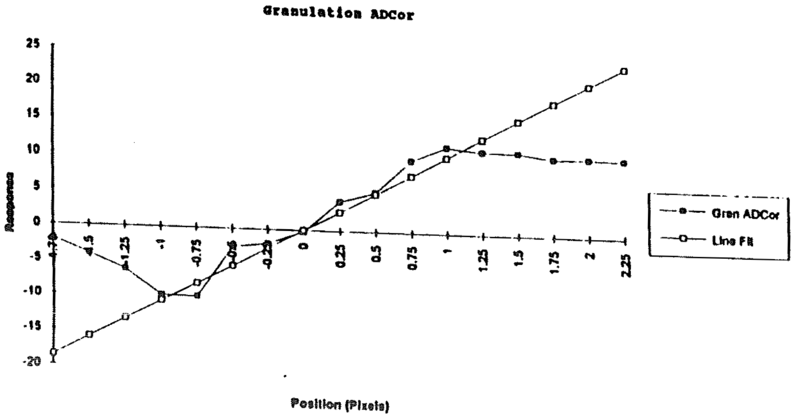


Figure 10. Absolute sum of difference response.

the simpler VC focal plane. For spot tracking with the VC focal plane a full projection centroid position sensor produces an excellent response [3]. In this case the detector mismatch is not an issue. A modified VC sensor is under development which includes several circuit improvements that should allow a sensor bandwidth of around 500kHz to 1 MHz, limited only by the photo detector response. In addition, this chip will incorporate the two axis servo electronics so that only a couple of power opamps will be required for the tracking mirror drive.

For future solar vector magnetograph applications, tracking granulation in the presence of illumination gradients is a requirement. A modified RVC chip is under development that will employ cross projected absolute difference correlation circuitry. For this application, bandwidths similar to the existing VC chip of around 40 kHz are adequate. We have not yet resolved the tradeoff of lock range versus monotonicity.

Of course, for space applications the performance of these chips in a radiation environment is an issue. Since the parts are analog, single event upsets are not a problem, and "radiation noise" effects are usually out of the loop bandwidth. The very low currents and voltage swings present in these parts allow the single 5 volt power supply to be current limited to about 1 ma so that if latchup occurs, it will not be destructive, and can easily be detected. We plan on investigating a shielding solution to total dose requirements based on either a pinhole or a periscope shielding configuration. Alternatively, these parts could be implemented in a radiation tolerant process.

In summary, the VC chip and the RVC retina are single monolithic parts which produce high bandwidth analog position signals unencumbered by readout electronics, as well as a scanned out video signal (We have found that a real-time video display of the sensor field of view is very valuable during system development and setup). The VC chip is being modified to operate at bandwidths of up to 1 MHz, and to provide most of the servo electronics for a two axis beacon position servo system on a single integrated circuit of modest size. Thus, in addition to higher bandwidth, these position sensors can be used to provide a beacon tracking servo system that is highly integrated with resulting higher reliability, and lower complexity, development time, power, weight, volume, and cost. We believe that this represents a very attractive system for use in laser beacon tracking systems, as well as for optical laser communication systems for space or terrestrial point to point communications systems.

References

[1] P.V. LaSala and Chris McLaughlin, "Beam tracker and point ahead system for optical communications II - servo performance," SPIE Vol. 1482 Acquisition, Tracking ,and Pointing V (1991), pp. 121-137.

[2] K. Strohbehn, D.M. Rust, A.G. Andreou, and R.E. Jenkins, "A Biologically-Inspired Image Position Sensor," in Real-Time and Post-Facto Solar Image Correction,Proceedings of the 13th Sacramento Peak Summer Workshop(1992), R. Radick(ed.,),National Solar Observatory, Sunspot, NM.

[3] K. Strohbehn, A.G. Andreou, R.E. Jenkins, and X. Sun, "Video Computation Organization," Proceedings of Government Microcircuit Applications Conference, November 1992, Las Vegas, Nev., pp 585-588.

[4] D.M. Rust, J.W. O'Byrne, and T. Harris, "An Optical Instrument for Measuring Solar Magnetism," Johns Hopkins APL Tech. Dig. 11,1988,p 77.

[5] C.A. Mead and T. DelBruck, "Scanners for Visualizing Activity of Analog VLSI Circuitry," Analog Integrated Circuits and Sig. Proc., I, p.93.

[6] K.A. Boahen and A.G. Andreou, "A Contrast Sensitive Silicon Retina with Reciprocal Synapses," in Advances in Neural Information Process. Syst., 4, Morgan Kaufmann, San Mateo, CA., p. 764.

[7] S.P. DeWeerth and C.A. Mead, "A Two-Dimensional Tracking Array," Advanced Research in VLSI, Fifth MIT Conference, March 1988.

Adaptive Analog Video Signal Conditioner

CheolHo Jeong, Brian Yu, Gregory Donohoe and Robert Vitello
Department of Electrical and Computer Engineering
University of New Mexico
Albuquerque NM 87131

Abstract - **This paper presents an adaptive analog video signal conditioner. This system supplements conventional irises and automatic gain controls to condition video signals for digitizing with a uniform analog-to-digital converter. Employing two degree of freedom, this circuit adjusts an analog gain and analog offset to increase the average information content of digitized images as defined by an entropy measure. The globally near-optimized implementation tracks smoothly to adapt to changing scene conditions.**

1 Introduction

Video signals are often digitized for efficient transmission and for automatic image analysis, usually with a uniformly quantized analog-to-digital converter (ADC) as shown in Figure 1 [1, 2]. It is important to condition the video signal so that the digitized images contain the desired information. This conditioning must be done on the analog signal: once an image is digitized, the information it contains cannot be increased by processing. To facilitate automated image analysis, the system must automatically adjust to changes in the scene such as changing illumination.

Currently, video cameras that provide data for vision systems usually exploit two mechanisms for mapping the visual scene onto the finite range of a digitizer. These are the automatic iris and the automatic gain control (AGC). Both provide one degree of freedom: the automatic iris controls the average energy striking the sensing elements of the camera, and the AGC adjusts the average energy in the analog video signal. Even though these mechanisms provide easy ways to normalize images, they are not enough to provide satisfactory results. Extending the ideas behind them, we would like to develop more robust schemes to reliably normalize images and adapt to the environment.

2 Information Measure

In connection with the concept of information, the computer vision system in Figure 1 can be treated as an information processor where information is provided in the form of images. Therefore, it is important to provide as much information as possible that can be processed and analyzed by the system. The most important change in information occurs through a digitizer when analog images are reduced to digital images. Information in images, however, cannot be increased by processing in the digital domain since information can be increased only by the nontrivial refinement corresponding to one-to-many mapping, which is impossible in the digital domain [3]. In this context it is crucial to process or normalize images in the

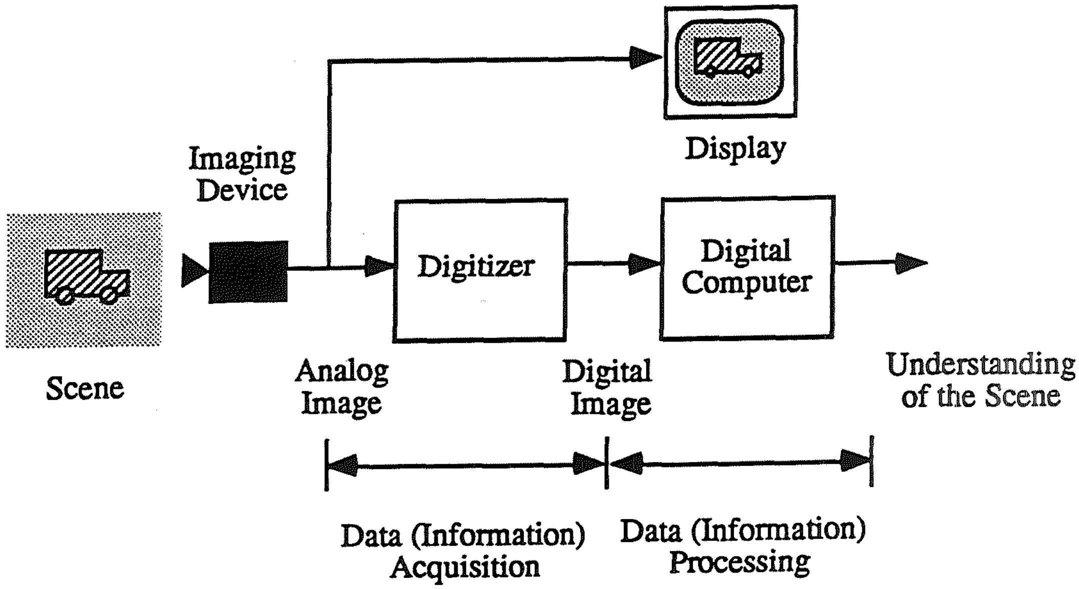

Display
Imaging Device
Digitizer
Digital Computer
Scene
Analog Image
Digital Image
Understanding of the Scene
Data (Information) Acquisition
Data (Information) Processing

Figure 1: Computer vision system

analog domain before digitization in order to achieve the greatest benefit from normalization to get as much information as possible.

To measure the content of information, several information measures called entropy have been introduced [3, 4]. Among them, the Shannon entropy is the most common and natural measure of information and accepted widely in various fields [5]. It provides useful and intuitive properties to interpret information in sources. The Shannon entropy $H(p)$ of a source is defined as

$$H(p) = -\sum_{n=0}^{N-1} p_n \log p_n \tag{1}$$

for the probability distribution p from the set Γ_N of all complete finite (N-ary) probability distributions

$$\Gamma_N = \left\{ p = (p_0, ..., p_{n-1}) \,\middle|\, \sum_{n=0}^{N-1} p_n = 1, p_n \geq 0 \right\} \tag{2}$$

The Shannon entropy can be said to represent information in general sense. In various kinds of applications, meaningful information in images could be different from one application to another. So it is important to find an entropy which can exploit effectively information that we are most interested in. In motion detection problems, information about moving objects is the most important part of information delivered by images. Based on a basic motion detector [6], we derive an information measure called *detectability* [7]:

$$D(p) = -\log \sum_{n=0}^{N-1} p_n^2 \tag{3}$$

which represents the probability of detection and is virtually identical to the Rényi entropy of order 2[4].

Based on the above discussion, we choose entropies to measure the quality of images and propose a mechanism to adaptively maximize the entropy of digitized images under changing scene conditions.

3 Quantizer

The digitizer in Figure 1 is usually composed of the cascade of a transformation and a uniform quantizer as shown in Figure 2 (disregarding spatial sampling) [8, 9]. Quantization is achieved by transforming the input signal x using a transformation $T(x)$ and by quantizing the transformed signal y into the output signal z employing a uniform quantizer. It is assumed that $T(x)$ is a continuous and single-valued function. Furthermore, since we wish $T(\,\cdot\,)$ to preserve an ordering relationship on signal amplitude, we require that $T(\,\cdot\,)$ be monotonically increasing.

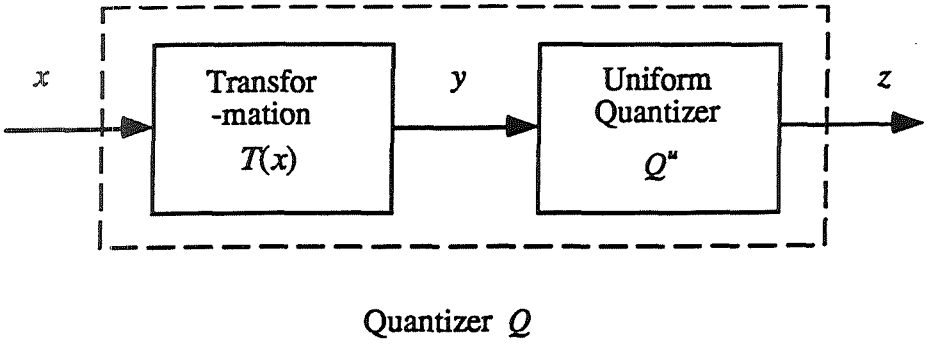


Figure 2: Quantizer configuration with transformation

Since the uniform quantizer has a fixed finite dynamic range, it is required to map the input signal x into the range to be properly quantized [10]. Without loss of generality, assume the dynamic range of the uniform quantizer Q^u be [0, 1]. Note that the cascade of two components can be considered as a quantizer Q whose characteristic is controlled by the transformation.

For commercially available digitizing systems, there are usually two analog parameters that can be applied to the signal in the analog domain before quantization. These are analog gain and offset which produce a simple linear transformation. In the case of images, they correspond roughly to contrast and brightness, respectively. We are going to concentrate our discussion on the case of linear transformation which provides two degrees of freedom:

$$y = T(x) = ax + b \tag{4}$$

where a is analog gain ($a > 0$) and b analog offset. Then the dynamic range of the overall quantizer becomes

$$\left[\frac{-b}{a}, \frac{1-b}{a}\right] \tag{5}$$

which can be regarded as a digital window on the analog world as shown in Figure 3. The window has a width and a position, determined by the minimum and maximum clipping levels. The gain parameter controls the width of the digitizing window; the offset controls the position of the window. The window width and position must be controlled together to produce images of high quality. Figure 3 shows that transformation *Line 1* places the window properly on the probability density function (PDF) of the input x to produce digital output of high entropy. Transformation Line 2 represents a case of inadequate transformation.

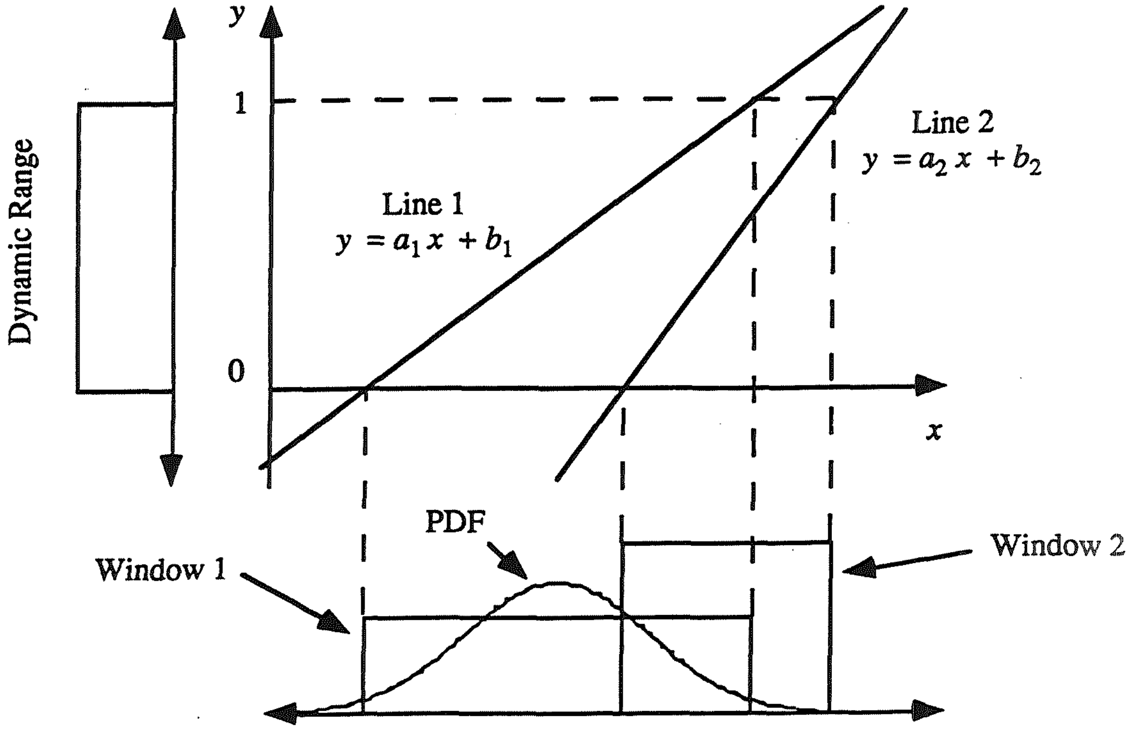


Figure 3: Digitizing window

4 Algorithm

We now face the problem of finding optimum values of gain and offset in (4) to maximize the entropy of digital output given PDF of the input analog image. This leads to a highly nonlinear optimization problem which requires numerical optimization methods to solve [11]. Numerical methods, in general, require a lot of computation, result in slow convergence, and suffer from the local optimum problem. This problems are very crucial when changing scene conditions introduces changes in the input PDF, in turn, different optimal values as time goes by. Here we propose a simple algorithm to address the problem of tracking the moving optimum. The optimal transformation producing maximum entropy is proved to be simply the cumulative distribution function (CDF) of the input x [9]:

$$T^*(x) = P_x(x) = \int_{-\infty}^{x} p_x(t)dt \tag{6}$$

This transformation is called the histogram equalization function since it transforms the input x into a y whose PDF is uniform in the dynamic range of the uniform quantizer [0, 1], as the probability distribution of the discrete output will be uniform.

From the above observation, we can find an appropriate linear transformation by simply approximating the optimal transformation as shown in Figure 4. Even though this approach does not provide the optimal solution in the sense of entropy, we can build a simple and easy way to normalize images using a linear transformation since we can find an explicit solution.

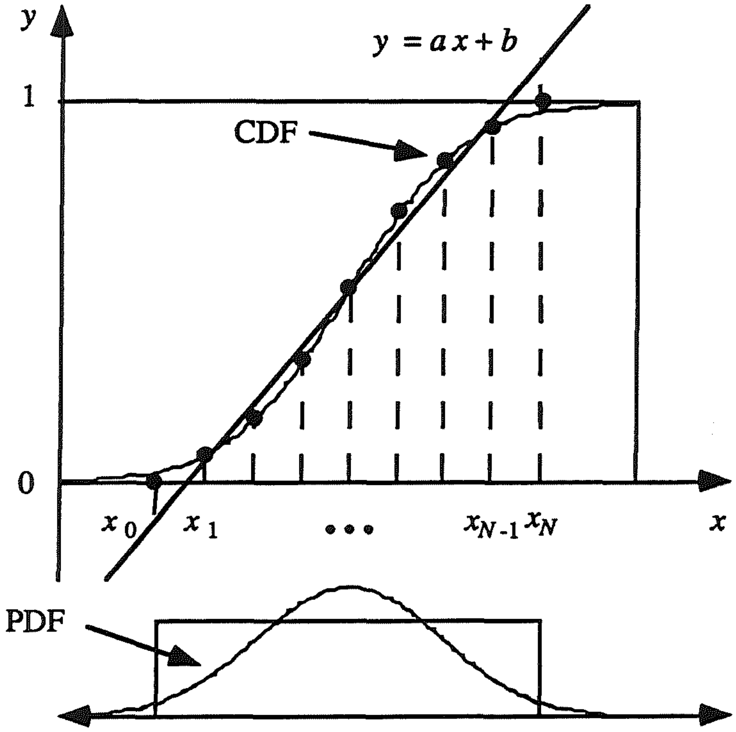


Figure 4: Line approximation method

Before applying the above idea, there are a few restriction to be considered in real applications. One of the important restrictions is that we have to guess the nature of the analog world from our view. In the context of estimating the CDF, we can have a good estimate only in the dynamic range of the uniform quantizer: because of saturation, we have no way to properly estimate the CDF outside the dynamic range. It basically prevents us from using an error function which requires complete description of the distribution. To address the above problems, we do the following assumption on the distribution: the distribution is concentrated on the dynamic range as if no saturation occurs. Having the assumption in mind, we can build an discrete CDF data from histograms:

$$y_n = \begin{cases} 0 & n = 0 \\ P_{x_n} & n = 1, ..., N \\ 1 & n = N \end{cases} \tag{7}$$

where N is the number of quantizer intervals and $x_n, n = 0, ..., N$ are uniformly sampling points in the dynamic range in (5):

$$x_n = \frac{1}{a}\left(\frac{n}{N} - b\right), n = 0, ..., N \tag{8}$$

which are equal to the decision levels of the quantizer Q except the outermost points, x_0 and x_N. Note that $y_n, n = 1, ..., N-1$ are corresponding to the sampling points of the CDF at x_n as shown in Figure 4.

Then using the mean-square error:

$$E(a, b) = \sum_{n=0}^{N} (y_n - ax_n - b)^2 \tag{9}$$

we can find an explicit solution a^* and b^* for gain and offset, respectively [11]. Finally, updating the parameter $a(k)$ and $b(k)$ at time k by

$$a(k+1) = a(k) + \alpha(a^* - a(k)), \ \ 0 \leq \alpha \geq 1 \tag{10}$$

$$b(k+1) = b(k) + \beta(a^* - b(k)), \ \ 0 \leq \alpha \geq 1 \tag{11}$$

where α and β are smoothing constants for gain and offset, respectively. Since it does not suffer from the local optimum problem due to direct use of the CDF, the proposed algorithm easily tracks the globally near-optimum as scene conditions change. Also it is simple to track the optimum quickly, which is important in real applications.

5 Signal Conditioner

5.1 Design Considerations

In order to build a test board for the proposed algorithm, we decided the design should at least consist of three functional modules which are analog hardware, digital hardware and a software driver. The analog hardware module will take a RS-170 compliant video signal, provide termination, buffering, sampling and conditioning, which is to supply the two degree of freedom for the algorithm. The digital hardware module includes the interface to an analog to digital converter (ADC), a digital to analog converter (DAC) and a personal computer (PC). Most importantly, it contains a histogram calculation circuit. The software module is actually the algorithm itself, plus some low level drivers to provide handshaking with the digital hardware module.

The adjustment range is $-10dB$ to $20dB$ for the gain parameter and 0 to 10 volts for the offset parameter. These two factors together will affect the selection of analog components. For the calculation of the histogram, which is the major data structure required in this algorithm, we decided to sample only the even field of an image with 8 bits per pixel and 256 samples per line.

There are several alternatives for repeatedly updating the histogram of an on-line image. Our solution is to use the output from the ADC as the address to the location in the histogram which can be stored in a static RAM (SRAM). The SRAM will undergo an access-increment-store cycle every time the ADC outputs a valid data sample. At the end of sampling a field, the final data kept in the SRAM is the histogram of the incoming image.

5.2 Implementation

As a result of a survey of available components, we come up with an off-the-shelf design as sketched in Figure 5. The 75 ohm, AC coupled, RS-170 video output from a CCD camera is terminated, buffered by an operational amplifier circuit, DC restored by a JFET circuit, and sync stripped by a diode bridge circuit. The sync detection is done by a video sync separation chip from National Semiconductor whose even/odd signal output simplifies the design of digital control logic.

The signal conditioning circuit that provides gain and offset adjustment is accomplished by a wideband precision analog amplifier from Burr-Brown which can perform

$$\frac{(x - x_0)(y - y_0)}{SF} + (z - z_0) \tag{12}$$

where x, x_0, y, y_0, z, z_0 are inputs and SF is an adjustable scale factor. Some mapping circuits are needed to allow direct interface with the ADC and DAC, which are donated by Signetics. The adjusted output from the signal conditioning circuit has to again pass through a limiter in order to be recombined with sync signal and buffered to be a RS-170 complied video signal for output to a monitor.

The digital hardware module is implemented using a digital input/output board (PC-DIO-24) from National Instruments, a two-ported SRAM from Cypress Semiconductor and some PLDs, PEEL22V10. The software is implemented in Borland C++ which has enough graphics driver for our display needs.

6 Experiment Results

We applied the proposed normalization scheme to a sequence of analog images used for a motion detection problem. Detectability D in (3) was used as the information measure. We observed that the system produced well-balanced images and adapted very well to changing image conditions. Figure 6 shows a result from one frame of the images. The original image Figure 6(a) has low contrast as seen from its histogram ($D = 5.52$). Examining the detectability surface of the image in Figure 6(c) shows that it has its maximum $D = 7.193$ at gain 10 (dB) and offset -316. The proposed scheme converged to gain 10 (dB) and offset -299 with $D = 7.190$, which is very close to the optimum point. The resulting normalized image Figure 6(b) shows very high contrast: Compare the histograms in Figure 6(d).

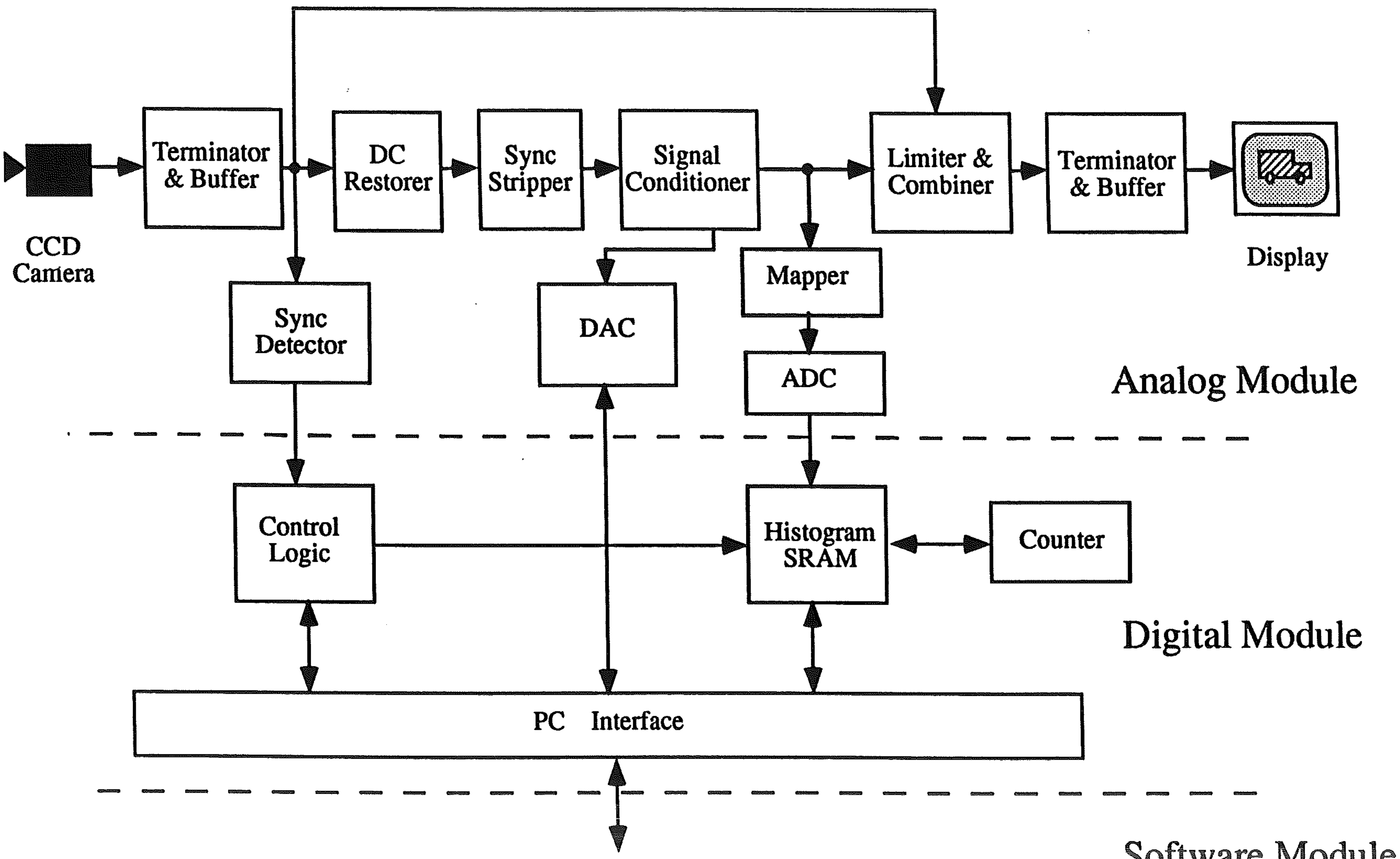
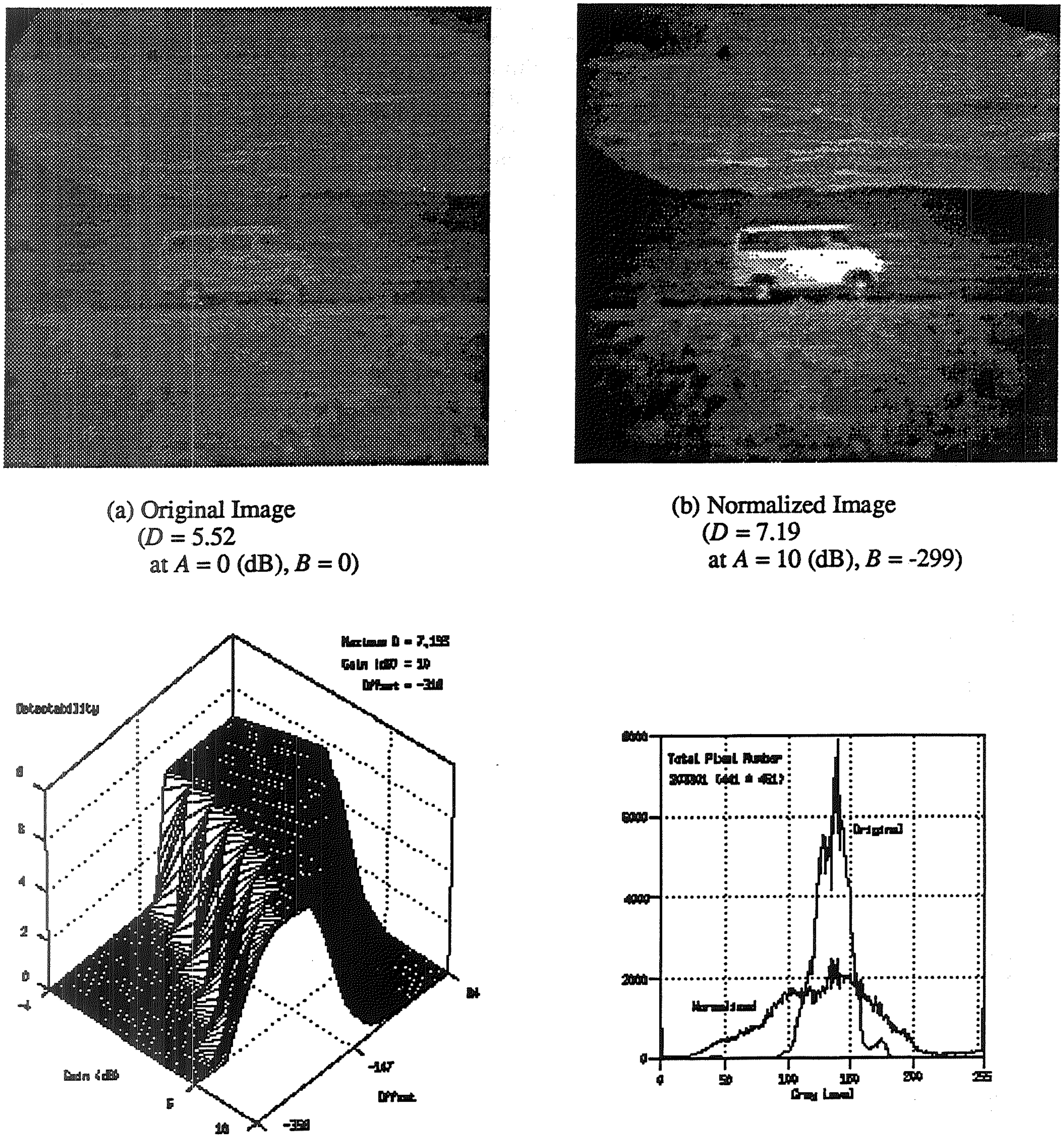


Figure 5 Adaptive analog video signal conditioner



(a) Original Image
($D = 5.52$
at $A = 0$ (dB), $B = 0$)

(b) Normalized Image
($D = 7.19$
at $A = 10$ (dB), $B = -299$)

(c) Detectability surface
(Max. $D = 7.193$
at $A = 10$ (dB), $B = -307$)

(d) Histogram
(Total pixel number = 203301)

Figure 6 Experiment result (D: detectability, A: gain, B: offset)

7 Conclusion

We have proposed a system to condition analog video signals for digitizing with a uniform analog-to-digital converter, using two degree of freedom. The system produces near-optimal images in the sense that they have nearly maximum entropy, given the linear constraints of the analog conditioning circuitry. The conditioned images are also suitable to be viewed by humans, as in video security applications. We have demonstrated the system in simulation, and designed and constructed a demonstration model. We find that the system often achieves entropies of over 7 bits per pixel out of a maximum of 8.

The image transformation is currently limited to a linear function. More flexible transformation will provide better entropies and extensions to a high-order polynomial or piecewise linear gain function are being examined.

References

[1] D.H. Ballard and C.M. Brown, *Computer Vision*, Englewood Cliffs: Prentice-Hall, 1982.

[2] W.N. Martin and J.K. Aggarwal, "Dynamic Scene Analysis", Computer Graphics and Image Processing, 356-374, 1978.

[3] R.E. Blahut, *Principles and Practice of Information Theory*, Addison-Wesley, 1987.

[4] J. Aczl and Z. Darczy, *On Measures of Information and Their Characterizations*, New York: Academic Press, 1975.

[5] J.A. , B. Forte, and C.T. Ng, "Why the Shannon and Hartley Entropies are 'Normal'", Adv. Appl. Prob., vol. 6, 7 - 12, March 1974.

[6] G.W. Donohoe, D.R. Hush, and N. Ahmed, "Change Detection for Target Detection and Classification in Video Sequences", in Proc. ICASSP-88, New York, April 1988.

[7] G.W. Donohoe and C. H. Jeong, "A Combined Analog-Digital Technique for Normalizing Video Signals for the Detection of Moving Objects", in Proc. ICASSP-92, San Francisco, March 1992.

[8] A. Jain, *Fundamental of Digital Image Processing*, Prentice-Hall, 1989.

[9] R.A. Hummel, "Histogram Modification Techniques", Computer Graphics and Image Processing, 209 - 224, 1975.

[10] N.S. Jayant and P. Noll, *Digital Coding of Waveforms - Principles and Applications to Speech and Video*, Prentice-Hall, 1984.

[11] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1988.

VLSI Mixed Signal Processing System

A. ALvarez and A. B. Premkumar
School of EEE and School of Applied Science
Nanyang Technological University
Nanyang Avenue, Singapore 2263.
E-mail: eamalvaraz@ntuvax.ntu.ac.sg

Abstract - **An economical and efficient VLSI implementation of a Mixed Signal Processing system (MSP) is presented in this paper. The MSP concept is investigated and the functional blocks of the proposed MSP are described. The requirements of each of the blocks are discussed in detail. A sample application using active acoustic cancellation technique is described to demonstrate the power of MSP approach.**

1 Introduction

Mixed Signal Processing (MSP) concept is by no means new. MSP allows both analog and digital signal processing to be performed simultaneously on silicon.

MSP incorporates the required conversion from the analog world and exploits the intrinsic advantages of digital processing techniques. The present day VLSI technology provides an economical and efficient solution for integrating both analog and digital counterparts for implementation as MSP. In the VLSI implementation of MSP, architectural decision and requirements influence speed and performance. This paper discusses a VLSI implementation of an MSP system for general purpose DSP applications. An adaptive acoustic noise cancellation application is described in which the cancellation algorithm is implemented on the proposed MSP architecture.

2 VLSI Mixed Signal Processing System

The proposed MSP system includes the five functional modules and is as shown in Figure 1. The modules are:

1. Input Signal Conditioning
2. Analog to Digital Conversion
3. Digital to Analog Conversion
4. DSP Interfacing
5. Output Signal Conditioning

Each of these modules is described in detail in the following sections.

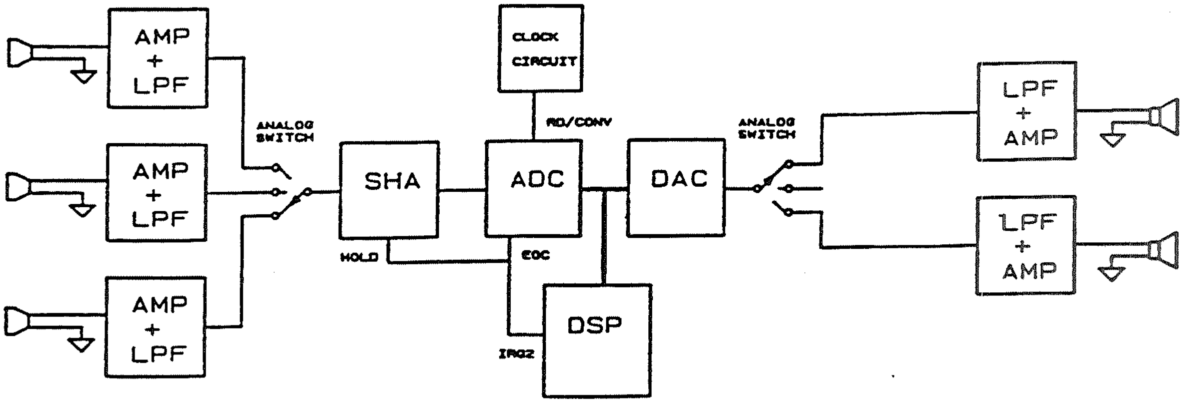


Figure 1: Block Diagram of MSP

2.1 Input Signal Conditioning Module

This module consists of five identical channels (Five channels were chosen for noise cancellation application).

A low frequency, high performance, high impedance amplifier with a variable gain of 15 to 2000 amplifies the input signals. The amplified signal is then filtered using a second order Butterworth filter. The filter is designed for flat response in the frequency range of interest. The flat response is maximized by choosing the resistor values in the filter. Any desired low frequency bandwidth can be obtained by cascading several such stages [1].

For example, for the noise cancellation application, two such stages are cascaded to obtain a bandwidth of 5780Hz.

2.2 Analog to Digital Conversion Module

The inputs from the signal conditioning module are multiplexed using analog switches and then passed through a Sample and Hold circuit. The Sample and Hold circuit is designed to have an acquisition time of $3\mu s$. The 12 bit A/D converter in the proposed architecture is designed to have a conversion time of $35\mu s$ and produces offset binary format at the output.

The conversion time enables sampling to be done at 28KHz which meets the design requirements in the noise cancellation application.

2.3 Clock Circuit

The waveform generator is designed to operate between 2KHz and 80KHz. Use of variable resistors in the design of the oscillator allows for the change in the frequency. The clock pulses are shaped to have a width of $1\mu s$. This is done to ensure proper conversion in the A/D converter.

2.4 Digital to Analog Conversion Module

The D/A is a 12 bit converter module designed to have a conversion time of $2\mu s$. The DSP processor activates this conversion by asserting the DAC signal line enabling the data from the DSP to pass through the buffer to the D/A converter. The output is switched to the output signal conditioning module.

2.5 Output Signal Conditioning Module

The output signal from the D/A converter is multiplexed to a number of output channels. In our present application four such channels are needed. A Sample and Hold circuit is used at the output in order to give a continuous waveform when switching is done between channels. The acquisition time is designed to be $4\mu s$. A fourth order Butterworth low pass filter is then used to filter off the high frequency components created by the D/A converter.

2.6 Digital Signal Processing Module

The DSP module is a programmable processor optimized for DSP and other numeric processing applications. There are three full function, independent computational units that process 16 bit data directly. Addresses are provided by two address generators and a program sequencer. The DSP processor has additional on board program and data memory, serial ports, timer and extensive interrupt capabilities.

The DSP, besides processing data, has been designed to have control capability, such as starting of A/D and D/A etc [2].

3 Application to Noise Cancellation

This section describes an application using the VLSI MSP processor designed as above. The application considered is a typical acoustic noise cancellation problem. Unwanted noise such as those generated by machinery often needs to be attenuated. To cancel the noise, the active noise cancellation technique is implemented on the MSP system. This cancellation technique superimposes a signal in antiphase and of equal magnitude to that of unwanted noise. The region where there is considerable amount of cancellation is determined by the geometric arrangement of the primary and secondary sources and the range of frequencies of interest. The higher the frequency the smaller the area of cancellation and vice versa [3, 4, 5, 6].

Since noise is random and the environment of operation is time varying, a digital signal processor is used with adaptive controller. In order to construct a practical adaptive controller, some measurable error criterion must be used. In this case the error will be the noise level picked up by the sensors (microphones) placed in the region of interest. The signal processing problem is then to apply an adaptive algorithm to adequately minimize the error. The Least Mean Square (LMS) adaptive algorithm is used due to its simplicity and robustness. The LMS algorithm is implemented as a transversal Finite Impulse Response (FIR) filter.

3.1 System and Controller Design

It follows from a theoretical consideration of field cancellation that active attenuation of noise through wave interference consists of three main processes, namely, detection, negation and superposition. Figure 2 shows the set-up of the system. The system consists of a single or primary noise source, a reference pick-up microphone, R, an observation point, O and a controller, C, driving a loudspeaker, S. A complex frequency domain S domain description of the system is shown in Figure 3, where $E(s)$, $F(s)$ $G(s)$ and $H(s)$ are the transfer functions of the paths re, rf, rg and rh respectively. The transfer function of the required control is then:

$$\frac{S(s)}{P(s)} = \frac{M(s)C(s)N(s)L(s)E(s)}{1 - M(s)C(s)N(s)L(s)F(s)} \tag{1}$$

Since the objective is to cancel or at best reduce the level of noise at the observation point, the observed primary and secondary signal, $P_o(s)$ and $S_o(s)$ should be opposite of each other. Given $P_o(s) = G(S)P(s)$, and $S_o(s) = H(s)A(s)P(s)$, then:

$$G(s) = -H(s)A(s) \tag{2}$$

where $A(s) = \frac{S(s)}{P(s)}$ Substituting Equation 2 into Equation 1 and simplifying:

$$C(s) = \frac{G(s)}{M(s)N(s)L(s)\Delta(s)} \tag{3}$$

where $\Delta = F(s)G(s) - E(s)H(s)$. Equation 3 gives the exact transfer function for the cancellation at a known fixed point of interest, if the transfer characteristics of all signal paths are known. Any change in the location of the the detector and/or the observation point with respect to the primary and secondary sources are reflected in the controller transfer function.

Figure 2: System Configuration

3.2 The LMS Algorithm

The LMS algorithm is a gradient search algorithm that constantly tries to search for the point of least error. The algorithm adaptively generates a new set of coefficients of an FIR filter at every sampling interval. In the present application the LMS algorithm will have to take into consideration the distortion to the signals due to various acoustic paths. The digital implementation of the FIR is expressed as [7, 8]:

$$y(n) = \sum_{k=1}^{M} h(k,n)x(n-k+1) \tag{4}$$

where M is the length of the filter, $x(n)$ is the input signal and $h(k,n)$ is the k^{th} tap weight at time n. The stability of the FIR is conditioned on the bounded filter tap coefficients generated by the LMS algorithm. In the case of active noise cancellation, there exist acoustic paths from the cancellation signal speaker and the noise source to the error pick up microphones. These cause delay to the signal and hence, the difference between the filter output and the desired output is no longer directly available to update the filter coefficients. It is, therefore, useful to change the input signal to a reference signal, $r(n)$. The LMS algorithm then, becomes:

$$h(k,n+1) = h(k,n) - \mu e(n)r(n-k) \tag{5}$$

where, $r(n)$ is the reference signal and μ is the convergence coefficient and $e(n)$ is the error signal between the desired and output signals. If it is assumed that the error path introduces only a pure delay, then a further simplified version is:

$$h(k,n+1) = h(k,n) - \mu e(n)x(n-k-t_d) \tag{6}$$

where t_d is the discrete time delay due to the acoustic path. The continuous time delay, t_{dc}, is computed as $t_{dc} = \frac{d}{v_s}$, where, v_s is the velocity of sound in air and d is the distance the canceler signal has to travel to the error detector. The discrete time delay, t_d, is dependent on the sampling frequency.

4 Computer Simulations

The verification of the proposed architecture and algorithm was carried by simulations. With the simulations, one is able to observe the error signal at the point of interest and the variation of each of the filter tap coefficient. The simulations, also, enable one to determine the initial values of the convergence coefficient. Several assumptions are made in the characterizing the acoustic paths and it is also assumed that there are no distortions to the noise signal picked up by the microphone. A suitable initial value of the convergence coefficient is assumed at the beginning of the simulations. Figure 4 shows a random signal and the canceler output, with a μ value of 0.0025. We observe that the cancellation signal is the exact inverse of the reference signal. The total error between the two signals is shown in Figure 5. Figure. 6 and 7 show the reference, cancellation and the error signal taking into account the delay due to the propagation of the signal and when the delayed LMS algorithm is applied.

5.5.6

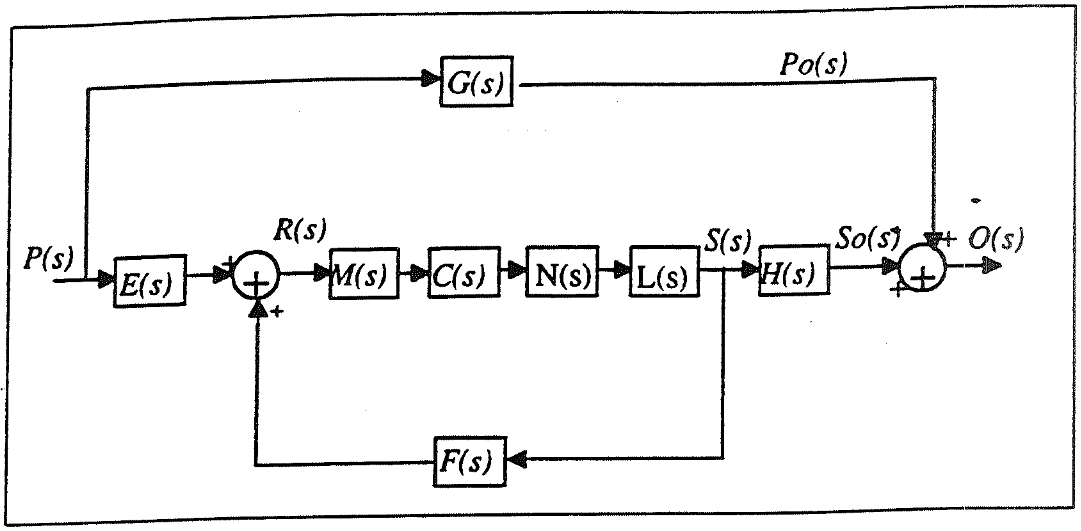


Figure 3: Domain Block Diagram

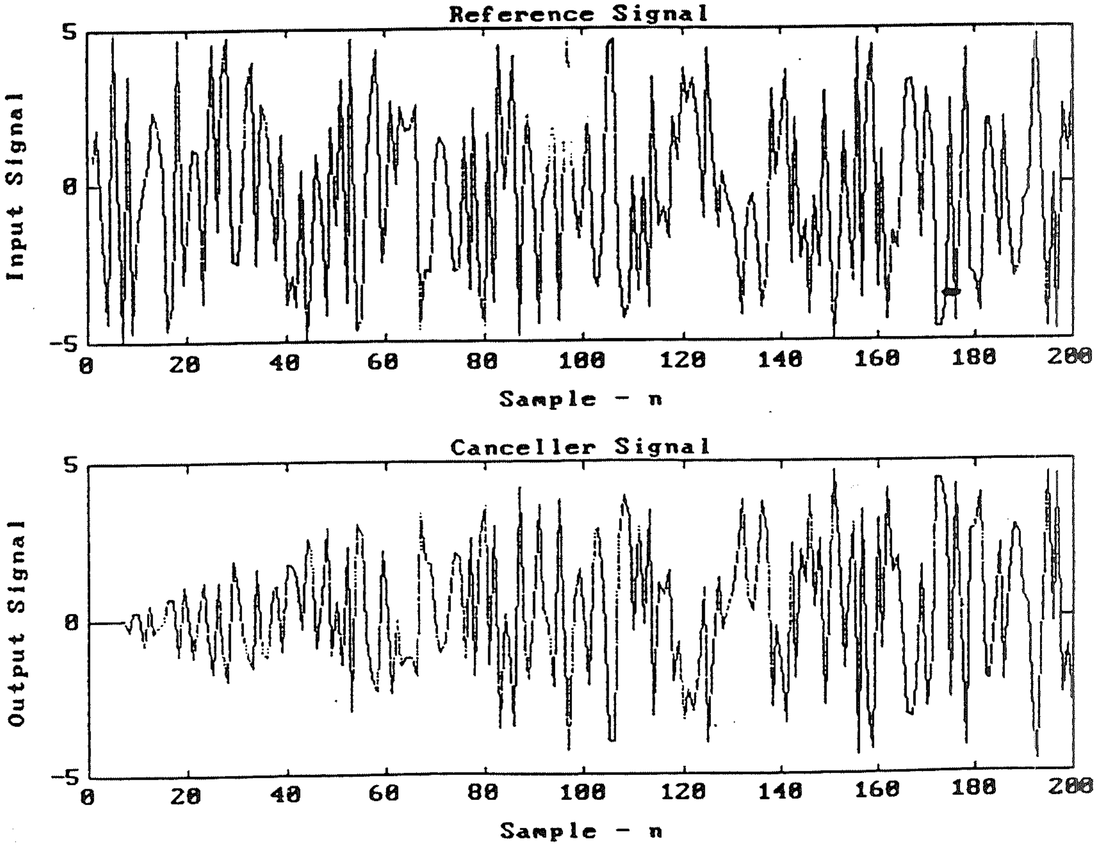


Figure 4: Reference and Cancellation Signals

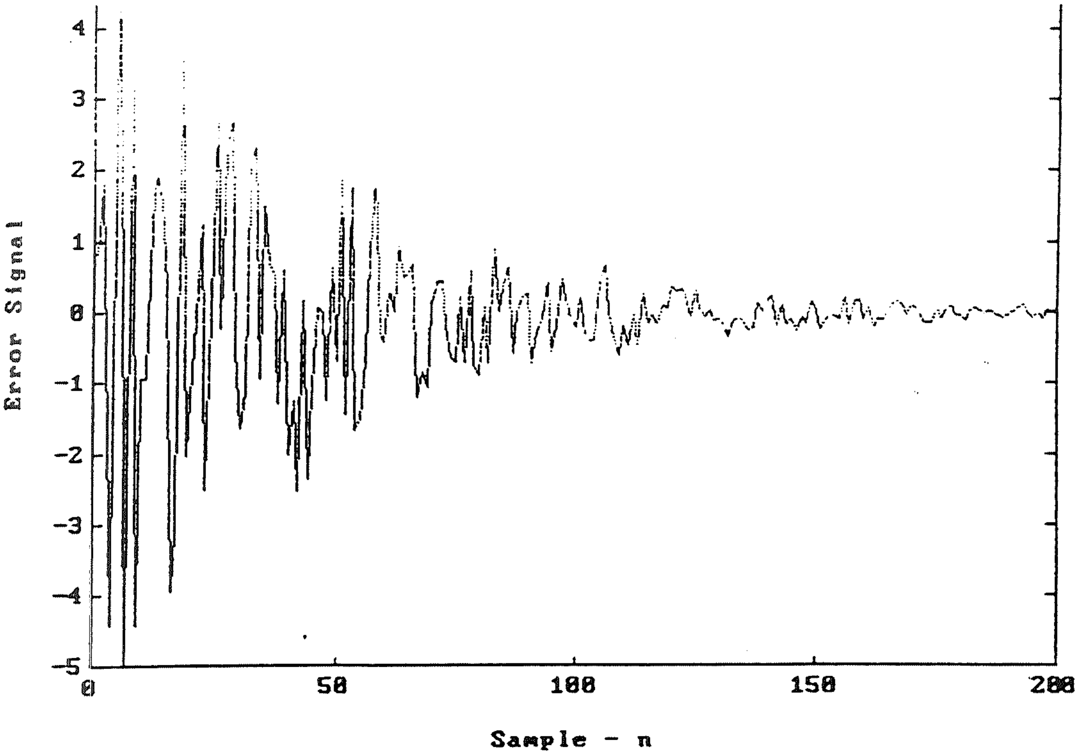


Figure 5: Error Signal

5 Results and Observations

The performance of the canceler circuit was evaluated using simple sine waves. The noise was simulated by an oscillator (280 Hz) and a power amplifier. Two microphones were used to pick up reference signal and the error signal at the point where noise is to be canceled. The canceler speaker and the noise source were placed close to each other to obtain wide spread cancellation. The plots in Figure 8 show noise source and the signal at the point where it is supposed to be canceled. Figure 9 shows the noise signal at the point of interest after the canceler has been activated. It is observed that the noise has been significantly attenuated. The fluctuations observed around the canceled noise is due to the ambient noise present. The results observed when the frequency was changed to 400 Hz were similar.

With two point cancellation the zone of coverage is found to be greater. However additional path delays need to be considered and these were input using switches. In the two point cancellation, if the noise source is varied randomly about a certain frequency that the system has been set for, the system becomes unstable. This may be overcome by modeling the acoustic paths as an FIR filter and applying it to the reference signal.

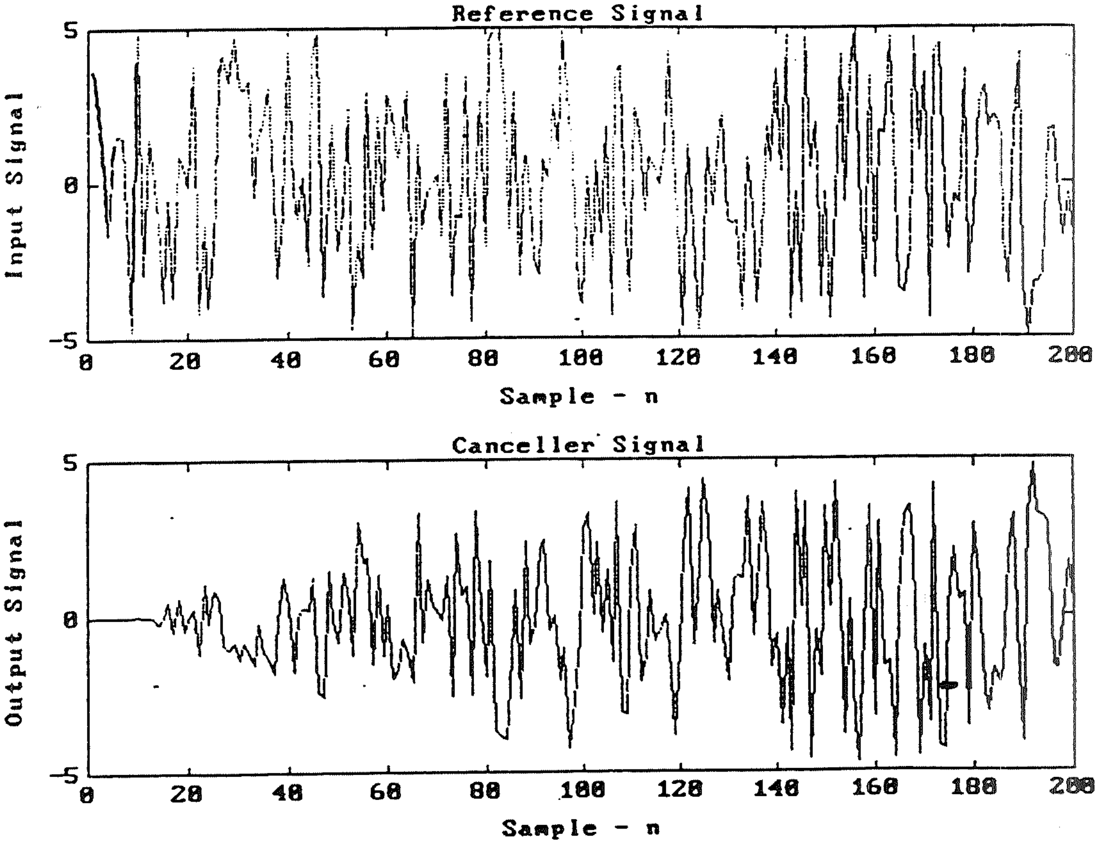


Figure 6: Reference and Cancellation Signals (Delay Included)

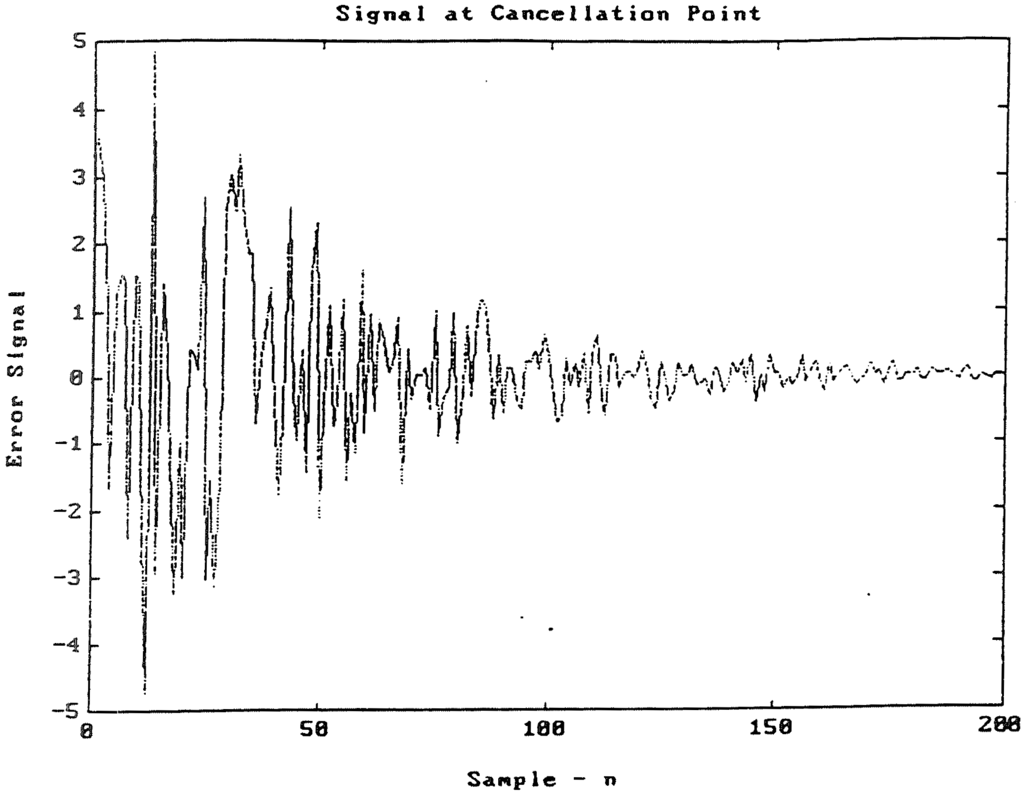
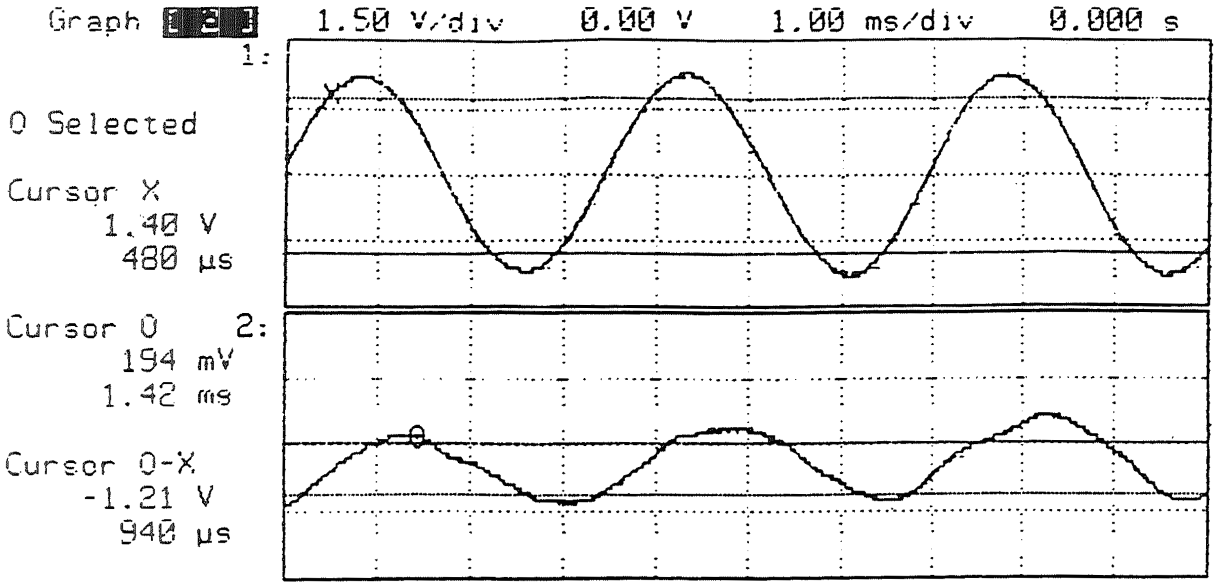


Figure 7: Error Signal (Delay Included)



```
Graph [1 2]   1.50 V/div   0.00 V   1.00 ms/div   0.000 s
          1:
O Selected

Cursor X
   1.40 V
    480 µs

Cursor O      2:
    194 mV
   1.42 ms

Cursor O-X
  -1.21 V
    940 µs
```

Figure 8: 280Hz Before Cancellation

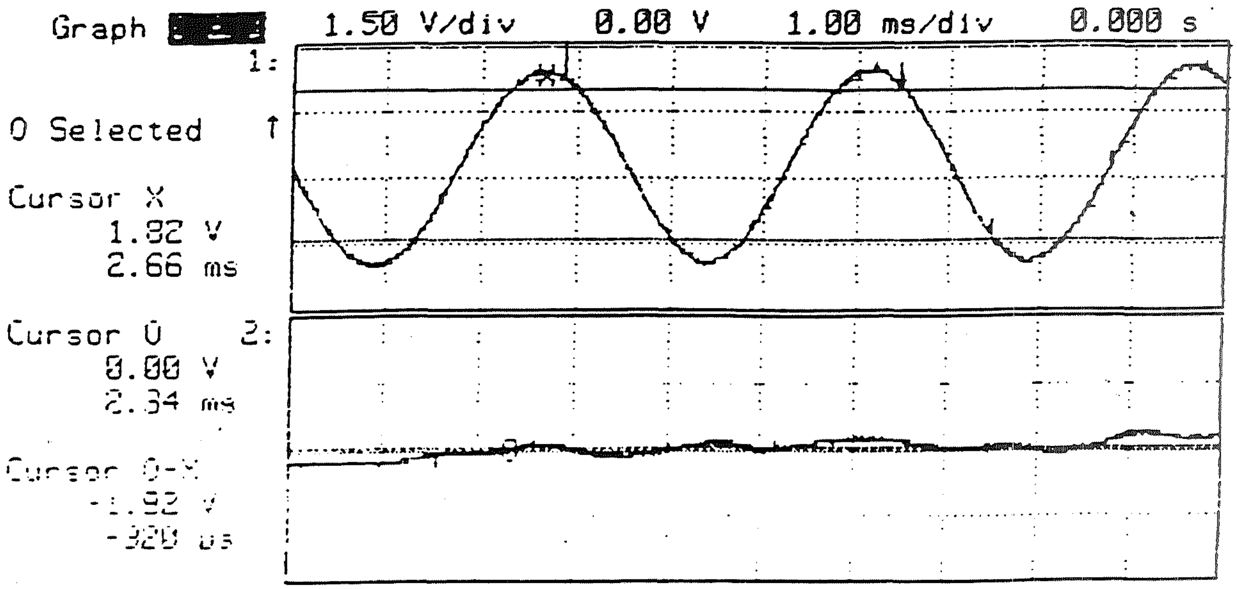


Figure 9: After Cancellation

6 Conclusions

An architecture for mixed signal processing has been proposed and the various functional blocks have been discussed in detail. The appropriate features needed for each of the modules have been discussed. An implementation of the proposed architecture in an example problem of noise cancellation has been discussed and the results show efficient DSP processing can be achieved with architecture proposed. Simulations performed show that cancellation of both sinusoidal and random signals is possible with the implementation of delayed LMS algorithm which takes into consideration the time delay due to different acoustic paths.

References

[1] Sergio Franco, *Operational Amplifiers and Analog Integrated Circuits*, New York: Mc-Graw Hill.

[2] Analog Devices, *Digital Signal Processing Applications*, New York: Prentice Hall.

[3] A. M. Alvarez, "Adaptive Algorithms for Active Attenuation of Acoustic Noise", *PhD Dissertation*, Heriot Watt University, 1986.

[4] M. O. Tokhi and R. R. Leitch, *Active Noise Control*, New York: Oxford University Press.

[5] S. J. Elliot, I. M. Stothers and P. A. Nelson, "A Multiple Error LMS Algorithm and ita Application to Acitve Control of Sound and Vibration", *IEEE transactions on ASSP*, vol. 35, no. 10, Oct 1987.

[6] S. J. Elliot and P. A. Nelson, *Active Control of Sound*, New York: Academic Press.

[7] S. Haykin, em Adaptive Filter Theory, New York: Prentice Hall.

[8] C. F. N. Cowen and P. M. Grant, i*Adaptive Filters*, New York: Prentice Hall.

Session 6
Design Techniques I

Chairman: Jody Gambles

Power Optimization in Logic Isomers [1]

Ramesh Panwar§†, David Rennels§†, and Leon Alkalaj§

§Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

†Department of Computer Science
University of California
Los Angeles, CA 90024

panwar, rennels, leon@telerobotics.jpl.nasa.gov

Abstract - **Logic isomers are labeled, 2-isomorphic graphs that implement the same logic function. Logic isomers may have significantly different power requirements even though they have the same number of transistors in the implementation. The power requirements of the isomers depend on the transition activity of the input signals. The power requirements of isomorphic graph isomers of n-input NAND and NOR gates are shown. Choosing the less power-consuming isomer instead of the others can yield significant power savings. Experimental results on a ripple-carry adder are presented to show that the implementation using the least power-consuming isomers requires approximately 10% less power than the implementation using the most power-consuming isomers. Simulations of other random logic designs also confirm that designs using less power-consuming isomers can reduce the logic power demand by approximately 10% as compared to designs using more power-consuming isomers.**

1 Introduction

The power consumed in CMOS VLSI primarily results from charging and discharging the node capacitances in the VLSI circuit. The implementation of a logic function can be repsented by a labeled graph. There are several different implementations possible even if we constrain the implementations to use the same number of transistors. Logic isomers are 2-isomorphic graphs that implement the same function. The isomers of a logic function have the same number of transistors but may have different CV^2f power requirements depending on the transition activity of the inputs. This paper addresses the power requirements of isomers of n-input NAND and NOR gates which can be represented by isomorphic graphs. The power requirements of general 2-isomorphic graph isomers will be the topic of a future paper. We show that a 2-input NAND gate has two isomers which may have different power requirements because the inputs to the gate are not equivalent from a consideration of the CV^2f power requirements even though they may be logically equivalent. We also show expressions for the power requirements of isomers of n-input NAND and NOR gates. We then present simulation results of a ripple-carry adder to show how the different power requirements of the isomers may lead to significant power savings by a logic designer who chooses to implement the design with the least power-consuming isomers based on transition

[1] Partial support for this work was provided by the Office of Naval Research under grant N00014-91-J-1009 at UCLA

activity simulations. The simulations were performed using a prototype TRansition ACtivity SIMulator called TRACSIM which we have developed for the purpose of studying transition activity in VLSI designs. Our simulation results on the ripple-carry adder and other random logic designs show that VLSI designers may be able to reduce the CV^2f power demand by approximately 10% by choosing the least power-consuming isomers in the design.

2 CV^2f Power

2.1 Node Activity

The CV^2f power [1] dissipated by a node depends on the switching behavior of the node. If the node switches an average of δ times every cycle, then the power dissipated by the node is δCV^2f. One needs to take into account the dynamic transitions rather than the logical transitions made by the node. To illustrate the difference between dynamic transitions and logical transitions, consider the circuit shown in Figure 1. Let us assume that the inputs (a, b, and c) come from registers or flip-flops that have the same output delay so that they all change at some time instant t_0 in the clock cycle under consideration. Let us assume that a, b, and c were all 0 before the start of the current clock cycle and that they all change to 1 in the current clock cycle. Prior to the start of the clock cycle, the value of signal y is 1. Assuming the two NAND gates have the same delay (Δ), signal y will change to 0 at time $t_0 + \Delta$ when the effect of the change of input c propagates to the output y. Signal y will then change back to 1 at time $t_0 + 2\Delta$ when the effect of the changes at the a and b inputs propagates to the output y. Signal y thus undergoes two dynamic transitions (manifested by a "glitch") even though it does not undergo any logical transition during the cycle under consideration. In other words, a node may logically switch only once during a cycle. However, a node may dynamically switch multiple times before it settles in a machine cycle depending on the arrival of the inputs.

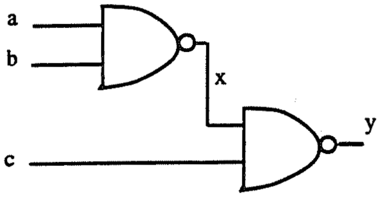


Figure 1: Example circuit for dynamic transitions

Definition 1 *The transition activity (δ_x) of a node (x) is defined to be the average number of dynamic transitions made by the node in a single cycle.*

2.2 Logic Isomers

A CMOS logic circuit may be represented by a labeled graph. The graph representing a CMOS circuit is constructed such that:

- The vertices in the graph represent the source/drain connections of the transistors in the circuit.
- The edges in the graph are the transistors that connect particular source-drain vertices.

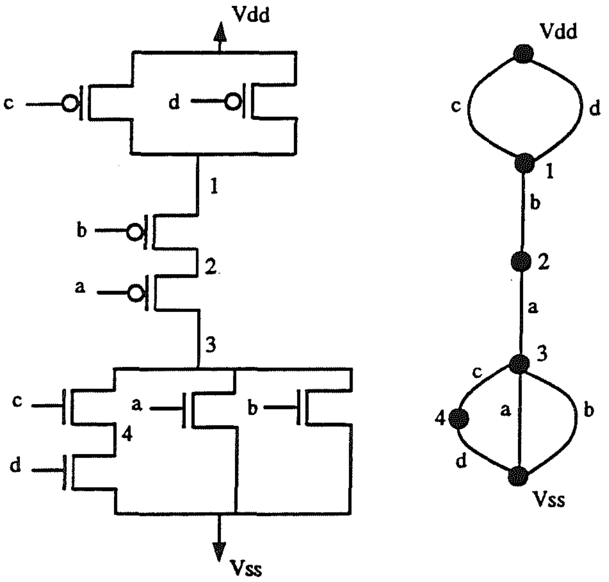


Figure 2: 4-input function

Figure 2 shows a circuit which implements the 4-input function $f(a,b,c,d) = a'b'c' + a'b'd'$ along with its labeled graph. Consider a 2-input NAND function of two inputs x and y. Figure 3 shows one implementation of the NAND function along with its graph representation. Figure 4 shows another implementation of the NAND function along with the graph representation. The two graphs are isomorphic but the two implementations may have different power requirements depending on the transition activity of the inputs. The two different isomorphic graphs implementing the NAND function are referred to as logic isomers of the NAND function. There may be non-isomorphic graph implementations of a function

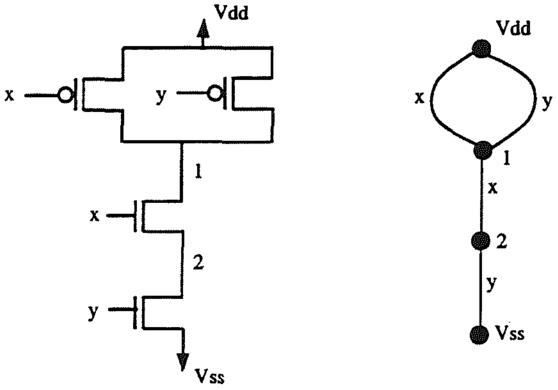


Figure 3: 2-input NAND Isomer 1

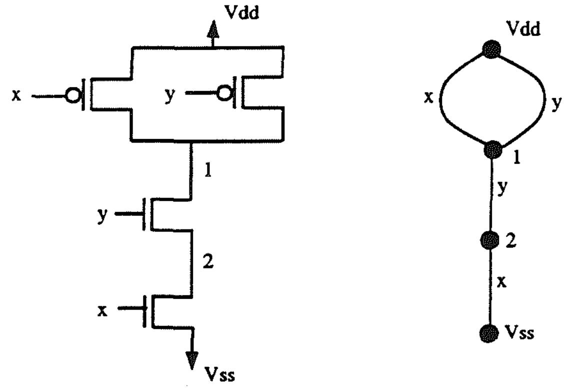


Figure 4: 2-input NAND Isomer 2

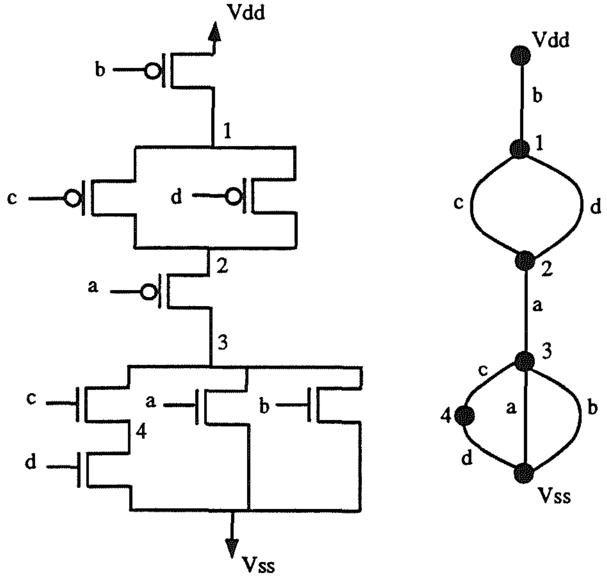


Figure 5: 4-input function

that have the same number of transistors (edges). An example of a function which has non-isomorphic graph implementations is the 4-input function shown in Figure 2. Another implementation of the function is shown in Figure 5 along with its graph representation. The two implementations of the 4-input function are 2-isomorphic [2]. This paper will address power requirements of only n-input NAND or NOR function realizations which have isomorphic graphs; the power requirements of general 2-isomorphic graph functions will be the subject of a later paper. However, for the sake of completeness, we provide the definition of a 2-isomorphic graph:

Definition 2 *Two graphs G_1 and G_2 are said to be 2-isomorphic if they have circuit correspondence which implies that there is a one-to-one correspondence between the edges of G_1 and G_2 and a one-to-one correspondence between the circuits of G_1 and G_2, such that a circuit in G_1 formed by certain edges of G_1 has a corresponding circuit in G_2 formed by the corresponding edges of G_2, and vice versa.*

Note that isomorphism is a special case of 2-isomorphism. We are now in a position to provide a formal definition of logic isomers.

Definition 3 *Logic isomers are labeled, 2-isomorphic graph realizations of a logic function.*

Definition 4 *The isomeric cardinality of a logic graph is the total number of logic isomers of the graph.*

Thus, an n-input NAND gate has an isomeric cardinality of $n!$. The circuit shown in Figure 2 has an isomeric cardinality of 12.

2.3 NAND Isomers

Consider the 2-input NAND gate for the purpose of illustrating the differences in power dissipation in logic isomers. Let us consider the behavior of the isomer in Figure 3 for input switching activity. Both nodes 1 and 2 discharge if signal x is 1 while y switches from 0 to 1. Also, both nodes 1 and 2 will charge if signal x is 1 while y switches from 1 to 0. Now consider the isomer shown in Figure 4. Only node 1 discharges if signal x is 1 while y switches from 0 to 1. Similarly, only node 1 charges if signal x is 1 while y switches from 1 to 0. If the capacitances of nodes 1 and 2 are denoted by C_1 and C_2, then the amount of energy required by the aforementioned operations in the two isomers are $(C_1 + C_2)V^2$ and C_1V^2 respectively. The power dissipated by the aforementioned operations is given by $\delta_y(C_1 + C_2)V^2 f$ and $\delta_y C_1 V^2 f$ in the two isomers since the operations occur with a frequency of $\delta_y f$. The power dissipated in the two isomers due to the switching activity of signal x is given by $\delta_x C_1 V^2 f$ and $\delta_x(C_1 + C_2)V^2 f$ respectively. Thus the total power required by the two isomers is given by:

$$P_1 = \delta_x C_1 V^2 f + \delta_y (C_1 + C_2) V^2 f$$
$$P_2 = \delta_x (C_1 + C_2) V^2 f + \delta_y C_1 V^2 f$$

where P_1 is the power dissipated in isomer 1 and P_2 is the power dissipated in isomer 2. The two isomers may thus have different power requirements depending on the transition activity δ_x and δ_y of signals x and y. If $\delta_x > \delta_y$, then the first isomer is preferable to the second to conserve power.

The values of the node capacitances C_1 and C_2 are as follows:

$$C_1 = 3C_d + C_L$$

$$C_2 = 2C_d$$

where C_d is the diffusion capacitance of a transistor and C_L is the load capacitance driven by the gate. The value of C_L will depend on the gate fanout and the interconnect capacitance (which depends on the net routing). The ratio of the power required by one isomer to the power required by the other is given by:

$$r_P = \frac{\delta_x(5C_d + C_L) + \delta_y(3C_d + C_L)}{\delta_x(3C_d + C_L) + \delta_y(5C_d + C_L)}$$

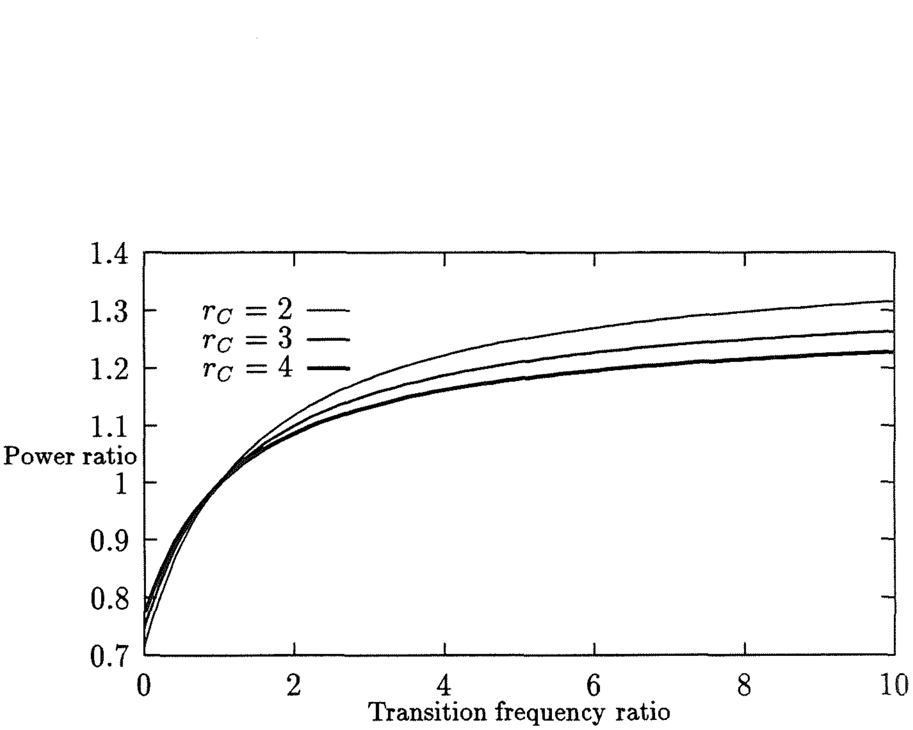


Figure 6: 2-input NAND power ratio

If we choose r_C to represent the ratio C_L/C_d, then the ratio of the power required by the isomers is given by:

$$r_P = \frac{5\delta_x + 3\delta_y + r_C(\delta_x + \delta_y)}{3\delta_x + 5\delta_y + r_C(\delta_x + \delta_y)}$$

To understand the typical value of the parameter r_C, we look at the Honeywell RICMOS process [3] which is used for ASIC design at JPL. The Honeywell RICMOS process has a typical routing capacitance of $1.4C_g$ for each load driven by the gate (where C_g is the gate capacitance of a transistor). If we assume that the average fanout is 4, then the typical value of C_L is $9.6C_g$. Thus the typical value of r_C is $9.6C_g/C_d$. Typically, $C_d = 4C_g$, which implies that a value of approximately 2.5 is typical for the parameter r_C.

2.4 Power Ratio

Definition 5 *The power ratio of a circuit is the ratio of the power required by the implementation using the most power-consuming isomer to the power required by implementation using the least power-consuming isomer.*

For a two-input NAND gate, the power ratio is given by:

$$r_P = \frac{5\delta_1 + 3\delta_2 + r_C(\delta_1 + \delta_2)}{3\delta_1 + 5\delta_2 + r_C(\delta_1 + \delta_2)}$$

where we have assumed that $\delta_1 > \delta_2$. Figure 6 shows the typical power ratios of a two-input NAND gate for various values of the parameter r_C. The asymptotic value of the power ratio (the value when one signal has a transition activity that is infinitely times that of the other signal) is given by $(5 + r_C)/(3 + r_C)$ which implies that the upper-bound on the power ratio for a 2-input NAND gate is 5/3. The power ratios are plotted as functions of the ratio of the transition activity of the two inputs ($r_\delta = \delta_1/\delta_2$).

The power ratio of an n-input NAND gate can be shown to be:

$$r_P = \frac{(3n + 1 + r_C)\sum_{i=1}^{n} \delta_i - 2\sum_{i=1}^{n} i\delta_i}{(n - 1 + r_C)\sum_{i=1}^{n} \delta_i + 2\sum_{i=1}^{n} i\delta_i}$$

where the activities are such that $\delta_1 > \delta_2 > ... > \delta_n$. It may be shown that an n-input NOR gate has a similar expression for the power ratio.

The power ratio of a 3-input NAND gate (or a 3-input NOR gate) is thus:

$$r_P = \frac{8\delta_1 + 6\delta_2 + 4\delta_3 + r_C(\delta_1 + \delta_2 + \delta_3)}{4\delta_1 + 6\delta_2 + 8\delta_3 + r_C(\delta_1 + \delta_2 + \delta_3)}$$

where $\delta_1 > \delta_2 > \delta_3$.

The power ratio of a 4-input NAND gate (or a 4-input NOR gate) can be shown to be:

$$r_P = \frac{11\delta_1 + 9\delta_2 + 7\delta_3 + 5\delta_4 + r_C(\delta_1 + \delta_2 + \delta_3 + \delta_4)}{5\delta_1 + 7\delta_2 + 9\delta_3 + 11\delta_4 + r_C(\delta_1 + \delta_2 + \delta_3 + \delta_4)}$$

To choose the best isomer of an n-input NAND or NOR gate, we now propose two rules which can be used by the logic designer (or a logic synthesis tool) to optimize the power dissipation in a logic design.

Rule 1 *If the logic graph has signals in series in the n-logic block, then the series signals should be ordered with respect to the signal transition activity; the lowest activity signal should be closest to the ground pin.*

Rule 2 *If the logic graph has signals in series in the p-logic block, then the series signals should be ordered with respect to the signal transition activity; the lowest activity signal should be closest to the* V_{cc} *pin.*

3 Experimental Results

3.1 TRACSIM

TRansition ACtivity SIMulator (TRACSIM) is a tool that we developed to study the transition activity in VLSI circuits. The input to the simulator is a netlist describing the circuit along with the transition activities of the circuit inputs. We use a simplified timing model of circuit behavior. We assume that all the gates have the same delay and that two signals changing at the inputs to a gate can cause at most one transition only on the output node. The latter assumption is quite valid since SPICE simulations confirmed that the output of a gate undergoes at most one transition even if two or more inputs change as long as the inputs change within a certain time window. The output from TRACSIM is a list of transition activities at the nodes of the circuit along with the power ratio of the circuit graph.

3.2 Ripple-carry Adder

TRACSIM was used to simulate a ripple-carry adder for various number of inputs (4-bit, 8-bit and 16-bit inputs) with different transition activities on the input signals. We simulated the ripple-carry adder with the same transition probability for each input to the adder (except the carry-in input which was set to 0) for 10000 cycles. Typically, the inputs to the adder are driven by register outputs and hence the transition activity of the inputs is less than 1. Each stage of the ripple adder consists of two circuits: one to compute the carry and the other to compute the sum. The circuit for computing the carry consists of three 2-input NAND gates and a 3-input NAND gate as shown in Figure 7. The circuit for computing the sum consists of four 3-input NAND gates and a 4-input NAND gate as shown in Figure 8. Of the signals internal to the ripple adder, the carry signal has a fanout of 6 and the outputs of the inverters have a fanout of 2 each. Assuming that the routing capacitance for each load is $1.5C_g$ and that the sum outputs of the adder drive unit loads, the average r_C for the ripple adder is approximately 1. Table 1 summarizes the power ratio of the ripple-carry adder for a 4-bit adder (RA4), an 8-bit adder (RA8), and a 16-bit adder (RA16) for various values of the input transition activity (δ_i). The table shows that the power ratio of the adder varies from 1.08 to 1.12 which implies that the implementation using the least power-consuming

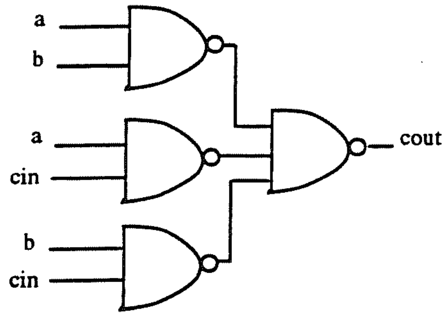


Figure 7: Carry circuit

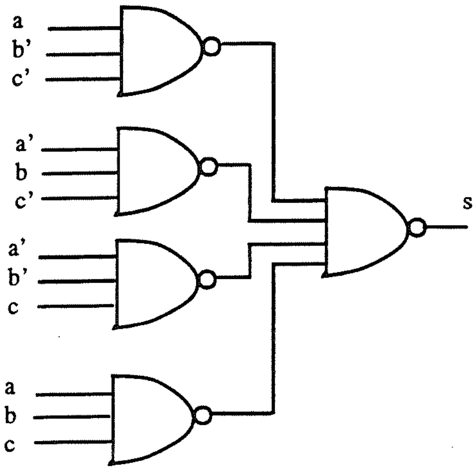


Figure 8: Sum circuit

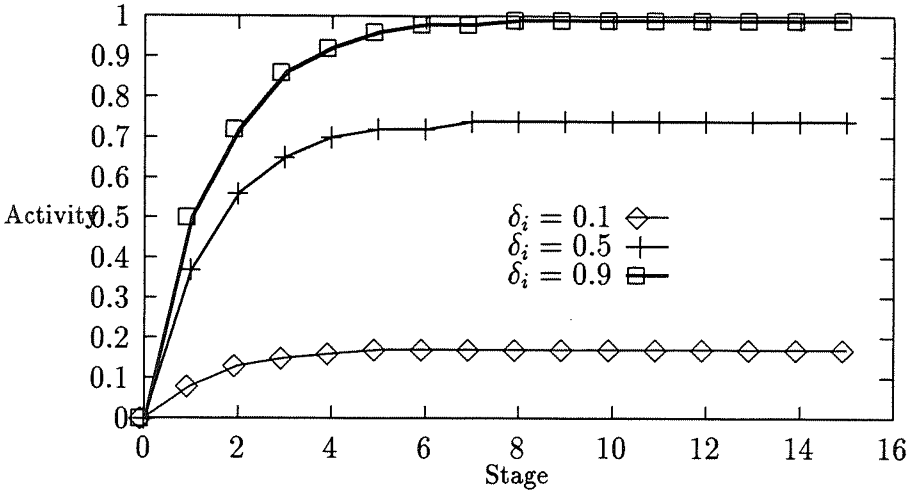


Figure 9: Ripple adder carry transition activity

isomers offers a power saving of approximately 10% over the implementation using the most power-consuming isomers. Figure 9 shows the transition activity of the carry signals of the ripple-carry adder while Figure 10 shows the transition activity of the sum outputs of the adder.

δ_i	RA4	RA8	RA16
0.1	1.12	1.12	1.11
0.3	1.12	1.12	1.11
0.5	1.12	1.11	1.11
0.7	1.12	1.10	1.09
0.9	1.12	1.09	1.08

Table 1: Ripple adder power ratio

3.3 Random Logic

We simulated various random logic designs with various input transition activites for $r_C = 2.5$ and found that the power ratio of 1.1 was typical for most logic designs. This implies that a power saving of 10% is possible by just a judicious choice of isomers by the logic designer.

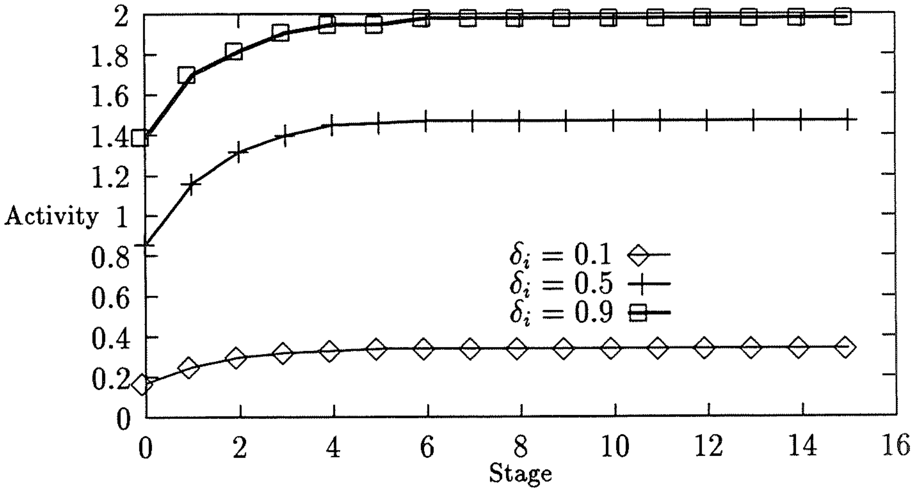


Figure 10: Ripple adder sum transition activity

4 Future Work

A couple of interesting problems remain for future research. The first problem that will be looked into is the generalization of the results on power ratio and the two rules of thumb to handle 2-isomorphic graphs. The second problem that will be looked into is the dependence of the power ratio on the logic depth of the circuit. We have not simulated a sufficient number of random logic designs at the time of this publication to make any conclusions regarding this problem.

5 Conclusion

We have presented a novel way of optimizing power in CMOS VLSI. We have shown that the inputs to a gate may not be equivalent from a consideration of the CV^2f power requirements even though they may be logically equivalent. Simulation results for a ripple-carry adder were presented which showed a power saving of approximately 10% in a low-power isomer design as compared to a high-power isomer design. This was also confirmed by simulating other random logic designs. The only side-effect that results from using this technique is that logically equivalent gate inputs may no longer be considered swappable by routing tools. Other than this side-effect, the power optimization offered by the technique is free.

References

[1] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design.* Reading, MA: Addison-Wesley, 1988.

[2] N. Deo, *Graph Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1974.

[3] *The Honeywell RICMOS ASIC Library Data Book.* 1991.

Computation of Signal Delays in RC Networks

Juan Carlos Hidalgo Paliath Narendran Seth Chaiken
Department of Computer Science
State University of New York at Albany
Albany, NY 12222

Abstract - **We present a model for signal delay computation in RC networks. The strength of the paradigm is its generality and simplicity. Our definition of delay is applicable to RC meshes with potential resitive attenuating paths to ground. Our algorithms can also be applied to undriven circuits (static charge sharing) and circuits with initial charge. To compute the delays, each node in the network is explored locally to derive a system of sparse linear equations. The solutions of the system are delay values based on the Elmore time constant at each point in the circuit.**

1 Introduction

Current approaches to MOS modeling use RC networks as a simplification from designs with active elements to circuits with purely passive components. An *RC network* is an *RC mesh* where resistors can be grounded. An *RC mesh* is any circuit consisting of two-terminal resistors and capacitors driven by independent sources, such that one side of every capacitor and every source is grounded and no resistor is grounded. If we assume that all voltage sources are normalized, steady state voltages and the time taken to achieve them can be estimated. Penfield and Rubinstein [10] provided efficient algorithms to compute bounds of the Elmore delay (also known as the Elmore time constant) [6] at the nodes of meshes with tree topologies. An *RC tree* is an RC mesh without a resistive cycle.

Lin and Mead [7] proposed a methodology for delay computation applicable to RC meshes. Their TREE algorithm can efficiently compute the Elmore delay for tree networks, and for more general topologies they proposed an algorithm to iteratively decompose the network into an RC tree. Nonetheless their paradigm cannot be applied to static charge sharing situations or networks with resistive paths to ground. Chan and Schlag [4] showed how to obtain bounds of the Elmore delay for RC meshes by bounding the entries of the node-conductance matrix of the circuit. For series-parallel topologies (i.e. RC meshes where resistors are interconnected in a series-parallel manner) Caisso, Cerny and Rumin [3] have found an efficient recursive technique to compute the Elmore time constant.

We present a simple method to compute delays in RC networks. By performing local examinations at each point in the network, we produce a system of sparse linear equations whose solutions corresponds to delay values based on the Elmore time constant.

Our method has three main advantages:

1. Our definition of delay is applicable to
   - RC meshes and networks with initial charge.
   - Charge sharing situations.

2. The derivation of the equations can be done in a uniform fashion since only local inspections of a node's neighborhood are needed.

3. The final computation consists of solving systems of sparse linear equations for which there are powerful automatic tools. The solutions for the system of equations can be computed numerically or symbolically depending on how the parameters of the network are specified.

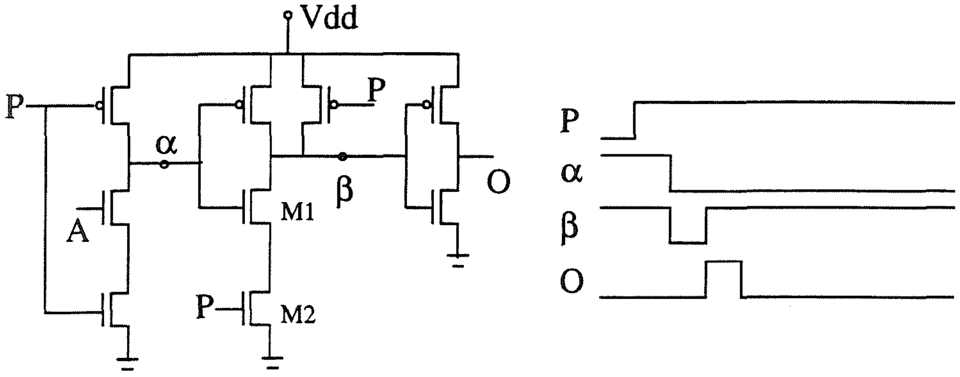
2 An RC model for MOS circuits

The timing model we have developed is designed to compute approximations of the time taken for each node of an Resitive-Capacitive network to achieve a percentage of its final voltage value. Our ultimate goal is to use these ideas to model digital MOS circuits, therefore we start by presenting the somewhat standard approach of modeling MOS circuitry by RC networks. Approximation of MOS circuitry by RC networks has been used by several authors [2, 10, 7, 11].

The first approximation step is to replace all active components with resistors and to linearize all capacitances. The resulting circuit is an RC network where each resistance approximates a conducting transistor channel and each capacitor approximates the gate capacitance of a transistor. Although for certain designs these approximations are somehow crude, many useful MOS circuits can be modeled correctly. The qualitative and quantitative nature of low level modeling using these ideas has been studied by [2, 5] and is a matter of some of our current research.

If switch-level modeling is desired, our timing model can be combined with models like that of Bryant [2]. In that model the behavior of the circuit is determined by an iterative process; at each cycle the circuit is seen as a graph where the edges stand for the channels of conducting transistors and the vertices stand for the circuit nodes. The model computes the values at the nodes and produces a new graph as a result of the transistors that change their state. The iteration is performed until the graphs do not change (i.e., a fixed point is reached) or a predetermined upper bound of the number of iterations is exceeded (the model assumes that the underlying clock cycle is long enough to let the circuit stabilize). Transistors are modeled as having a unit delay, which means that each transistor reacts to a change at its gate after one unit simulation step (if the transistor is not declared to have zero delay); the nodes assume their new value instantly.

The COSMOS simulator [1] uses Bryant's MOSSIM [2] as the underlying low level model with the unit delay timing model. We use a design extracted from the COSMOS user manual as an example of how this simple timing model can produce an incorrect behavior. The circuit on Figure 1 (a) consists of a precharged NOR gate followed by a NAND gate (to force β to 1 during precharge), followed by an inverter. The inputs to the NOR gate are the gates of transistors connected in parallel with the one having gate A. The problem is the race that occurs between the opening of M1 as node α discharges and the closing of M2 which would otherwise pulldown node β when A equals 1 and P goes to 1. If transistors are modeled with unit delays the race is won by the pulldown and β glitches to 0 for a unit delay, as shown in Figure 1 (b). In [1] we are told how to avoid this problem by assigning



(a) (b)

Figure 1: Simulation under unit delay timing model

zero delay to some of the transistors. This is not a satisfatory solution since it is ah-hoc and requires previous knowledge of the behavior of the circuit.

The choice of a switch level model is not bound to any formalism in particular; we chose Bryant's low level model for explanation purposes only. Our intent is to include the paradigm presented in the following sections in our own verification model.

3 Definitions and Approximations

In [6] Elmore defines "delay" as the first moment of the response function $y(t)$. This definition is equal to $T_D^0 = \int_0^\infty t y'(t)\, dt$ where $y'(t)$ is $y(t)$'s first derivative. We borrow the notation used in [7] to point out the zero initial charge condition (that is, the superscript in T_D^0). The intuition behind this definition is explained in [14]: the step response function is assumed to rise monotonically from zero to 1, hence the impulse response is a nonnegative function with unit area. Since the impulse response has the properties of a probability density, using T_D^0 as an approximation for delay is like approximating a probability density function by its mean value.

The circuit equations for a RC network can be formulated in the following matrix form: [V(t)] = [R] [C] [V'(t)]. A quantitative characterization of the Elmore time constant is found by using the entries from the matrix $[RC]$; the Elmore time constant at node i is numerically equal to $\sum_{k=1}^{N} R_{i,k}C_k$ where $R_{i\,k}$ correspond to the $(i,k)^{th}$ entry of the resistance matrix $[R]$ and C_k is the capacitance value at node k. This characterization is particularly useful for tree networks since the entries of the resistance-capacitive matrix are very easy to compute. For an RC tree the entries of the $[R]$ matrix can be computed by inspection: the entry $R_{i,j}$ is simply the sum of the resistances along the path obtained by intersecting the path between node i and the root (i.e. input node) with the path between node j and the root.

Perhaps the most important characteristic of Elmore's delay is its relationship with the Laplace transform of the response. If $y(t)$ stands for the transient response of a node whose final value is different from 0, then $g(s) = \mathcal{L}(y(t))$ can be expressed in the following normal form: $g(s) = \frac{a_0+a_1 s+a_2 s^2+\cdots+a_m s^m}{s(b_0+b_1 s+b_2 s^2+\cdots+b_n s^n)}$.

If there is no initial charge in the circuit, the Elmore time constant is numerically equal to $\frac{b_1}{b_0} - \frac{a_1}{a_0}$. $g(s)$ is in general complicated and difficult to compute, but $\frac{b_1}{b_0} - \frac{a_1}{a_0}$ can be efficiently calculated for any node in a RC tree network; moreover the computation can be done in a hierarchical manner.

Lin and Mead generalized Elmore's definition of delay in order to handle RC networks with initial charge and arbitrary topologies [7]. Their definition of delay is the following: $T_D = \int_0^\infty 1 - y(t)\, dt$. This definition is more applicable than Elmore's since it provides a consistent delay value even when there is initial charge distributed in the network (which translates into a potential nonmontonic response). Although their definition of delay is different from Elmore's, the two are equivalent in the cases of zero initial charge.

Our goal in this paper is to define delay in a way applicable to more general situations and topologies than those considered by Lin and Mead. Moreover we want to maintain the definition consistent with the ones above, which means that when the topology of the network (and the initial conditions) correspond to one of the situations considered by Lin and Mead's model, our concept of delay corresponds to that of theirs. Finally, we are interested on providing an efficient form of computation since otherwise our model would be of little practical use.

4 A timing model for RC networks

As we saw in the previous section, Lin and Mead's definition of delay extended Elmore's in order to cope with non monotonic responses. The problem with their definition is that it is only consistent with situations where the steady state of the network is Vdd. We extend their definition so it can be applied to static charge sharing situations and networks with resistive paths to ground.

Definition 1 *We define delay at a node as the area between the response function of the node $y_i(t)$ and its final voltage scaled by the final value, that is $\frac{\int_0^\infty k_i - y_i(t)dt}{k_i}$ where k_i is the final value of node i ($k_i = \lim_{t\to\infty} y(t) \neq 0$).*

Given the fact that only the terms $\frac{a_1}{a_0}$ and $\frac{b_1}{b_0}$ of $g(s)$ (assuming that the final value is not 0) are needed to characterize T_D^0, we can approximate the Laplace transform of the response, for small s, by ignoring all but the linear terms in $g(s)$ and using the fact that $\frac{1}{1+x} \approx 1 - x$:

$$g(s) = \frac{a_0}{b_0} \frac{1 + \frac{a_1}{a_0}s + \frac{a_2}{a_0}s^2 + \cdots + \frac{a_m}{a_0}s^m}{s(1 + \frac{b_1}{b_0}s + \frac{b_2}{b_0}s^2 + \cdots + \frac{b_n}{b_0}s^n)} \approx \frac{a_0}{b_0}\left[\frac{1}{s\left(1 + s\left(\frac{b_1}{b_0} - \frac{a_1}{a_0}\right)\right)}\right] \tag{1}$$

Note that as a result of the Final Value Theorem $sg(s)\mid_{s\to 0} = y(t)\mid_{t\to\infty} = \frac{a_0}{b_0}$ and therefore the final voltage at a node is equal to $\frac{a_0}{b_0}$,provided that the steady state value is not 0. When the steady state value of the network is 0 the Laplace transform of the response has a different form, and therefore the above approximations are not valid. These situations will be considered later in the paper. We define as the general expression of the approximate transform at node i as $v_i(s) = \frac{k_i}{s(1+s\tau_i)}$ where τ_i equals $\frac{b_1^i}{b_0^i} - \frac{a_1^i}{a_0^i}$ and k_i is the final voltage value of node i, i.e. $\frac{a_0^i}{b_0^i}$. In the following section we will show that for a given node this expression can be written in terms of the final values and delays of its neighbors.

Up to this point we have two concepts of delay: one in terms of area, and one in terms of coefficients of linear terms of the Laplace transform of the response; the next lemma shows how the two concepts are equal.

Lemma 2 $\tau_i = \frac{b_1^i}{b_0^i} - \frac{a_1^i}{a_0^i} = \frac{\int_0^\infty k_i - y_i(t)dt}{k_i}$

If the final value of the network is Vdd (as it is the case in driven RC meshes, i.e. networks with no resistive paths to ground), then the attenuation factor k_i equals 1 and our definition of delay is equivalent to (3). Moreover, if there is no initial charge in the network, our definition of delay coincides with Elmore's since the response is monotonic with final value 1 (hence $a_0 = b_0 = 1$).

We have stressed the fact that the final value at a node has to be different than zero for our definitions to be meaningful. The problem arises due to the form of the Laplace transforms at the nodes of circuits whose final state is the ground potential (we call these designs, discharging circuits). The normal form of the transform at a node of a discharging circuit is the following: $g(s) = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_m s^m}{b_0 + b_1 s + b_2 s^2 + \cdots + b_n s^n}$.

Since the final state k of a discharging circuit is 0, our current definition of delay is not applicable. To fix this problem we redefine delay (for discharging cicuits only) to the following: $\int_0^\infty y(t)dt$

This definition is consistent with our previous concept of delay because it corresponds to the area between the response function and the final state (in this case the x axis), but we make no claims about its qualitative interpretation other than that for simple circuits it corresponds to the common concept of "delay".

In the spirit of our first definition of delay, we want to relate the area interpretation of delay with the Laplace transform of the response. The new delay is numerically equal to the ratio of the constant terms of the polynomials in the transform.

Lemma 3 *Let n be a node of a discharging circuit with transform g(s), then* $\int_0^\infty y(t)dt = \frac{a_0}{b_0}$

To maintain consistency with our original definitions we use the symbol τ as the reference for delay, hence we can make the following manipulations with g(s):

$$g(s) = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_m s^m}{b_0 + b_1 s + b_2 s^2 + \cdots + b_n s^n} = \frac{\tau + a_1' s + a_2' s^2 + \cdots + a_m' s^m}{1 + b_1 s + b_2 s^2 + \cdots + b_n s^n} \quad (2)$$

where $a_i' = \frac{a_i}{b_i}$ for all $1 \leq i \leq m$.

4.1 Computation of time constants

In this section we describe a method of computing delays at each point of an RC network. The idea is to express the approximated transform of a node as a function of the final values and delays of its neighbors. These formulas will form a system of sparse linear equations from which the delays can be calculated. Figure 2 describes the three situations that we consider:

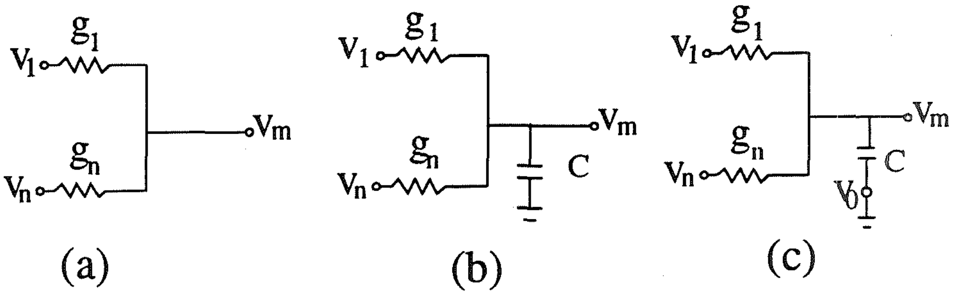


Figure 2: Possible connections to a node

(a) A node with no capacitance.

(b) A node with a grounded capacitor connected to it (no initial charge).

(c) A node with a grounded capacitor with initial charge.

Theorem 4 *If n nodes with transients $v_i(s)$ for $1 \leq i \leq n$ are connected to a point m through resistors with conductances g_i for $1 \leq i \leq n$ (see Figure 2 (a)), then $v_m(s) = \frac{k_m}{s(1+s\tau_m)}$ with*

$$k_m = \frac{k_1 g_1 + \cdots + k_n g_n}{g_1 + \cdots + g_n} \text{ and } \tau_m = \frac{k_1 g_1 \tau_1 + \cdots + k_n g_n \tau_n}{k_1 g_1 + \cdots + k_n g_n}$$

Proof

From standard circuit analysis we can see that $v_m(s) = \frac{v_1(s)g_1 + \cdots + v_n(s)g_n}{g_1 + \cdots + g_n}$. Since voltages are normalized, we have the following derivation

$$\begin{aligned}
v_m(s) &= \frac{\frac{k_1 g_1}{s(1+s\tau_1)} + \cdots + \frac{k_n g_n}{s(1+s\tau_n)}}{g_1 + \cdots + g_n} \\
&= \frac{1}{s(g_1 + \cdots + g_n)} \left[\frac{k_1 g_1}{1+s\tau_1} + \cdots + \frac{k_n g_n}{1+s\tau_n} \right] \\
&= \frac{1}{s(g_1 + \cdots + g_n)} \left[\frac{\sum_{i=1}^{n} k_i g_i \left[(1+s\tau_1)\ldots(1+s\tau_{i-1})(1+s\tau_{i+1})\ldots(1+s\tau_n) \right]}{(1+s\tau_1)\ldots(1+s\tau_n)} \right] \\
&= \frac{1}{s(\sum_{i=1}^{n} g_i)} \left[\frac{k_1 g_1 \left[1 + s(\tau_2 + \cdots + \tau_n) + A_1 \right] + \cdots + k_n g_n \left[1 + s(\tau_1 + \cdots + \tau_{n-1}) + A_n \right]}{(1+s\tau_1)\ldots(1+s\tau_n)} \right]
\end{aligned}$$

where the $A_i's$ represent higher order (non-linear) terms in s. This last expression can be further reduced as follows

$$\begin{aligned}
v_m(s) &= \frac{k_1 g_1 + \cdots + k_n g_n}{s(g_1 + \cdots + g_n)} \left[\frac{1 + \frac{\sum_{i=1}^{n} k_i g_i \, s(\tau_1 + \cdots + \tau_{i-1} + \tau_{i+1} + \cdots + \tau_n) + \sum_{i=1}^{n} A_i}{k_1 g_1 + \cdots + k_n g_n}}{(1+s\tau_1)\ldots(1+s\tau_n)} \right] \\
&\approx \frac{k_1 g_1 + \cdots + k_n g_n}{g_1 + \cdots + g_n} \left[\frac{1 + s\frac{\sum_{i=1}^{n} k_i g_i \, (\tau_1 + \cdots + \tau_{i-1} + \tau_{i+1} + \cdots + \tau_n)}{k_1 g1 + \cdots + k_n g_n}}{s(1 + s(\tau_1 + \cdots + \tau_n))} \right]
\end{aligned}$$

if we drop all but the linear terms in s.

Hence

$$k_m = \frac{k_1 g_1 + \cdots + k_n g_n}{g_1 + \cdots + g_n}$$

and

$$\begin{aligned} \tau_m &= (\tau_1 + \cdots + \tau_n) - \left[\frac{\sum_{i=1}^{n} k_i g_i (\tau_1 + \cdots + \tau_{i-1} + \tau_{i+1} + \cdots + \tau_n)}{k_1 g_1 + \cdots + k_n g_n} \right] \\ &= \frac{[(\tau_1 + \cdots + \tau_n)(k_1 g_1 + \cdots + k_n g_n)] - \sum_{i=1}^{n} k_i g_i (\tau_1 + \cdots + \tau_{i-1} + \tau_{i+1} + \cdots + \tau_n)}{k_1 g_1 + \cdots + k_n g_n} \\ &= \frac{k_1 g_1 \tau_1 + \cdots + k_n g_n \tau_n}{k_1 g_1 + \cdots + k_n g_n} \end{aligned}$$

□

The expressions for final value and delay of the previous theorem appear as part of the formulas for final value and delay of the next two cases to consider. For this reason, we define abbreviations for k_m and τ_m.

Definition 5 *Let the expressions of k_m and τ_m of Theorem 4 be referred as k and τ respectively.*

We expand the previous result to include nodes with capacitance (as it is the case for every non input node in an RC network). The proof of the next theorem is similar to the previous one therefore we do not include the argument.

Theorem 6 *If n nodes with transients $v_i(s)$ for $1 \leq i \leq n$ are connected to a point m, with capacitance C with no initial charge, through resistors with conductances g_i for $1 \leq i \leq n$ (see Figure 2 (b)), then $v_m(s) = \frac{k_m}{s(1+s\tau_m)}$ with*

$$k_m = k \text{ and } \tau_m = \tau + \frac{C}{g_1 + \cdots + g_n}$$

The previous theorem gives us the foundations of an algorithm to compute the time constants for all points of an RC network. The function that describes the delay at a point is a linear combination of the delays of its neighbors. Since the data nodes are assumed to have zero delay, the delay functions form a system of n equations with n variables, where n is the number of nodes in the network.

We extend our results to include initial distribution of charge, the last case to be considered in this section. We are going to look at capacitances with initial charge as being discharged capacitors in series with a voltage source whose magnitude equals the initial voltage stored at the node (see (c) of Figure 2). Like in the previous theorem we do not include the proof of the following claim because of the similarities with the proof of Theorem 4.

Theorem 7 *If n nodes with transients $v_i(s)$ for $1 \leq i \leq n$ are connected to a point m, with capacitance C that stores an initial voltage V_0, through resistors with conductances g_i for $1 \leq i \leq n$ (see Figure 2 (b)), then $v_m(s) = \frac{k_m}{s(1+s\tau_m)}$ with*

$$k_m = k \text{ and } \tau_m = \tau + \frac{C}{g_1 + \cdots + g_n} - \frac{CV_0}{k_1 g_1 + \cdots + k_n g_n}$$

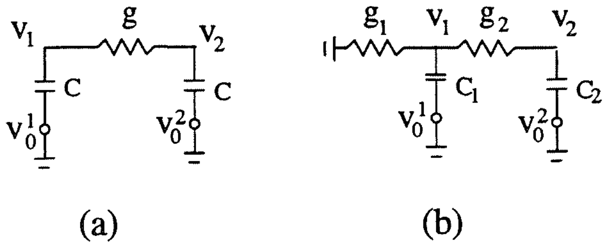


Figure 3: Examples of charge sharing and discharging circuits

Notice that if there is no initial charge (i.e. $V_0 = 0$) the expression for delay reduces to the one in the previous theorem (when initial charge was not assumed to be present). As with the previous cases, the methodology to compute the time constants consists of solving the system of equations for the final values and then solving the system for the delays.

4.2 Charge Sharing

Charge sharing situations pose a problem: the equations that are derived from the previous calculations will form a degenerate system. This arises due to the fact that there are no driving nodes in the network. To solve this problem a linearly independent equation has to be introduced; that equation is provided by the following lemma.

Lemma 8 *Let n be the number of nodes of an RC mesh with no driving nodes. Let c_i and τ_i be the corresponding capacitance and time constant for $1 \leq i \leq n$. The following equation always holds: $\sum_{i=1}^{n} c_i\tau_i = 0$*

To illustrate the usefulness of the previous lemma, let's consider the example in Figure 3 (a). In this case the equations for the final values are $k_1 = \frac{k_2 g}{g}$, $k_2 = \frac{k_1 g}{g}$ and therefore $k_1 = k_2$. On the other hand we have the equations for the delays $\tau_1 = \frac{k_2\, g\, \tau_2}{k_2\, g} + \frac{C}{g} - \frac{C\, v_0^1}{k_1\, g}$ and $\tau_2 = \frac{k_1\, g\, \tau_1}{k_1\, g} + \frac{C}{g} - \frac{C\, v_0^2}{k_2\, g}$.

By adding the previous equations we can conclude that $k_1 = \frac{v_0^1 + v_0^2}{2}$, which correspond to what one would expect: the total charge will be distributed evenly in the circuit.

Now, because of Lemma 8 we have that $\tau_1 = -\tau_2$ which intuitively is telling us that one capacitor charges and the other one discharges. We find that $\tau_1 = \frac{1}{2\, g}\left[\frac{C\, v_0^2}{(v_0^1 + v_0^2)}\right]$, as expected.

4.3 Discharging Circuits

As we have mentioned before, circuits like the one in Figure 3 (b) cannot be handled by the current theory. On the other hand these situations cannot be handled by Lin and Mead's model since their theory assumes that the steady state of the network is always Vdd. Although we are able to include a "delay constant" for these kind of situations, the results are not as elegant as the ones in the previous sections. Our concept of delay for networks where the steady state is 0 does not come with a natural qualitative interpretation. In the remaining of this section, we refer to "delay" defined as $\int_0^\infty y(t)dt$.

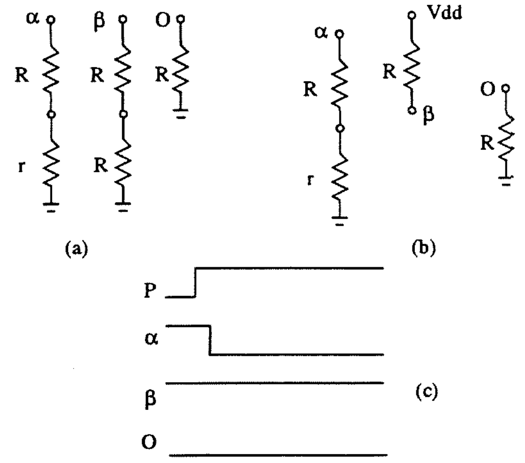


Figure 4: Simulation under variable delay step

Theorem 9 *Let n nodes of a discharging circuit be connected to a point m with capacitance C that stores an initial voltage V_0, through resistors with conductances g_i for $1 \leq i \leq n$ as in Figure 2 (c) (page 7). If τ_i $(1 \leq i \leq n)$ represent the delay of the nodes, then*

$$\tau_m = \frac{g_1\tau_1 + \cdots + g_n\tau_n + V_0C}{g_1 + \cdots + g_n}$$

To illustrate the previous theorem, let's concentrate on the circuit described in Figure 3 (b). The equations derived from Theorem 9 are the following $\tau_1 = \frac{g_2\tau_2 + v_1^0C_1}{g_1+g_2}$, $\tau_2 = \frac{\tau_1 g_2 + v_2^0 C_2}{g_2}$.

By solving the system we obtain the following values as delays: $\tau_1 = \frac{v_1^0C_1 + v_2^0C_2}{g_1} = R_1v_1^0C_1 + R_1v_2^0C_2$, and $\tau_2 = \frac{g_2(v_1^0C_1+v_2^0C_2)+g_1v_2^0C_2}{g_1g_2} = R_1v_1^0C_1 + R_1v_2^0C_2 + R_2v_2^0C_2$ which happen to correspond to the exact constant terms of the numerators of the transforms (when the functions are in the form described in (2)).

We are now in a position to revise the circuit in Figure 1. Figure 4 (a) describes the RC network computed by Bryant's simulator right after the signal P has gone up to Vdd. We can see that the situation at hand is of two discharging networks (with points α and β as the charged nodes) and one stable network with node O at ground potential. The two discharging lines can be analyzed in the same way that we studied the line in Figure 3 (b), but in this case the equivalents of node v_1 do not have capacitance. If we take the results for the delay values calculated above and make $C_1 = 0$ we obtain the following: $\tau_\alpha = C_\alpha v_\alpha^0(R+r)$, $\tau_\beta = C_\beta v_\beta^0(R+R)$

Assuming that $r < R$ and $C_\alpha = C_\beta$, we conclude that node α will change its value before node β. This means that the next RC network that the simulator will consider is the one described in Figure 4 (b), which is the correct behavior of the circuit. In Figure 4 (c) we present the voltage curves that the simulator would compute. Notice the absence of the glitches at nodes β and O.

5 Conclusions

We have developed a timing model for general RC networks. By extending Lin and Mead's notion of delay we are able to cope with static charge sharing and attenuations. On the computational side our method is also very efficient since the delays are the solutions of systems of sparse linear equations that are extracted from local inspections of the network. We can use one of various mathematical tools (i.e. MAPLE, MATHEMATICA) in order to find the solutions, numerically or symbolically.

An implicit assumption of the model is that we can detect charging, discharging and charge sharing situations (since different sets of equations need to be derived in each case). This problem can be solved at the level of the MOS simulator or verifier by calculating approximations of the "strongest" path in the network at each node. For example, if we use Bryant's model [2] as the underlying simulator, the "strength" of a node with the addition of some extra computation can be used to decide which kind of equations apply. By computing the transitive closure of the graph formed by the unblocked paths it is easy to decide if a node has a path to Vdd, Gnd or both. Connections to only Gnd represent a discharging situation and therefore the last set of equations should be used. If there are connections to Vdd and Gnd then the first set of equations are the right ones. One the other hand if the stength of the node is a capacitance value then we have a charge sharing situation at hand. Several models include this idea of "strength" as part of the approximations to calculate the final value of the node, although the "strength value set" varies in each paradigm. If other MOS models are used, there has to be some mechanism to detect what kind of situation applies to each node or our timing model cannot be used. Models like BIDS [9] and Winskel's model for MOS circuits [12, 13] include strength information suitable for our theory to be applicable.

We are currently working on implementations of our method to be used with general CMOS simulation and verification models. It is important to point out again that our timing model is suitable for any hardware simulation or verification paradigm that models circuits (implicitly or explicitly) as resistive networks with only lumped capacitors. For our own experimentation purposes we have developed a program that translates an RC network into a set of linear equations in a format readable by MAPLE, which we use later to find the delay values.

References

[1] Beatty, D., Brace, K., Bryant, R.E., Cho, K., Huang, L., *User's Guide to COSMOS: a COmplied Simulator for MOS Circuits.* Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

[2] Bryant, R.E. "A switch level model and simulator for MOS digital systems," *IEEE Transactions on Computers* C-33 (2) 160-177, February 1984.

[3] Caisso J., Cerny E., Rumin N. "A recursive technique for computing delays in Series-Parallel MOS transistor circuits," *IEEE Transactions on Computer-Aided Design*, Vol 10, No. 5 June 1991.

[4] Chan, Pak., Schlag M. "Bounds on Signal Delay in RC Mesh Networks," *IEEE Transactions on Computer-Aided Design*, Vol 8, No. 6 June 1989.

[5] Ditlow, G.S., and A.E. Ruehli. "Logic equations and switch-level circuit simulation." in *Circuit Analysis, Simulation and Design; VLSI Circuit Analysis and Simulation.* ed. A.E. Ruehli. *Advances in CAD for VLSI Vol. 3 Part 2.* pp. 167-209. North-Holland. 1987.

[6] Elmore, W. C. "The Transient Response of Damped Linear Networks with Particular Regard to Wide-Band Amplifiers," *Journal of Applied Physics*, vol 19, no. 1, pp. 55-63, January 1948.

[7] Lin, T. M., Mead, C. "Signal Delay in General RC Networks," *IEEE Trans. on Computer-Aided Design* , vol CAD-3, no 4., pp. 331-349, October 1984.

[8] Mead, C. and Conway, L., *Introduction to VLSI systems.* Addison-Wesley Publishing Company, 1980.

[9] Musser, D., Narendran, P., Premerlani, W. "BIDS: A Method for Specifying and Verifying Bidirectional Hardware Devices," in *VLSI Specification, Verification and Synthesis*, Kluwer Academic Publishers, 1988.

[10] Rubinstein, J., Penfield, P., Horowitz, M. A. "Signal Delay in RC Tree Networks," *IEEE Trans. on Computer-Aided Design* , vol. CAD-2, no 3., pp. 202-211, July 1983.

[11] Terman, C. J. "Timing Simultion For Large Digital MOS Circuits," in *Advances in Computer Aided Engineering Design*, vol. 1, pp 1-92, JAI Press, London, 1985.

[12] Winskel, G., "A Compositional Model of MOS Circuits," in *VLSI Specification, Verification and Synthesis*, Kluwer Academic Publishers, 1988.

[13] Winskel, G., "Models and logic of MOS circuits," Logic of Programming and Calculi of Discrete Design, Springer-Verlag, 1987 1988.

[14] Wyatt, J. L. "Signal Propagation Delay in RC Models for Interconnect," *Advances in CAD for VLSI*, vol 3, part 2, North Holland, New York, 1987.

Delay Analysis of Combinations of Pass Transistors and Classical Logic Gates

Jonathan S. Gibson
Hewlett Packard Company
Integrated Circuit Business Division
Corvallis, Oregon

Abstract - **Combinations of pass transistors and logic gates driving nonlinear capacitive loads are analyzed for the presence of characteristics that will permit easier and more accurate digital logic simulation. Accurate time delay models are developed, by studying the nature of the response of simplified circuit models to variations of input waveform rise and fall times and output loading. The nonlinear effects of the CMOS logic devices are minimized to permit easier interpretation of the influence of nonlinear capacitive loads. The performance of a CMOS inverter with a complex nonlinear load consisting of a pass transistor that separates a range of capacitances, is compared to the same inverter circuit with a linear capacitive load, to develop an understanding of the unique requirements of modeling a nonlinear system. Several methods of modeling the delay of CMOS circuits are reviewed, and a multi-parameter linear model is described. General guidelines for designing CMOS circuits with complex load circuits are developed, emphasizing that the circuit output rise delays and fall delays must be separately analyzed.**

1 Introduction

Making a good initial choice of transistor sizes when designing high speed digital CMOS integrated circuits for a particular level of performance is sometimes difficult when pass transistors are mixed with classical logic gates. The delay of individual logic gates can be very well characterized and the manner in which logic gates perform when connected together can likewise be simulated accurately, based on predetermined delay values for each of the logic gates. The propagation delays of pass transistor circuits are reasonably well understood for series connected n-channel devices [1], within a logic structure such as a NAND gate. No references have been found on the subject of propagation delay through an inverter driven nonlinear circuit composed of a pass transistor separating two linear capacitances (Figure 1). The amount of capacitance on both sides of the pass transistor, the influence of body-effect on the pass transistor, the size of the driving logic gate and the size of the pass transistor interact to produce circuit delays that are, in some cases, much longer than might generally be expected. If the pass transistor source and drain are connected to the output of a logic gate and the input of another logic gate, respectively, the influence of the body effect on the effective resistance of the pass transistor can be very significant if the pass transistor source voltage is well above ground potential [2].

The delay characteristics of CMOS logic devices are very well known and can, by the use of multi-parameter expressions, be approximated to virtually any desired level of accuracy,

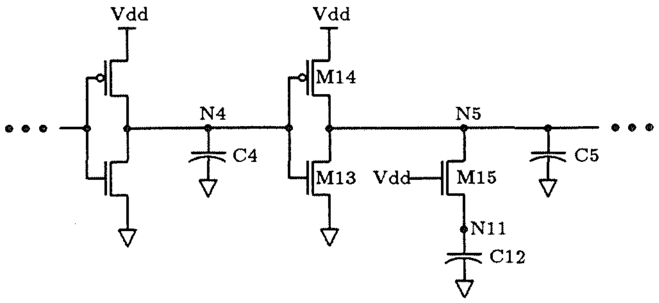


Figure 1: CMOS Inverter with a Non-linear Load

when compared to SPICE simulations. Delay calculation accuracy is generally balanced with circuit simulation time [4], depending upon the design methodology and particular circuit techniques being utilized. The characteristics of delays between logic gates have been analyzed in terms of RC trees, where the resistances and capacitances are derived from the parasitic resistance and capacitances of the integrated circuit (IC) layout between logic gates [3]. The IC layout parasitic extraction programs will accurately determine the resistance and capacitance between the logic gates in Figure 1, but the determination of circuit delay times for a logic simulation with timing analysis is not just a matter of lumping the RC delays of the parasitic circuit elements. The position of the pulldown transistor in Figure 1, between capacitive elements of various size and ratio of capacitance, causes nonlinear delay effects in the output transition of the device composed of transistors M13 and M14, greatly influencing the accuracy of derived circuit delay if a simple lumped RC model of the circuit is assumed. Device M15 may be conducting for a full logic transition, or switched to the conducting state sometime during a logic transition. As many different design applications could be addressed with the general circuit form of Figure 1, the focus in this work will be on the condition where the pulldown transistor is conducting for the entire test sequence, for both high to low and low to high logic transitions.

2 General Output Waveform Analysis

The analysis of the performance of the nonlinear output load circuit of Figure 1 is partitioned into discussions of the affect of logic gate transistor sizes, capacitance ratio of the load circuit (on both sides of the pulldown transistor), the size of the pulldown transistor, and a comparison with the baseline circuit configuration, which uses a simple, linear lumped capacitance as a load between the logic gates.

The inverter short circuit current remains less than about 5% of the total inverter current since the inverter input and output rise times remain close to the same values. The total node capacitance between the inverters is maintained at 10pF or close to that value.

2.1 Inverter Size

A waveform that describes the input of a logic gate must model the delay and the output waveform of the previous stage. For this analysis, as described earlier, a cascaded chain of inverters is used to develop a characteristic waveform that is used to represent the normally desired form of signal transitions between logic gates. Enough stages have been used in the inverter cascade, along with sufficient resolution in the circuit simulator so that the stage to stage difference in the logic transition rise and fall times and the stage to stage logic device delay is constant to well within one percent. Achieving better than one percent stage to stage delays insures that the circuit simulator is converging to a stable result after a logic change, a sometimes subtle but very important factor in achieving accurate delay analysis.

The expected influence of increasing device sizes in the inverters of an inverter cascade is easily demonstrated to be faster rise and fall times and less intrinsic logic device delay. With linear load capacitances, and with the transistor capacitance models disabled, so that the device parasitic capacitances are also modeled as linear capacitors, the stage to stage delays of the inverter cascade decrease linearly with increasing inverter device size. For this work, when the size of the n-channel device of an inverter is increased, the p-channel device is also changed to be about 2.5 times the channel width of the n-channel device, which keeps the rise and fall times of the inverter nearly equal. The exact ratio between the n-channel and p-channel devices is determined by achieving equal inverter rise and fall times. If the normal nonlinear parasitic capacitance models of the transistors are used, the nonlinear influence on delay times can range from about 3% to 10%, depending on the relative sizes of the inverter transistors and the load capacitance. The proportion of load capacitance due to the source-drain capacitances of transistors is generally greater for the larger transistor sizes, depending on the type of load driven by the inverter. For a constant load capacitance, the nonlinear nature of the source-drain device capacitances become a significant portion of the load capacitance, as the inverter transistor sizes increase. A specially modified version of the HPSPICE circuit simulator was created for this analysis where the capacitance models of the transistors could be disabled.

2.2 Capacitance Ratio

For the study of the influence of the capacitive ratio of the capacitances on either side of the pulldown transistor, the 10 picoFarad (pF) load capacitance was split into ratios of 1pF to 9pF, 2pF to 8pF, 3pF to 7pF, and so on, until 9pF to 1pF. The sum of the capacitances in each of the capacitance ratios always equals 10pF, to allow direct comparison to the simpler, single 10pF linear lumped capacitive load model. The first capacitance is between the pulldown device and ground and the second capacitance is between the inverter driving the nonlinear load and the next inverter in the inverter cascade. This form of divided capacitance simulates the effect of bus capacitances, where a pass transistor may be used to couple the output of a logic gate onto a datapath bus to be transferred to one or more other logic structures.

Since the number of inverter and pulldown transistor sizes and capacitance ratios produce a large number of unique combinations, the interpretation of the simulation results was simplified by normalizing the nonlinear load simulation data with the rise and fall delay times from the linear load inverter cascade. By dividing the nonlinear load delay times with the baseline linear load delay time for the same inverter size, the nonlinear load delay times become a percentage of the baseline linear load delay time. The range of delay times observed covers about two orders of magnitude and makes difficult the graphical observation of the differences in rise and fall delay times for each of the capacitance ratios, inverter sizes and pulldown transistor sizes in absolute time values. Figure 2 compares the time required to reach the switching voltage (approximately one half of the supply voltage) for the circuit node between the inverters (node N5 in Figure 1) with time required for the node between the pulldown pass transistor and ground to reach the same switch voltage. The nonlinear load of the circuit in Figure 1 alters the fall delay time of the test inverter by coupling some portion of the load capacitance below the pass transistor into the load that the test inverter must conduct to ground or from the supply rail, depending upon the direction of the logic transition.

2.3 Fall Delay Analysis

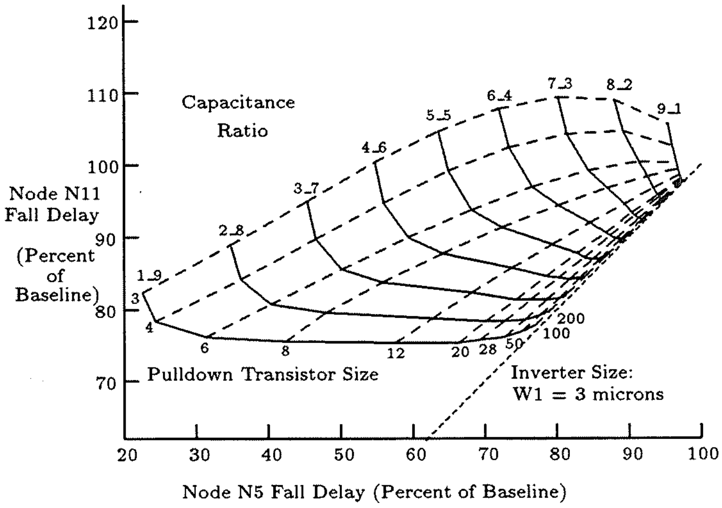


Figure 2: Fall Delay with an Inverter W_n of 3 microns

If the capacitance ratio is 1pF to 9pF, the 1pF capacitance is between the two inverters and the 9pF is between the pulldown transistor and ground. When the output of the test inverter is falling (transitioning from high to low), the pull-down transistor does not start to turn on until the inverter output voltage drops at least one n-channel voltage threshold below the circuit supply voltage (the gate of the pulldown transistor is tied to the supply voltage). Looking at Figure 2, it is apparent that the influence of the pulldown transistor in coupling the charge of capacitor C12 (Figure 1) into the effective capacitive load of the inverter does not become significant until after the threshold voltage of the n-channel pulldown device is reached. This delay in charge coupling is generally true, as long as the channel width of the pulldown transistor is nearly the same size or larger than the n-channel transistor in the test inverter. In the case when the inverter transistors are much larger than the pulldown transistor, the inverter may reach the switch measurement voltage before the pulldown transistor achieves significant charge transfer. Conversely, if the driving inverter is small and the pass transistor is large, the delay times will be close to the baseline delay times for all of the capacitive ratios, since the pass transistor capacitance (C12), is relatively well coupled to the output node of the driving inverter.

When the pulldown pass transistor is an n-channel device, the fall time switching delay at node N5 (Figure 1) is always less than the baseline fall time switching delay, for the same size of inverters, for all of the capacitance ratios. The reason for the lower delay time for the fall delay of node N5 is that some of the capacitance that is separated from the test inverter output node by the pass transistor is shielded from the switching inverter by the delay in the onset of conduction by the pass transistor. The finite resistance of the pass transistor provides a limit to the conduction of current from the simulated bus load, allowing the driving inverter to reach the switching voltage more quickly than the linear model.

Observation:

For the inverter driving a load circuit consisting of capacitances and a pass transistor as in Figure 1, there is an optimal load pass transistor size that minimizes the fall delay on both the source and drain nodes of the load pass transistor. This load pass transistor size is equal to or slightly larger than the n-channel pulldown transistor of the driving inverter.

2.4 Rise Delay Analysis

In contrast to the fall delays, the rise delays for the node N11 are always greater than the linear model baseline fall delay, for a n-channel pulldown pass transistor (Figure 3).

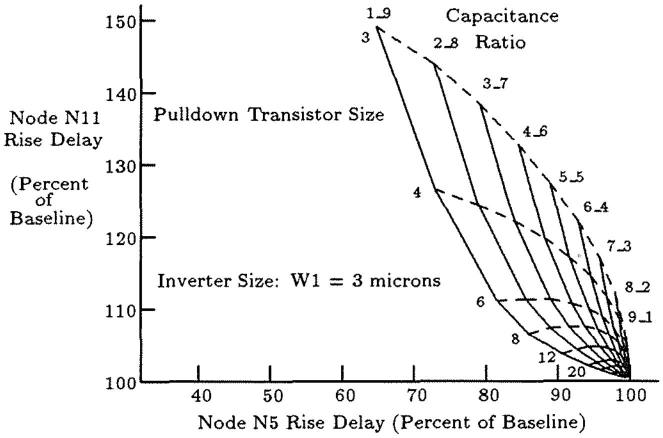


Figure 3: Rise Delay with an Inverter W_n of 3 microns

Node N11 has been discharged to very near ground potential by the previous high to low transition. Since the gate of the n-channel pulldown pass transistor is connected to the supply voltage, the pass transistor channel is fully conducting from the point that the transistor source is raised above ground potential by the test inverter output. Therefore, the capacitance at node N11 is immediately available for charging by the output of the test inverter. The time required by the test inverter to reach the switching voltage ($\frac{1}{2}$ V_{dd}) is decreased relative to the linear model baseline rise time delay by an amount proportional to the series resistance of the pulldown pass transistor and the total capacitance on node N11. The effective series resistance of the nonlinear load will always be greater than the equivalent series resistance of the baseline circuit, due to the presence of the pass transistor. The rise delay of node N5 is always less than the delay of the baseline circuit for the same inverter sizes. The capacitor C12 is relatively isolated from node N5 by the pass transistor, presenting less effective capacitive load to the node N5, which allows the test inverter output voltage to reach the switching voltage sooner.

Observation:

For the rise delay of the driving inverter, the load pass transistor fully conducts from the beginning of the rising output transition, allowing both parts of the load capacitance to influence the rise time of the driving inverter for the entire transition period. The rise delay of the driving inverter is made smaller by decreasing the size of the load pass transistor, which reduces the influence of the capacitance below the load transistor. The family of rise delay curves for each inverter size produces very similar delays for the pulldown pass transistor size that equals the size of the n-channel device of the driving inverter.

2.5 Pulldown Pass Transistor Device Size

As the pulldown pass transistor in the circuit of Figure 1 increases in size, the fall delay observed at node N5 increases and approaches the fall delay of the baseline lumped linear capacitance circuit, for all of the ratios of capacitance. As the pulldown pass transistor becomes larger, its effective impedance becomes lower, which facilitates the coupling of the capacitance on node N11 with that of node N5, with the result that the driving inverter takes longer to move node N5 to the switch voltage (approximately $\frac{1}{2}$ V_{dd}). By comparing the graphical representation of the fall delays for the circuit of Figure 1 as seen in Figures 2 and 4, it can be seen that the lines of constant capacitance ratio cross the same horizontal and vertical percentage points for the node N5 and node N11 fall delays.

The absolute value of the fall times vary considerably for each inverter size, but the relationship of the node N5 and node N11 fall delays with respect to the baseline delay for the same inverter size remains constant. As the inverter size increases, a correspondingly larger pulldown pass transistor is required to maintain the same relative delays on nodes N5 and N11. For large inverter sizes, the smallest pulldown pass transistor sizes are capable of increasing the node N11 delay time by several hundred percent. From the three fall delay figures, it is apparent that the pulldown pass transistor size should always be as large as the n-channel device of the driving inverter, to prevent possibly excessively long node N11 fall delay times.

The fall delay curves indicate that an optimum pulldown pass transistor size may be fifty percent larger than the inverter n-channel device, regardless of the node N5 to node N11 capacitance ratio. For the measured capacitance ratios, the size of the pulldown pass transistor can be selected to achieve the desired delay on node N5, while minimizing the delay on node N11. If minimum fall delay times are not important, the size of the pulldown pass transistor can be minimized to reduce the overall circuit power dissipation or circuit size, or some combination of circuit power and size.

Figures 3 and 5 show the rise delay relationship of nodes N5 and N11 (of the circuit of Figure 1) to the baseline lumped linear capacitance inverter load. As noted earlier, for an n-

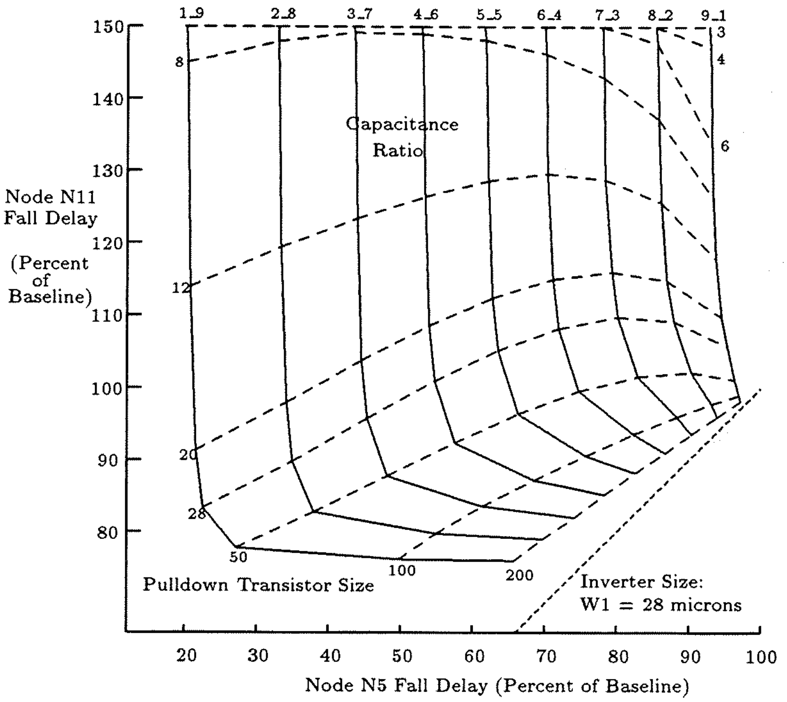


Figure 4: Fall Delay with an Inverter W_n of 28 microns

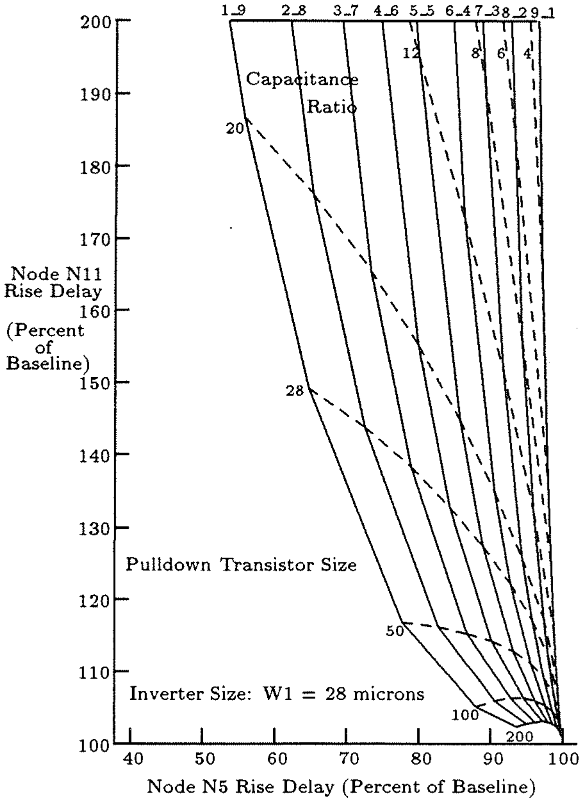


Figure 5: Rise Delay with an Inverter W_n of 28 microns

channel pulldown pass transistor, the node N11 delay is always larger than the corresponding rise delay for the baseline circuit with the same inverter size, due to the larger effective series resistance of the capacitive discharge path. The node N5 delay is less than the baseline rise delay, achieving a minimum delay for the small pulldown pass transistor sizes and delays nearly equal to the baseline circuit for the largest pulldown pass transistor sizes.

3 Effective Capacitance

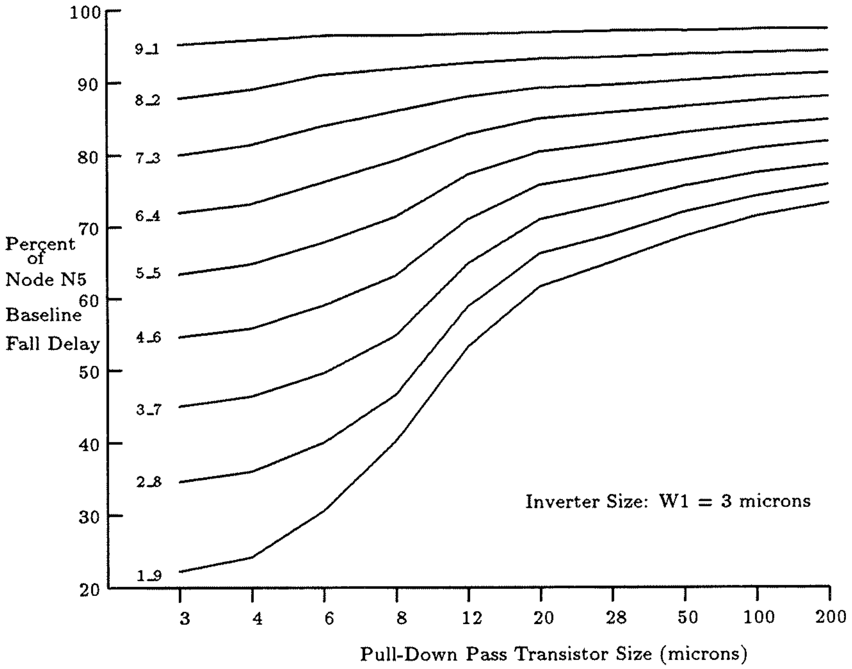


Figure 6: Percent Fall Delay, $W_n = 3$ microns

For a particular ratio of load capacitance of the form described in the circuit of Figure 1, define **effective capacitance** as the value of a single linear capacitor applied to the same inverter which produces the same rise and fall delay time as the baseline lumped linear capacitive load. The purpose of defining effective capacitance is to provide the designer with a simplified design criteria for determining how quickly a node may reach the logic switching voltage when a nonlinear capacitive load is created by the presence, in this case, of a pass transistor.

3.1 Fall Delay Effective Capacitance

The nonlinear capacitive load can introduce significant distortion in an inverter output waveform when compared to the inverter characteristic waveform, but it is only the portion of the waveform from the beginning of a logic transition to the logic switching voltage that is important for the purpose of calculating propagation delay. The use of the appropriate effective capacitance can assist in the selection of the proper transistor sizes for the logic gates,

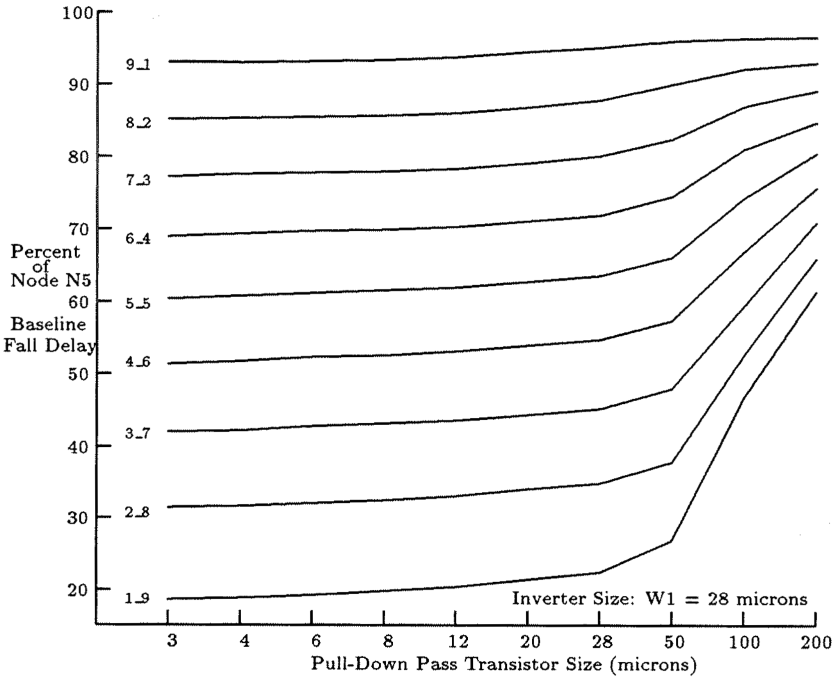


Figure 7: Percent Fall Delay, W_n = 28 microns

potentially reducing the time required to produce a satisfactory design. The possibility of inadvertently designing a circuit that has an unexpectedly long rise or fall delay time should be reduced by having a general knowledge of the effective capacitance of a complex circuit configuration. Figures 6 and 7 show the fall delays of node N5 of the circuit of Figure 1 as a percentage of the fall delay of the baseline lumped linear capacitance circuit, for small to intermediate inverter sizes (W_n of 3 and 28 microns). The percent of baseline rise or fall delay is directly proportional to the effective capacitance. The effective capacitance is computed by multiplying the percent of baseline delay times the total capacitance on node N5 for the baseline linear lumped capacitance circuit. For the circuits with intermediate inverter sizes, the smallest pulldown pass transistors provide the greatest reduction of effective capacitance on node N5, since the small pass transistors do not conduct a significant amount of capacitive charge by the time the switch voltage is reached. If the pulldown pass transistor is about 10 times the inverter W_n size, the effective capacitance is about 90% of the baseline fall delay.

3.2 Rise Delay Effective Capacitance

Figures 8 and 9 show the rise delays of the same circuit (Figure 1) as a percentage of the rise delay of the baseline lumped linear capacitance circuit. The influence of the pulldown pass transistor on reducing the effective capacitance of node N5 can be seen to be greater for the fall delays, when compared to the rise delays. For the rise delays of the small inverter sizes(W_n of 3 and 28 microns), the smallest pulldown pass transistor sizes produce the smallest effective capacitance at node N5. For large inverters, the pulldown pass transistor influence on the effective capacitance remains significant even for large pulldown pass

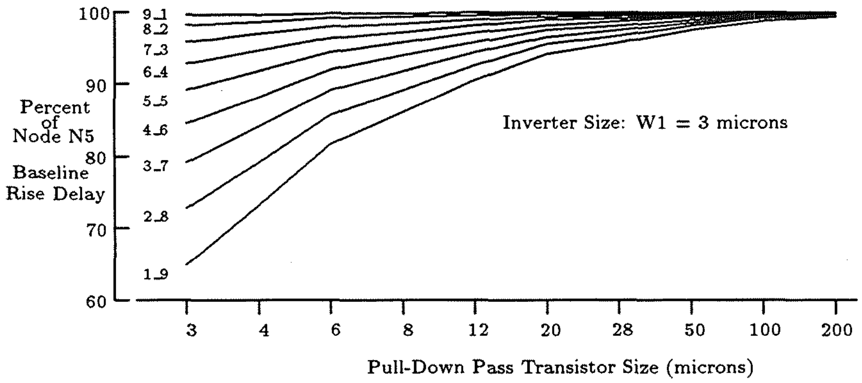


Figure 8: Percent Rise Delay, W_n = 3 microns

transistors, due primarily to the very fast switching time of the large inverters. When a n-channel pulldown pass transistor is utilized, the potential exists for very long rise delays, depending upon where the switch voltage is set for any logic gates connected to node N11. For small pulldown pass transistors and relatively large inverters, the rise time of node N11

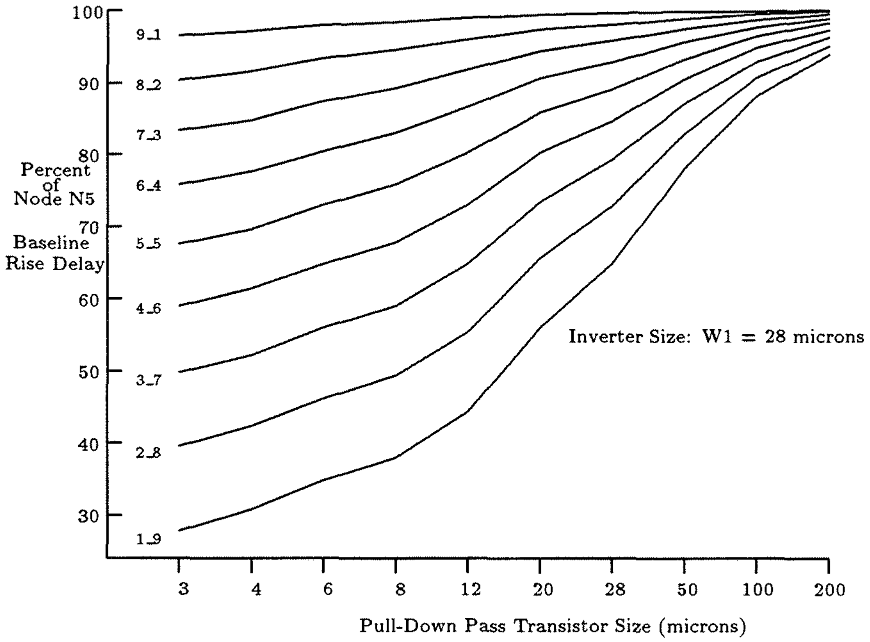


Figure 9: Percent Rise Delay, W_n = 28 microns

can become very slow, when compared to the rise time of node N5, the output of the inverter.

If the definition of the logic transition level for node N11 is changed from $\frac{1}{2}$ V_{dd} to $\frac{1}{2}$ the final level of node N11, the value of the rise delay is correspondingly reduced. The general form of the rise delay relationships between node N5 and node N11 are only slightly altered.

Observation:

The linear portion of the fall delay curves (left edge) very closely matches the circuit performance of an inverter with a linear capacitive load. The area on the right hand side of the fall delay curves where the slope changes rapidly shows the

point where the pulldown pass transistor is fully turned on, allowing the charge on the capacitor (C12) below the pulldown pass transistor to add to the charge of capacitor C5 (Figure 1), which increases the fall delay time. The output voltage waveform of the inverter actually changes slope at this point. When the combination of capacitive ratio, inverter size and pulldown pass transistor size combine to move the point of where the charge of capacitor C12 influences the output of the driving inverter above $\frac{1}{2}$ V_{dd}, the extra delay is seen in the fall delay graphs.

4 Summary

The rise and fall delay performance of an inverter, as a function of inverter size, capacitance ration and pulldown transistor size, was investigated. When an n-channel pass transistor is used as a part of the nonlinear load, careful sizing of the components of the circuit of Figure 1 will prevent extremely large rise delays.

The rise and fall times on the input and output of an inverter should be approximately equal to minimize the inverter short circuit current. Nonlinear loads of the type studied here do influence the peak short circuit current and when it occurs but do not greatly affect the average short circuit current.

When the nodes N5 and N11 of the circuit of Figure 1 are compared to the baseline linear lumped capacitance circuit model, distinctive patterns of delay are observed for the rise and fall delays. Optimization of the design of a CMOS circuit could be achieved by relating the size of the capacitive load to the driving inverter size chosen to meet performance specifications, and then choosing an optimal pass transistor size that is just large enough to produce the desired time delays. An advantage of the comparative graphical display of delay values is that the designer can easily see when a device size is chosen that is larger than necessary or is small enough to generate a much longer delay than is desired, depending upon the capacitive load.

References

[1] T. Sakurai, A.R. Newton, "Delay Analysis of Series-Connected MOSFET Circuits", IEEE Journal of Solid-State Circuits, Vol. 26, No. 2, February 1991, pp. 122-131.

[2] R.L. Geiger, P.E. Allen, N.R. Strader, *VLSI Design Techniques for Analog and Digital Circuits*, McGraw-Hill (New York), 1990.

[3] C. Lee, H. Soukup, "An Algorithm for CMOS Timing and Area Optimization", Journal of Solid State Circuits, 1984, Vol. 19, No. 5, pp. 781-787.

[4] A. Martinez, "Automated Library Characterization and Timing Model Accuracy Issues When Interfacing to Different CAD Tools ", Hewlett Packard Design Technology Conference Proceedings, pp. 511-520, Denver, Colorado, May 1991.

Mixed Voltage VLSI Design [1]

Ramesh Panwar§†, David Rennels§†, and Leon Alkalaj§

§Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

†Department of Computer Science
University of California
Los Angeles, CA 90024

panwar, rennels, leon@telerobotics.jpl.nasa.gov

Abstract – **We present a technique for minimizing the power dissipated in a VLSI chip by lowering the operating voltage without any significant penalty in the chip throughput even though low voltage operation results in slower circuits. Since the overall throughput of a VLSI chip depends on the speed of the critical path(s) in the chip, it may be possible to sustain the throughput rates attained at higher voltages by operating the circuits in the critical path(s) with a high voltage while operating the other circuits with a lower voltage to minimize the power dissipation. The interface between the gates which operate at different voltages is crucial for low power dissipation since the interface may possibly have high static current dissipation thus negating the gains of the low voltage operation. We present the design of a voltage level translator which does the interface between the low voltage and high voltage circuits without any significant static dissipation. We then present the results of the mixed voltage design using a greedy algorithm on three chips for various operating voltages.**

1 Introduction

The power consumed in CMOS VLSI primarily results from charging and discharging the node capacitances in the VLSI circuit. The CV^2f power dissipated in a VLSI chip depends on the square of the operating voltage [3]. Thus operating a chip with 3V instead of 5V results in 64% power savings. Low voltage operation is one way of achieving low power in systems. Lower voltage operation also results in improved signal integrity and reliability due to lower crosstalk and ground bounce. Space systems that require low power with high reliability should find the migration to lower voltages a very attractive proposition. Lowering the operating voltage has an adverse effect on the speed of the circuits. Thus low voltage design reduces the overall chip throughput unless some other compensating measures are taken. In this paper, we propose a mixed voltage design technique which addresses the problem of lowering the power dissipation without affecting the chip throughput. The circuits in the critical path(s) are operated with the higher voltage to maintain the chip throughput while the circuits that are not in the critical paths are operated with the lower voltage to minimize the power dissipation. The interface between circuits that operate at different voltages can have large static current dissipation because the transistors in both the p-block

[1] Partial support for this work was provided by the Office of Naval Research under grant N00014-91-J-1009 at UCLA

and the n-block may be on resulting in a conducting path from the power to the ground rail. We present a voltage level translator that can be used in the interface. The translator is implemented using 6 transistors with appropriate transistor sizing. The translator has an excellent DC transfer characteristic and fast propagation delays with very minimal static current dissipation (of the order of pico-amperes). We then present a greedy algorithm for doing the mixed voltage design. The algorithm is not optimal but gives very satisfactory results. Finally we present the results of the mixed voltage design algorithm on three chips that were designed at JPL for various flight projects using the Honeywell RICMOS process [4].

2 Mixed Voltage Design

2.1 Low Voltage Operation

The delay ($t_d = 4C_L/\beta V$) of a VLSI circuit depends on the inverse of the operating voltage [3] while the power ($P = CV^2f$) consumed by the circuit depends on the square of the voltage. It follows from this observation that if we decrease the operating voltage of a chip, then we must increase the clock cycle time to allow for the slower operation of the circuits on the chip. Increasing the chip clock cycle time implies a reduced overall chip throughput. A way to overcome the problem of reduced chip throughput for low voltage design is by architectural techniques like pipelining and parallelism as discussed in [2]. Two problems with the architectural solutions outlined in [2] is that they involve architectural redesign and they require more chip real-estate. Essentially, the techniques proposed in [2] trade off power for area. We propose a technique which does not require any architectural redesign. The technique relies on the fact that the chip clock cycle time (and throughput) depend on the speed of the critical path(s). The elements in the critical path(s) may be operated with a high voltage to ensure that there is no (or minimal) loss of chip throughput. The elements that are not in the critical paths may be operated with the lower voltage to ensure lower power dissipation. Let us consider two operating voltages V_{ddh} and V_{ddl} for operating a VLSI circuit where $V_{ddh} > V_{ddl}$. We begin with some definitions to illustrate the tradeoffs in mixed voltage design.

Definition 1 *The switching energy ratio r_{SE} of a circuit with respect to two voltages V_{ddh} and V_{ddl} is the ratio of the energy required for switching the circuit with the lower voltage to the energy required for switching the circuit with the higher voltage.*

Definition 2 *The switching delay ratio r_{SD} of a circuit with respect to two voltages V_{ddh} and V_{ddl} is the ratio of the switching delay of the circuit with the lower voltage to the switching delay of the circuit with the higher voltage.*

Definition 3 *The chip energy ratio r_{CE} of a chip with respect to two voltages V_{ddh} and V_{ddl} is the ratio of the switching energy of the chip with a mix pf V_{ddh} and V_{ddl} operation to the switching energy of the chip with V_{ddh} operation alone.*

Definition 4 *The chip delay ratio r_{CD} of a chip with respect to two voltages V_{ddh} and V_{ddl} is the ratio of the fastest chip clock cycle time with a mix of V_{ddh} and V_{ddl} operation to the fastest chip clock cycle time with V_{ddh} operation alone.*

Definition 5 *The MVD ratio r_{MVD} of a chip is the ratio of swtching energy ratio and the chip energy ratio.*

Given the expression for the circuit delay and power dissipation, we have the following relationships:

$$r_{SE} = \frac{V_{ddl}^2}{V_{ddh}^2}$$

$$r_{SD} = \frac{V_{ddh}}{V_{ddl}}$$

The chip delay ratio r_{CD} is a parameter that should be determined by the system architect. The parameter will depend on how much loss of throughput can be tolerated by the chip when we go from a high voltage operation to a mixed voltage operation. For example, if a compression chip is designed such that its maximum path delay is 87 ns but the chip needs to handle input data rates of 10 MHz, then the chip can withstand a chip delay ratio of approximately 1.15. Once an acceptable value for the chip delay ratio has been determined by the system architect, the technique of mixed voltage design may used to lower the chip power dissipation. The reduced power of the chip will result in a chip energy ratio which is a function of the switching delay ratio and chip delay ratio:

$$r_{CE} = f(r_{SD}, r_{CD})$$

It should be noted that the switching energy ratio is dependent on the switching delay ratio:

$$r_{SE} = 1/r_{SD}^2$$

2.2 Voltage Level Translator

In a chip with mixed voltage design, a gate that operates with the higher voltage V_{ddh} may receive inputs from gates that operate with the lower voltage V_{ddl}. In such a case, the gate input swings between V_{ss} and V_{ddl}. The input signal may produce a conducting path between V_{ddh} and V_{ss} in the gate when it is at V_{ddl} since both the p-transistors and the n-transistors in the path may be on continuously. This causes a large static current dissipation in the gate which may negate the gains of low voltage operation. Hence, we need a voltage level translator to translate the low voltage signals produced by gates that operate with V_{ddl} to the high voltage signals that can be applied as inputs to the gate operated with V_{ddh}. It should be noted that the reverse voltage level translator to translate high voltage signals to low voltage signals is not required since high voltage signals driving a gate operated at a lower voltage can never produce a conducting path between the power and ground rails (other than the transient conducting path that produces the short-circuit current found in conventional CMOS). We have designed a voltage level translator using 6 transistors which allows us to convert low voltage signals (signal swing between V_{ss} and V_{ddl}) to high voltage signals (signal swing between V_{ss} and V_{ddh}). The desireable characteristics of such a translator are the following:

- The translator should have good DC characteristics for noise tolerance.
- The translator should have reasonably fast propagation delays.
- The translator should have no (or minimal) static dissipation.
- The translator should not be too expensive in terms of VLSI real-estate.
- The translator should not require any special processing for VLSI fabrication.

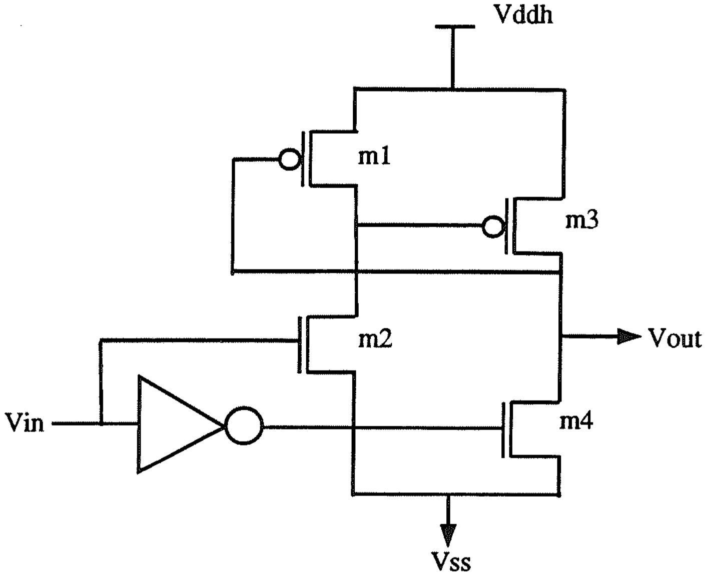


Figure 1: Voltage level translator

The voltage level translator is shown in figure 1. The operation of the translator is as follows:

- Asssume that the input to the translator V_{in} has been steady at the logic level 0 which implies that $V_{in} = V_{ss}$. In this state, transistor *m2* is turned off while transistor *m4* is turned on. Since *m4* is on, the output V_{out} is pulled low. Since V_{out} is fed back to the input of *m1*, transistor *m1* is on which implies that the gate of transistor *m3* is pulled to V_{ddh}. Thus, transistor *m3* is off which allows the output V_{out} to remain at V_{ss}. It should be noted that there is no conducting path from V_{ddh} to V_{ss} in this state.

- When the input to the translator switches from V_{ss} to V_{ddl}, then transistor *m2* is turned on while transistor *m4* is turned off. Since transistor *m1* is on at the same time, there is a conducting path from V_{ddh} to V_{ss} which leads to a flow of short-circuit current. However, if transistor *m2* is larger than transistor *m1*, then it will overpower transistor *m1* and discharge the gate input of transistor *m3* thus forcing *m3* to turn on. When *m3* turns on, the output V_{out} is pulled to V_{ddh} which turns off transistor *m1*, thus allowing the translator to settle into a stable state with $V_{out} = V_{ddh}$. There is no conducting path from V_{ddh} to V_{ss} in the stable state thus eliminating the possibility of any static current dissipation.

- When the translator input changes back to V_{ss}, transistor *m2* is turned off while transistor *m4* is turned on. Since transistor *m3* is on at the same time, there is a conducting path from V_{ddh} to V_{ss} which leads to a flow of short-circuit current. However, if transistor *m4* is larger than transistor *m3*, then it will overpower transistor *m3* and discharge the gate input of transistor *m1* thus forcing transistor *m1* to turn on. *Vout* consequently goes low. When *m1* turns on, the gate input of transistor *m3* is pulled to V_{ddh} which turns off transistor *m3* thus allowing the translator to settle in the stable state with $V_{out} = V_{ss}$. There is no conducting path from V_{ddh} to V_{ss} in the stable state thus eliminating the possibility of any static current dissipation.

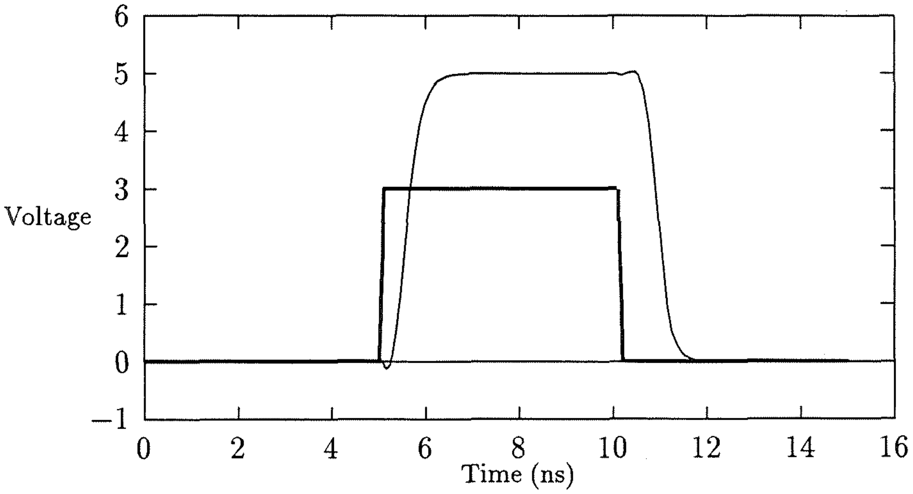


Figure 2: Pulse response

Figure 2 shows the response of the voltage level translator to a 5 ns pulse. The simulations were done using SPICE with models for 2μ devices from [1]. A critical parameter in the design of the voltage level translator is the sizing of transistors *m2* and *m4* relative to *m1* and *m3*. For the simulations, we sized the transistors such that the W/L ratios of *m2* and *m4* were 5 times those of *m1* and *m3*. This ratio is sufficient for attaining a reasonably fast propagation delay as shown in figure 2. The rise time delay in the simulation was 1.08 ns and the fall time delay was 1.18 ns. Figure 3 shows the DC transfer characteristic of the voltage level translator. The figure shows that the translator output has a very sharp transition when the input voltage is approximately 1.4V (assuming that the translator voltage levels are 3V and 5V). The sharpnesss of the response implies a very good noise immunity. The static current dissipation in the translator was 13.9 pA.

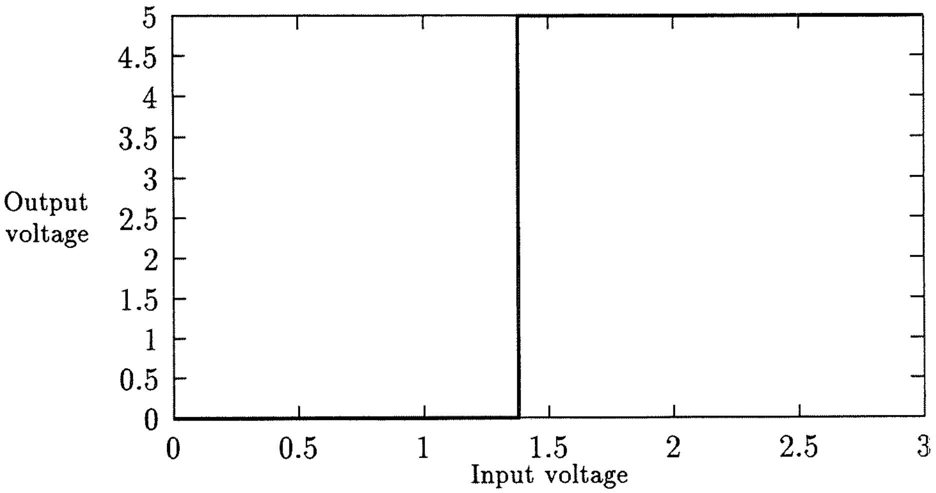


Figure 3: DC transfer characteristic

2.3 Optimization Problem

For the purpose of illustrating the optimization problem in mixed voltage design, we shall assume a standard cell based design. Each cell may have a number of inputs and outputs and may occur on a number of different paths in the chip. We denote the delay presented by cell i to path j by d_{ij}. Assume that each cell i has a weight w_i which is proportional to the internal capacitance of the cell. The weight of a cell is used to select between two cells for the optimization algorithm since a cell with a higher weight will lead to greater power savings if the cell is operated with the lower voltage. A path is essentially an ordered set of cells which occur on the path. Let us denote the set of cells representing the path j by S_j. If we use D_c to denote the chip critical path delay when the entire chip is operated at V_{ddh}, then the optimization problem can be cast as follows:

$$\begin{array}{ll} \text{Maximize} & \sum_{i=1}^{N} \delta_i w_i \\ \text{subject to} & \sum_{i \epsilon S_j} (1 - \delta_i + \delta_i r_{SD}) d_{ij} \leq r_{CD} D_c \quad \forall j \end{array}$$

where

$$\delta_i = \begin{cases} 1 & \text{if cell } i \text{ is operated with lower voltage} \\ 0 & \text{otherwise} \end{cases}$$

3 Experimental Results

3.1 Chip Benchmarks

We considered three chip designs that were done at JPL for various flight projects as benchmarks for our study of power optimization using mixed voltage design. The chips were designed using the Honeywell RICMOS process [4] by three different VLSI designers and consequently had very different path length distributions. The chips were approximately of the same size and various statistics on the chips are shown in table 1. The table shows the total number of paths, cells and registers in each of the chips. The cells used in the chip designs ranged in complexity from a simple inverter to a 4-to-1 multiplexer. The designs had a total of 39 different types of cells. The weight w_i of a cell was assumed to be the number of gate equivalents used by the cell [4]. Figures 4-6 show the distribution of path lengths in the three chip designs. The length of a path is the total number of cells in a path. The cumulative frequency (on the y-axis) is the total fraction of paths whose length exceed that of the path length shown on the x-axis. The figures and table 1 show that one of the chips (*chip*2) was significantly different from the other two chips. The largest path length in *chip*2 was 34 while the largest path length for the other designs was 18. Also, *chip*2 had a very large number of paths compared to the other chips inpite of the fact that all the chips had about the same number of cells and registers.

	chip0	chip1	chip2
n_{paths}	2699	2846	4123
n_{cells}	1936	2121	2038
n_{regs}	491	552	519

Table 1: Chip statistics

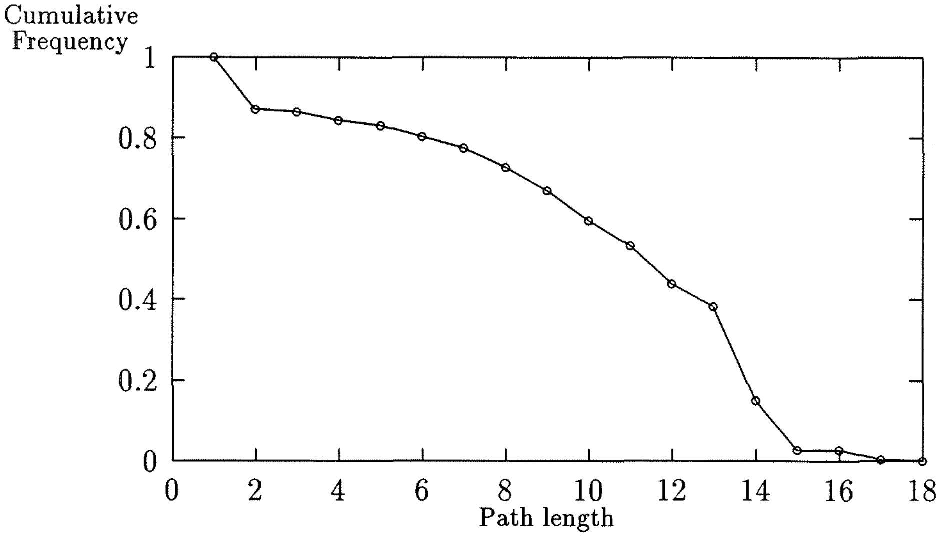


Figure 4: Path length distributions in *chip*0

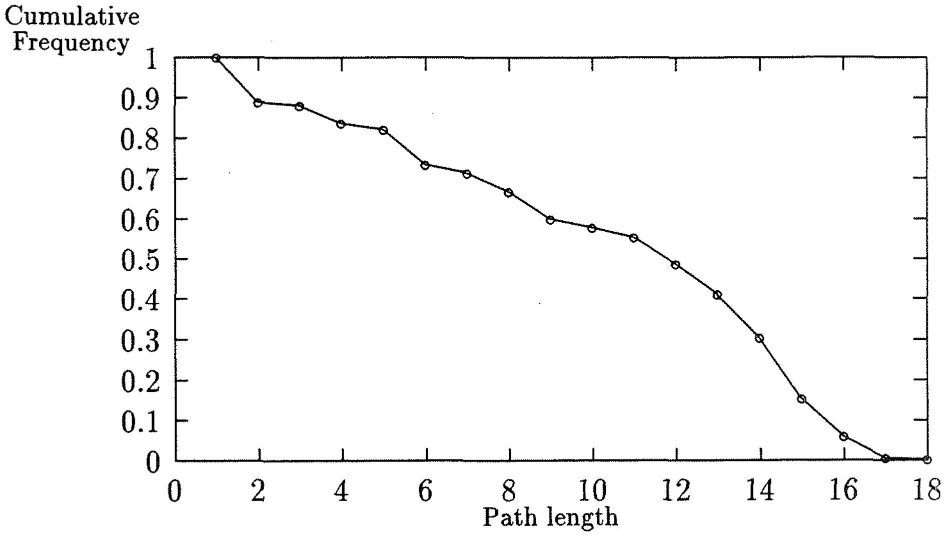


Figure 5: Path length distributions in *chip*1

3.2 Greedy Optimization Strategy

We adopt a greedy optimization algorithm for mixed voltage design. The input to the algorithm is an adjacency list representation of the chip design. Each list on the adjacency list represents a path. We sort the cells by their weights and then traverse down the sorted list of cells to see which cells can have their voltage lowered to V_{ddl}. The voltage of a cell can be lowered to V_{ddl} if the delays of all the paths that contain the cell are still bounded by $r_{CD}D_c$ when we lower the voltage of the cell to V_{ddl}. It should be apparent to the reader that a greedy strategy on the cell weights is not optimal. To prove the non-optimality of the greedy strategy, consider three cells i, j, and k on path S_l. If the weights of the cells are such that $w_i > w_j > w_k$, then during the execution of the greedy algorithm, cell i will have its voltage lowered to V_{ddl}. However, doing so may preclude cells j and k from having their voltages lowered because we may violate the timing constraint if the delay of S_j exceeds $r_{CD}D_c$. On the other hand, if we had lowered the voltages of j and k first, we may have still stayed within the bounds imposed by the timing constraint and yet have attained a lower power dissipation. It is easy to see that this will be the case if the following conditions hold:

$$d_{jl} + d_{kl} < d_{il}$$

$$w_j + w_k > w_i$$

One may consider other types of greedy strategies (involving greed on the ratio of the cell weight and the cell delay) or other approaches using dynamic programming. The greedy algorithm (which is greedy on the cell weigths) gives us very good results for power savings and we did not investigate the other greedy approaches or dynamic programming approaches for this paper. We plan to look into the other approaches for future research.

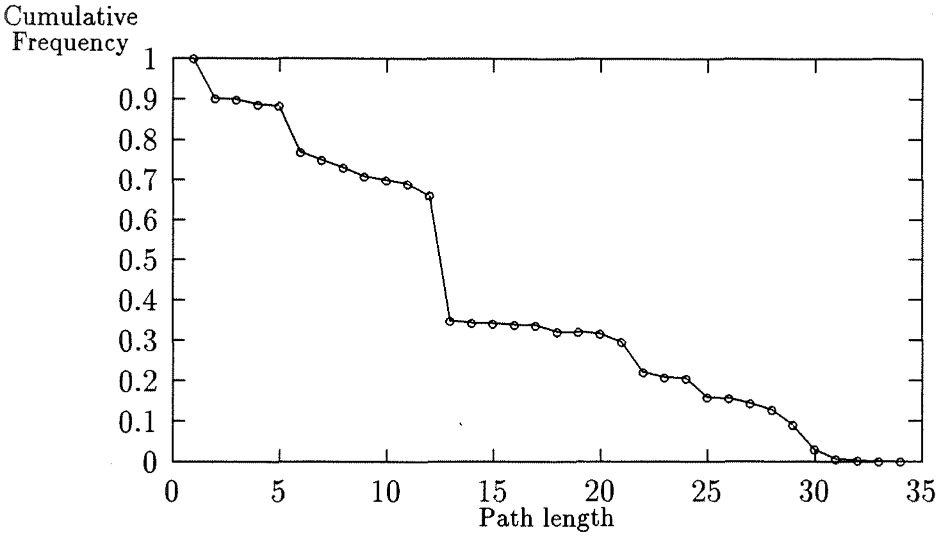


Figure 6: Path length distributions in *chip*2

3.3 Power Savings

Figures 7-9 show the results of the greedy strategy on the three chip benchmarks. The figures show the MVD ratio (r_{MVD}) as a function of the chip delay ratio (r_{CD}) for various values of the switching delay ratio r_{SD}. The MVD ratio should be a monotonically increasing function of the chip delay ratio. The non-monotonicity of the MVD ratio as shown in the figures is an artifact of the non-optimality of the greedy algorithm. The power savings is given by $(1 - 1/r_{SD}^2 r_{MVD}) \times 100\%$. It can be seen that all the chip designs have a power savings of 65-70% without any significant penalty in the chip throughput ($r_{CD} = 1$) and that all the designs have a power savings of about 83% if the chips are allowed to be slowed down by a factor of 2. The MVD ratio is almost a linear function of the chip delay ratio which implies that the power dissipated is inversely proportional to the chip delay ratio. A higher value of r_{SD} (as a consequence of lower V_{ddl}) results in lower power dissipation up to a certain point after which the power dissipated is constant. For example, for *chip0*, the power saved is approximately 65% (for chip delay ratios of $r_{SD} = 2.0$ and $r_{SD} = 2.5$) which implies that any futher increase of r_{SD} by reducing V_{ddl} will not result in any more power savings. In fact since the power savings is identical for both values of r_{SD}, the designer may prefer operating the chip with $r_{SD} = 2.0$ since the MVD ratio r_{MVD} is higher for $r_{SD} = 2.0$ as compared to the MVD ratio for $r_{SD} = 2.5$. The higher MVD ratio implies less translators in the design though we have not quantified this in our study yet. Future work will address this relationship along with analytical models of the MVD ratio and the power savings that can result.

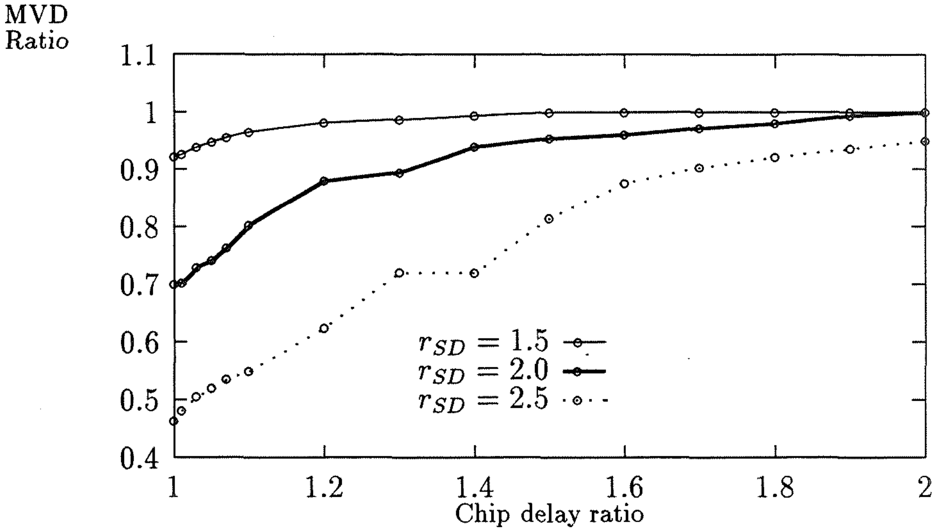


Figure 7: Mixed voltage design results for *chip*0

4 Conclusions

We have presented a technique for minimizing the power dissipation in CMOS VLSI by operating the VLSI chip with two voltages. The higher voltage is used to ensure that the chip throughput isn't adversely affected by the low voltage operation while the lower voltage is used for the majority of the cells in the chip to ensure low power dissipation. The key to designing chips with two operating voltages is an innovative voltage level translator that we designed using 6 transistors that allows us to translate low voltage signals to high voltage signals without any significant static power dissipation. Results of using the mixed voltage design technique with a greedy optimization strategy were presented for three VLSI chips with considerably different designs showing that it is possible to save about 65-70% of the power dissipated in the chips without paying any penalty in the chip throughput.

5 Acknowledgements

We would like to thank Gary Bolotin who provided the chip designs used as benchmarks in this study.

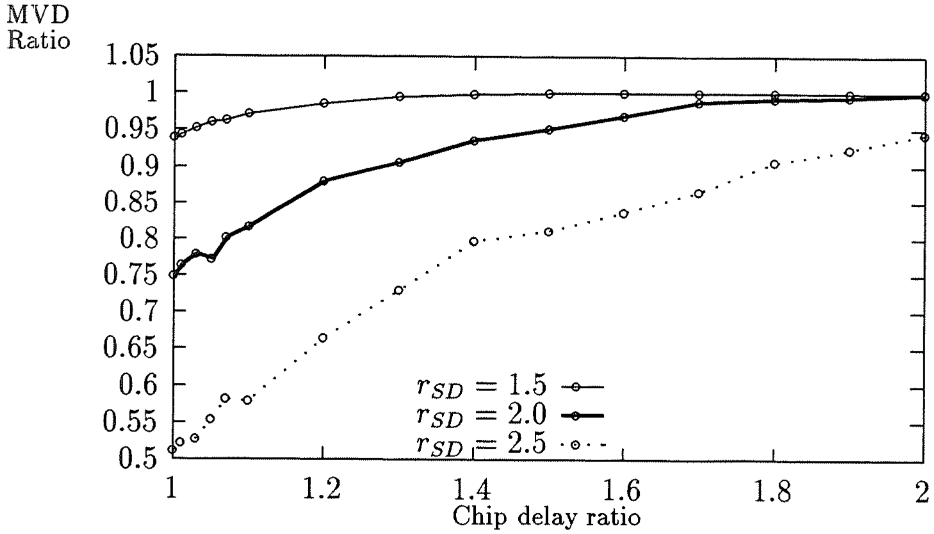


Figure 8: Mixed voltage design results for *chip*1

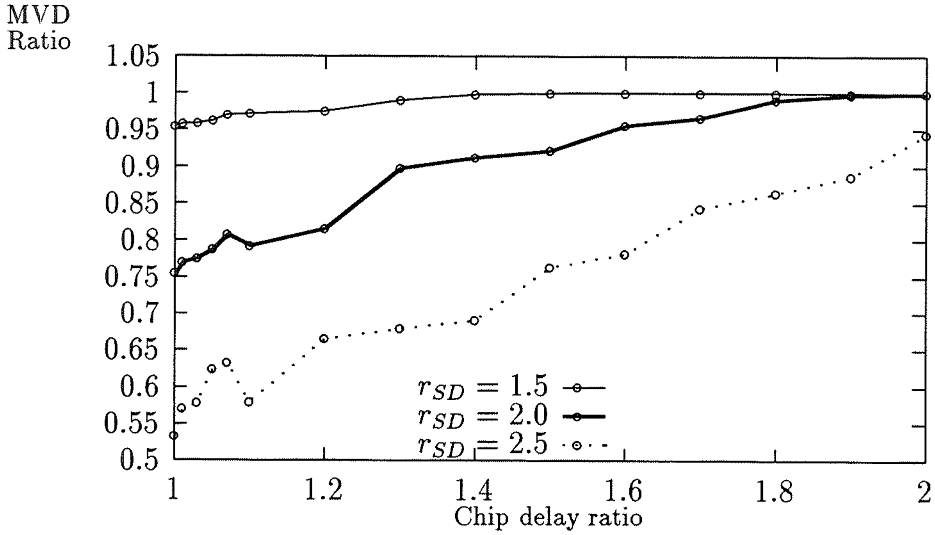


Figure 9: Mixed voltage design results for *chip*2

References

[1] H. Bakoglu, *Circuits, Interconnections and Packaging for VLSI.* Reading, MA: Addison-Wesley, 1990.

[2] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 4, Apr. 1992.

[3] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design.* Reading, MA: Addison-Wesley, 1988.

[4] *The Honeywell RICMOS ASIC Library Data Book.* 1991.

Session 7
Fault Testing

Chairman: Tom Caudell

1994016632
587430
N94-21105 P.10

Bridging Faults in BiCMOS Circuits[1]

Sankaran M. Menon, Yashwant K. Malaiya[†] and Anura P. Jayasumana
Dept. of Electrical Engineering, [†]Dept. of Computer Science
Colorado State University
Ft. Collins, CO 80523

Abstract - **Combining the advantages of CMOS and bipolar, BiCMOS is emerging as a major technology for many high performance digital and mixed signal applications. Recent investigations have revealed that bridging faults can be a major failure mode in ICs. This paper presents effects of bridging faults in BiCMOS circuits. Bridging faults between logical units without feedback and logical units with feedback are considered. Several bridging faults can be detected by monitoring the power supply current (I_{DDQ} monitoring). Effects of bridging faults and bridging resistance on output logic levels have been examined along with their effects on noise immunity.**

1 Introduction

Combining the advantages of CMOS and Bipolar, BiCMOS is emerging as a major technology for many high performance digital and mixed signal applications. The main advantages of CMOS technology over bipolar are lower power dissipation and higher packing density. Bipolar technology offers better output current drive, switching speed, I/O speed and analog capability. Combining the advantages of bipolar and CMOS, BiCMOS offers the following advantages [1]; improved speed over CMOS, lower power dissipation compared to bipolar, flexibility in I/O (TTL, ECL, CMOS compatibility), high performance analog capability and latch up immunity. Compared to the CMOS counterparts, BiCMOS circuits can be faster by a factor of up to two for the same level of technology. Access times of less than 10ns have been reported for 0.8 μm BiCMOS ECL input/output 256K and 1M-bit SRAMs [2]. BiCMOS is even being considered for high performance microprocessors and dynamic RAMs, and it is felt that it will be one of the main technologies to drive almost all functions in the decade ahead [3].

Rapid advances, increasing complexities and shrinking device geometries in VLSI have enabled complex integrated circuits to be manufactured for extremely complex systems at lower costs. Various modes of failures can occur in such complex VLSI devices, where bridging faults have been shown to be about half of the faults in CMOS circuits [4]. Another study [5], based on layout level defects using statistical data from fabrication process concludes that bridging faults can be up to 30% of all faults. Hence, bridging is an important failure mode which needs careful and systematic analysis. Detailed examinations of bridging faults in nMOS/CMOS have been presented in [6, 7] and on ECL in [8]. Effects of stuck-short and open of transistors in ECL OR/NOR gate and 2-level complex ECL gates were presented in [9, 10]. Bridging faults can occur within an integrated circuit or a printed circuit board

[1] This research was supported by a SDIO/IST funded project monitored by ONR.

during manufacturing or at a later stage. Most of the literature on bridging faults assumes a simple gate level model.

Major causes of bridging faults are related to the manufacturing defects. In the photolithography stages of the manufacturing process, diffraction and proximity are the prime source of excess metal causing bridging faults. Impurities and diffusion of metal are other sources responsible for such faults.

The most common type of BiCMOS circuits employ bipolar transistors to perform the function of driving output loads and CMOS to perform logic functions. Analysis of bridging faults in BiCMOS circuits is presented in [11]. In this paper, we present the behavior of BiCMOS devices under the following classes of bridging [6], namely; bridging of logical nodes without feedback and bridging of logical nodes with feedback. Output behavior of a BiCMOS device under output bridging fault with varying resistance is shown and values of bridging resistance affecting the output logic levels is presented along with their effects on noise immunity. Reference [11] does not present bridging of BiCMOS gates in the above different fault classes.

This paper is organized as follows. The operation of a basic BiCMOS NAND is explained in Section 2. Sections 3 and 4 cover bridging of logical nodes without feedback and with feedback respectively. Finally, conclusions drawn from the study are given in section 5.

2 D-BJT BiCMOS device

A Double BJT BiCMOS NAND realization is shown in Figure 1. There are many other realizations possible and we have investigated the above realization. The functioning of the BiCMOS NAND can be explained by first applying logic '0' to one or both of the inputs which would cause at least one P-device to be ON and at least one N-device in each serial N-pairs to be OFF. With at least one N-device in each serial N-pairs being OFF, no current is supplied to the base of Q_2 resulting in transistor Q_2 being OFF. With the P-devices (P_1 and/or P_2) ON, the base of the bipolar NPN transistor would be about 5V supplying base current and turning ON the bipolar transistor providing logic '1' at the output. With either of the inputs being at logic '0' and the other input at logic '1' would still cause either of the parallel connected P-devices to be ON and either of the series connected N-devices to be OFF. This would still supply base current to the bipolar transistor Q_1 causing logic '1' at the output. With both the inputs at logic '1', the P-devices (P_1 and P_2) would be turned OFF, and the N-devices N_1, N_2, N_3 and N_4 would be turned ON, supplying base current to Q_2 which discharges the load. Transistor N_1 and N_2 draw current from the base of Q_1 thus rapidly turning this device OFF. This will cause the output to be a logic '0'. Thus the circuit realizes the NAND function. It may be noted that the static power consumption of the circuit is negligible neglecting reverse biased leakage currents.

D-BJT BiCMOS devices do not have the full V_{DD} to Ground logic swing of CMOS devices. The output High voltage (V_{OH}) is limited to V_{DD}-$V_{BE(Q1)}$ and output Low voltage (V_{OL}) is limited to V_{DD}-$V_{BE(Q2)}$. V_{ILmax} and V_{IHmin} were determined to be 2.2V and 2.7V respectively, by finding the $\frac{dV_{out}}{dV_{in}}$ = -1 points [12] on the voltage characteristics. The logic levels for BiCMOS are 0.6V to 2.2V for logic level '0' and 2.7V to 4.4V for logic level '1' [13]. Any voltage between 2.2V and 2.7V is considered indeterminate.

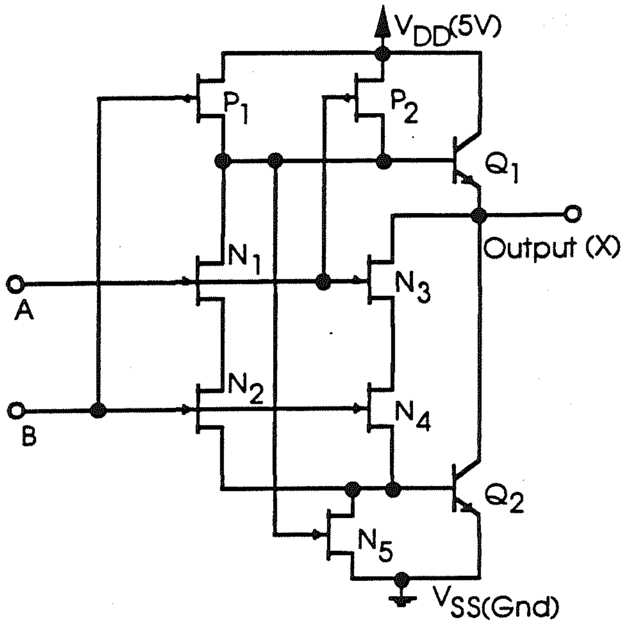


Figure 1: A D-BJT BiCMOS Nand.

3 Bridging of logical nodes without feedback

In this class of bridging faults, bridging of logical input and output nodes of logic elements are considered.

3.1 Hard bridging faults in D-BJT BiCMOS

In this subsection we assume hard shorts for bridging faults under consideration. In CMOS, the effects of bridging can be modeled by taking into account the resistances of the paths and bridging connections [6]. Consider a hard short (R_X=0Ω) between two D-BJT BiCMOS output inverters as shown in Figure 2. When the logic levels at nodes V_1 and V_2 are same, i.e., 00 or 11, then there is no direct path from V_{DD} to ground and hence, the same logic levels, i.e., 00 or 11, are maintained. Considering the case when one of the nodes is at logic level high (1) and the other node is at logic level low (0). A path is created between V_{DD} and ground leading to enhanced I_{DDQ}. Such output bridging faults in BiCMOS can be detected using power supply current monitoring (I_{DDQ} monitoring).

Since a hard short is assumed for the bridging fault, voltage levels appearing at node V_1 is the same as the voltage level appearing at node V_2 ($V_1 = V_2$). Consider the condition where V_1 is at logic level '1' (transistor Q_1 ON) and V_2 is at logic level '0' (transistor Q_2' ON). With the output bridging fault under consideration and since the parameters of transistors Q_1 and Q_2' are the same, the output settles at one-half of the output high ('1') logic level ($V_{OH}/2 = \approx 2.2$V), for input vectors A B = 0 1 and 1 0 as shown in Figure 3 for low resistance values of R_X (R_X up to ≈3Ω). Since there is a direct path from V_{DD} to ground, enhanced I_{DDQ} values are observed for output hard bridging faults. Hence, current monitoring (I_{DDQ}) techniques can be implemented to detect such faults.

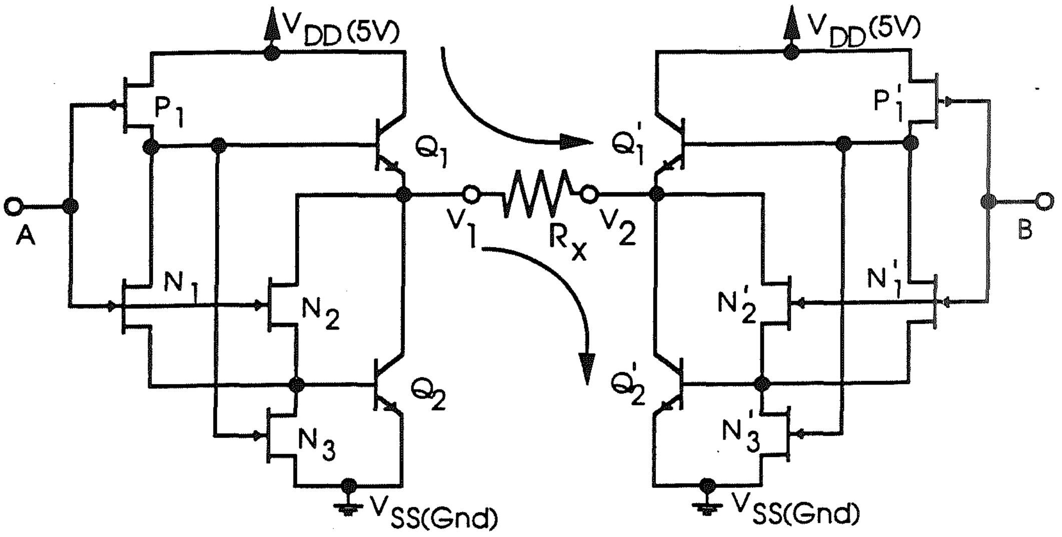


Figure 2: D-BJT BiCMOS output bridging fault (with significant resistance).

In a multilevel network, an input bridging fault can be analyzed similarly by considering the bridging fault to be output bridging fault of the previous level.

3.2 Output bridging with varying resistances

Since bridging is not a deliberate feature, it may not always be a hard short and in general may exhibit a significant resistance. A very high resistance may imply a bridging fault of no consequence. The range of values exhibiting significant impact on the logic values is investigated.

Consider the two BiCMOS output stages as shown in Figure 2 with unknown bridging resistance R_X. When the logic levels at V_1 and V_2 are same, i.e., 00(11), then there is no direct path from V_{DD} to ground, leading to the same logic levels, i.e., 00(11) being maintained at V_1 and V_2. When one of the nodes is at logic level 'High' (1) and the other node is at logic level 'Low' (0) with a bridging resistance of a significant value ($R_X > 0\Omega$), the output logic levels at V_1 and V_2 needs to be studied.

Spice simulation of the BiCMOS output bridging fault was done for varying bridging resistances (R_X) and the output logic levels for both the nodes V_1 and V_2 plotted are shown in Figure 3. The inference drawn from this simulation is as follows:

1. Node V_1 and V_2 remain at $\approx V_{OH}/2$ ($\approx$2.2V) for bridging resistance $R_X < \approx 3\Omega$. Output level of $\approx$2.2V is classified as logic level '0' for BiCMOS devices. However, the nodes V_1 and V_2 exhibit degraded noise immunity for these bridging resistances.

2. Node V_2 remains at logic level '0' for all bridging resistances. However, noise immunity of the node V_2 is degraded for lower bridging resistance values.

3. For bridging resistances $\geq\approx 3\Omega$ and $\leq\approx 40\Omega$, the logic level for node V_1 is in the 'Undefined' level. Though the output logic level is in the 'Undefined' level, current monitoring can be used to detect this bridging.

4. When bridging resistance is greater than $\approx 40\Omega$, the bridging is of no consequence at the logic level.

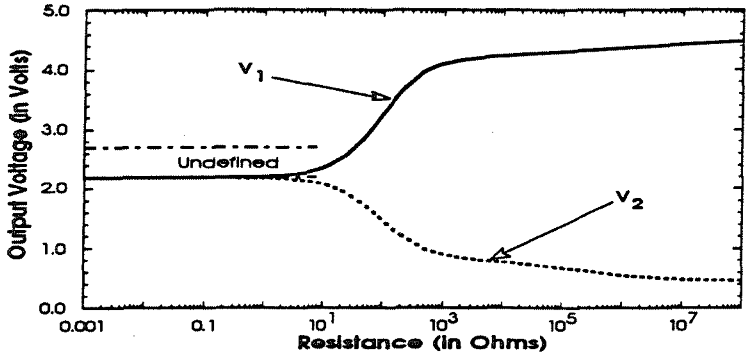


Figure 3: V_1 and V_2 under D-BJT BiCMOS output bridging fault (with signifcant resistance).

In Emitter Coupled Logic devices using bipolar transistors, an undefined logic level input to a successive stage may cause an 'undefined' voltage level to appear at the output [8]. In nMOS/CMOS devices, the 'Undefined' level is interpreted by the successive stage as either a '1' or '0' depending on the logic threshold. As in nMOS/CMOS devices, the successive stage of a BiCMOS device employing both bipolar and CMOS devices interprets an undefined level as either a '1' or '0' depending on the logic threshold. Figure 4 shows the current [i(V_{DD})] drawn by the devices for varying bridging resistances.

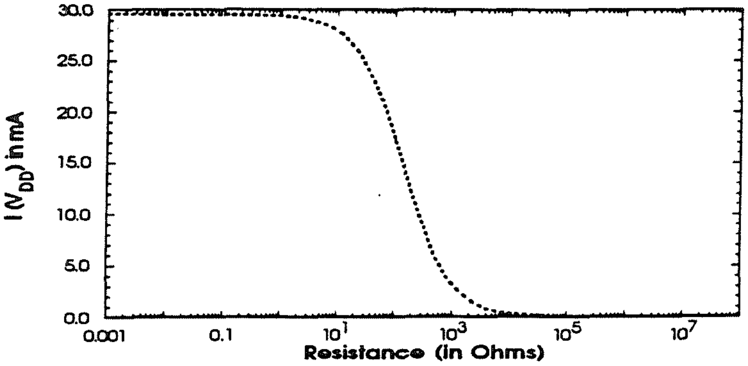


Figure 4: Current [$i(V_{DD})$] vs. bridging resistance for output bridging faults.

Estimation of noise immunity for both the nodes were carried out. The voltage level obtained at the two nodes (V_1 and V_2) under varying bridging resistances were obtained.

Noise immunity for 'High' logic level was estimated by computing the difference between the node voltage and V_{IHmin} (Minimum input voltage for logic level 'High'). Similarly, noise immunity for 'Low' logic level was estimated by computing the difference between V_{ILmax} (Maximum input voltage for logic level 'Low') and the node voltage.

Node V_2 remains at logic level 'L' irrespective of the bridging resistance. The noise immunity of this node is highly degraded at low bridging resistances. Figure 5 shows noise immunity for node V_2 under 'Low' logic level conditions. Degraded noise immunity at low bridging resistance can be seen in Figure 5. As the output bridging resistance increases, the noise immunity of node V_2 increases. Figure ?? shows noise immunity for node V_1 under 'High' logic level condition. It can be seen that under high bridging resistances, the logic level stays on the correct side (logic level 'H'). As the value of the bridging resistance is lowered ($\approx 40\Omega$), the noise immunity of the node V_1 for logic level 'High' keeps getting lower until noise immunity becomes zero. Further lowering the value of bridging resistance causes the node V_1 to be at an undefined level. Continuing the process of lowering the value of the bridging resistances (below $\approx 3\Omega$) results in logic level 'Low' at the node V_1 which is a faulty logic level with a very low noise immunity (≈ 0.006V), hence cannot be seen on the plot. A logic device being driven by this node may interpret this faulty logic level correctly.

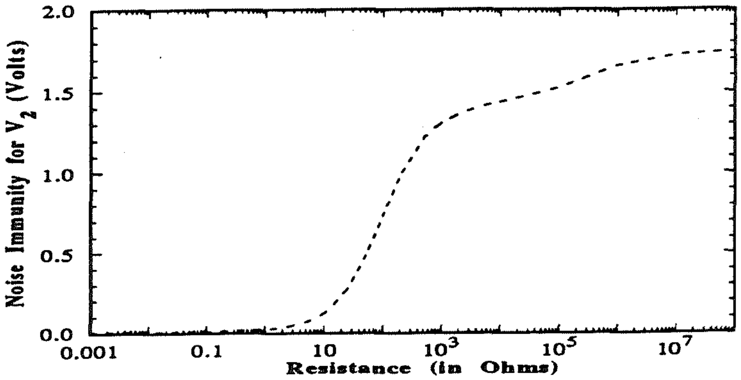


Figure 5: Noise immunity of node V_2 under D-BJT BiCMOS output bridging fault.

4 Bridging of logical nodes with feedback

This class of bridging faults deal with logical nodes deriving its logic value from another due to bridging. If the feedback loop does not contain clocked storage elements, then asynchronous feedback paths are frequently introduced transforming combinational block into an asynchronous sequential circuit. It has been shown that if a feedback loop contains an odd number of inversions, then oscillations can occur [20]. The anomalous behavior seen in this class of bridging faults depends on propagation delay, rise time, fall time as well as analog transfer characteristics of the bipolar transistors used in the device.

To analyze bridging faults of this class, consider a combinational block with inputs A_1 ... A_n and outputs Z_1 .... Z_m. Consider that input A_1 is bridged with output Z_1 through a sufficient bridging resistance (R_X). Assuming that an input vector is applied under which

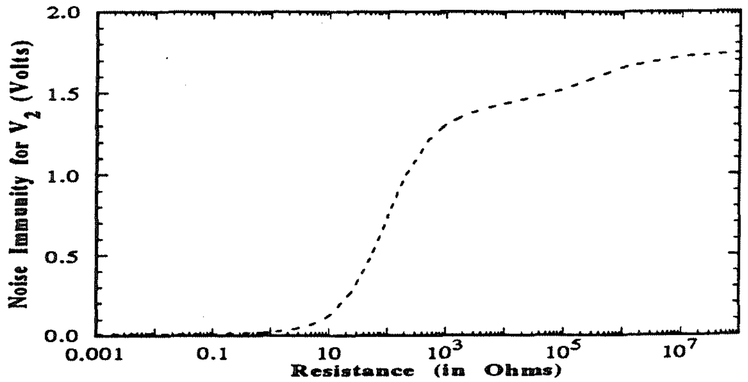


Figure 6: Noise immunity of node V_1 under D-BJT BiCMOS output bridging fault.

there is a sensitized path from A_1 to Z_1 through an odd number of inversions. When A_1 is logic level '1', Z_1 is '0' without feedback bridging. The situation will be different in the presence of feedback bridging. Considering the case when rise and fall times are much smaller compared to propagation delay, then the output logic level (Z_1) changes after the propagation delay. In the above situation, oscillations will occur where the propagation delay determines the clock period of oscillations.

On the contrary, when the propagation delay is smaller than the rise or fall times, the output will start changing before the input has time to stabilize. This leads to a situation where a rising or falling input signal starts influencing itself in the opposite direction before it reaches the switching threshold. This behavior needs to be analyzed using dynamic analysis taking into account the transistor characteristics, node capacitances etc., as static analysis alone would not suffice.

The two cases are illustrated in D-BJT BiCMOS gates with a feedback bridging fault using SPICE simulation for an odd number of inverter chain (3 inverters) as shown in Figure 7. When the propagation delay was small, for example, one D-BJT BiCMOS gate, oscillations did not occur and the voltage level stabilized at an intermediate level (undefined). Effects of bridging resistance under feedback bridging conditions were studied. For low bridging resistances, the voltage level stabilized at intermediate level. However, as the value of bridging resistance was increased, the effect of bridging became insignificant, resulting in correct logic levels.

When propagation delay is sufficiently large (with 3 or more inversions of D-BJT BiCMOS stages), oscillations occur. When the bridging resistance (R_X) is sufficiently small, the oscillations will touch either '0' or '1'. When propagation delay is just sufficiently large enough for oscillations, it will cause signals to cross the threshold voltage, and the requirement for oscillation [6] to occur is observed to be as given below.

$$t_{pd} > \frac{t_r + t_f}{2}$$

Where t_{pd} is the propagation delay, t_r is the rise-time and t_f is the fall-time.

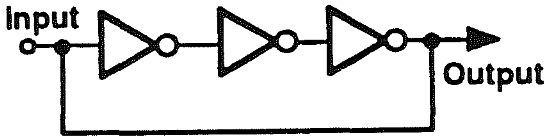


Figure 7: Feedback bridging across odd number of D-BJT BiCMOS inverters.

Example: SPICE simulation using a specific set of parameters for a chain of D-BJT BiCMOS NAND gates exhibited the following characteristics: t_{pd}/gate= 0.50 nSecs, t_r= 1.2 nSecs, t_f= 1.2 nSecs. The above characteristics suggest that oscillations would occur with a propagation delay of 1.5 nSecs for the device chain. Simulations do not exhibit oscillations with one device (t_{pd}=0.5nSecs) but oscillations do occur with a device chain of 3 (t_{pd}= 1.5 nSecs), as shown in Figure 8.

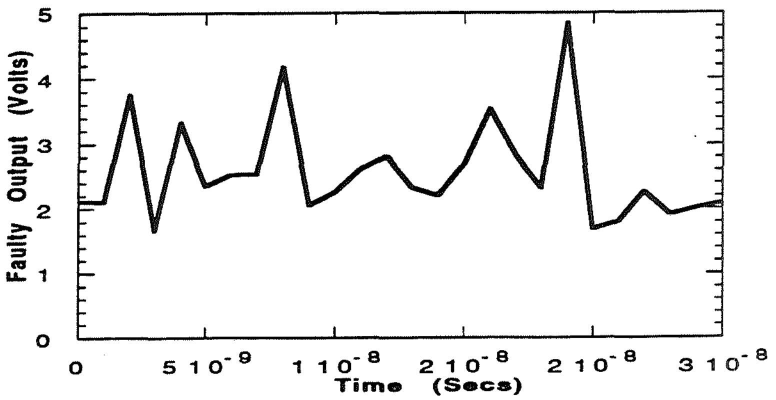


Figure 8: Plot illustrating oscillations under bridging fault with feedback brid ging.

5 Conclusions

Effects of bridging faults in BiCMOS devices were examined. Range of resistance values affecting the logic levels in BiCMOS under bridging of logical nodes without feedback was presented along with their effects on noise immunity. Power supply current (I_{DDQ}) monitoring for detection of bridging faults was shown. Feedback bridging faults resulting in oscillations was presented.

References

[1] A. R. Alvarez, *BiCMOS Technology and Applications*, Kluwer Academic Publishers, 1989.

[2] R. Haken, *'Process technology for submicron BiCMOS VLSI'*, IEEE Intl. Symp. on Circuits and Systems, pp. 1971-1974, 1990.

[3] B. C. Cole, *'Is BiCMOS the next technology driver?'*, Electronics, pp. 55-57, Feb. 1988.

[4] F. J. Ferguson and J. P. Shen, *'Extraction and Simulation of Realistic CMOS Faults using Inductive Fault Analysis'*, Proc. IEEE Int. Test Conf., pp. 475-484, 1988.

[5] J. P. Shen, W. Maly and F. J. Ferguson, *'Inductive Fault Analysis of nMOS and CMOS Integrated Circuits'*, Research Report No. CMUCAS-85-51, Dept. of Electrical and Computer Engineering Carnegie-Mellon University, August 1985.

[6] Y. K. Malaiya, A. P. Jayasumana and R. Rajsuman, *'A Detailed Examination of Bridging Faults'*, Proc. IEEE Conf. on Computer Design, pp. 78-81, 1986.

[7] R. Rajsuman, Y. K. Malaiya and A. P. Jayasumana, *'On Accuracy of Switch-Level Modeling of Bridging Faults in Complex Gates'*, Proc. IEEE Design Automation Conf., pp. 244-250, 1987.

[8] S. M. Menon, A. P. Jayasumana and Y. K. Malaiya, *'On Bridging Faults in ECL Circuits'*, Proc. 5th Intl. Conf. on VLSI Design, pp. 55-60, Jan. 1992.

[9] S. M. Menon, A. P. Jayasumana and Y. K. Malaiya, *'Fault Modeling of ECL devices'*, Electronics Letters, pp. 1105-1108, July 1990.

[10] S. M. Menon, Y. K. Malaiya and A. P. Jayasumana, *'Fault Modeling and Testable Design of 2-level Complex ECL Gates'*, Proc. 4th CSI/IEEE Intl. Symp. on VLSI Design, pp. 23-28, Jan. 1991.

[11] M. Favalli, M. Dalpasso, P. Olivo and B. Ricco, *'Analysis of steady state detection of resistive bridging faults in BiCMOS digital ICs'*, Intl. Test Conf., pp. 466-474, 1992.

[12] D. A. Hodges and H. G. Jackson, *Analysis and Design of Digital Integrated Circuits*, McGraw-Hill, Ch. 3, 1983.

[13] S. M. Menon, *'Modeling, Analysis and Design for Testability of Bipolar/MOS VLSI'*, Ph.D. Dissertation, Dept. of Electrical Engineering, Colorado State University (in preparation).

[14] P. Banerjee and J. A. Abraham, *'Generating Tests for Physical Failures in MOS Logic Circuits'*, Proc. Intl. Test Conf., pp. 554-559, Oct. 1983.

[15] Y. K. Malaiya, *'Testing Stuck-On Faults in CMOS Integrated Circuits'*, Proc. IEEE Intl. Conf. on Computer-Aided Design, pp. 248-250, Nov. 1984.

[16] J. Gailiay, Y. Crouzet and M. Vergniault, *'Physical versus logical fault models in MOS LSI circuits: Impact on the testability'*, IEEE Trans. Computers, Vol. C-29, pp. 527-531. June 1980.

[17] C. Timoc, M. Buehler, T. Griswold, C. Pina, F. Stott and L. Hess, *'Logical models of physical failures'*, Proc. IEEE Intl. Test Conf., pp. 546-553, Nov. 1983.

[18] C. C. Beh, K. H. Arya, C. E. Radke and K. E. Torku, *'Do stuck fault models reflect manufacturing defects?'*, Proc. IEEE Test Conf., pp. 35-42, Nov. 1982.

[19] J. Hlavica and E. Kotteck, *'Fault Model for TTL Circuits'*, Digital Processes, Vol. 2, pp. 169-180, 1976.

[20] S. Xu and S. Y. H. Su, *'Testing Feedback Bridging Faults among Internal Input and Output Lines by Two Patterns'*, Proc. Intl. Conf. on Circuits and Computers, pp. 214-217, Sep. 1982.

Fault Detection in Digital and Analog Circuits Using an i_{DD} Temporal Analysis Technique

J. Beasley
Department of Engineering Technology
and
the Department of Electrical and
Computer Engineering
New Mexico State University
Las Cruces NM

D. Magallanes, A. Vridhagiri and Hema
Ramamurthy
Department of Electrical and Computer
Engineering
New Mexico State University
Las Cruces NM

Mark DeYong
Intelligent Reasoning Systems Inc.
Las Cruces NM

Abstract - **This paper presents an i_{DD} temporal analysis technique which is used to detect defects (faults) and fabrication variations in both digital and analog ICs by pulsing the power supply rails and analyzing the temporal data obtained from the resulting transient rail currents. A simple bias voltage is required for all the inputs, to excite the defects. This paper presents data from hardware tests supporting this technique.**

1 Introduction

The complexity of integrated circuit testing is rapidly increasing, due to the more prevalent use of mixed-mode (digital-analog) circuits [1]. This paper presents a temporal analysis technique for detecting defects in both digital and analog circuits by using a unified method that analyzes the transient response of the pulsed power supply rails. The method is called i_{DD} Pulse Response Testing [2]. Studies of the transient supply current have shown that it is possible to detect defects in ICs using the i_{DD} Pulse Response Test method [2,3]. In fact, it appears that it may be possible to monitor the "health" of an integrated circuit through supply line monitoring only.

Neural Networks are used to analyze the temporal data generated by pulsing the power supply rails. The temporal analysis technique presented incorporates either the use of a conventional neural network, using a back propagation algorithm, [4,5] which has been "trained" to detect a defect-free or defective condition or by using a biologically realistic hardware model of an axon hillock [6] which has been "adapted" to detect defects. The output of the hillock is a pulse stream. Each pulse stream is a signature uniquely identifying a given condition of the circuit.

2 The Test Method

The data used in the temporal analysis is obtained by applying the i_{DD} Pulse Response Test to the circuit under test. i_{DD} Pulse Response Testing consists of pulsing the V_{DD} and V_{SS}

voltage rails while applying a simple bias voltage to the inputs, then performing analysis on the temporal characteristics of the resulting transient rail current (i_{DD}). The bias voltage for the inputs is set to the midpoint in the power supply rail voltages. For example, the inputs to a digital $5V$ circuit will be set to $2.5V$. For a device which requires a dual power supply (i.e. $\pm 5V$) the inputs are set to $0.0V$. SPICE simulations verified that this particular bias technique will enable all the transistors within the IC to turn on at some point in the pulsing process. Simulations and hardware tests have shown that the IC turn-on behavior (transient current) will change for defective circuits [2,3]. These changes are shown to be detectable using the i_{DD} temporal analysis technique accompanied by data analysis using Neural networks.

3 Temporal Analysis Using a Conventional Neural Network

The back-propagation algorithm is a conventional neural network which is designed to minimize the mean square error between the actual output of the network and the desired output. [4]and [5]. The network is trained by initializing the weights (wt#) to random values. A set of input patterns ($x_1..x_{46}$, see Figure 1) and an expected output (see Table 2) for each training pattern is then input to the network. The weights are adjusted after each pass until the desired and actual output values for the training pattern satisfy a desired accuracy.

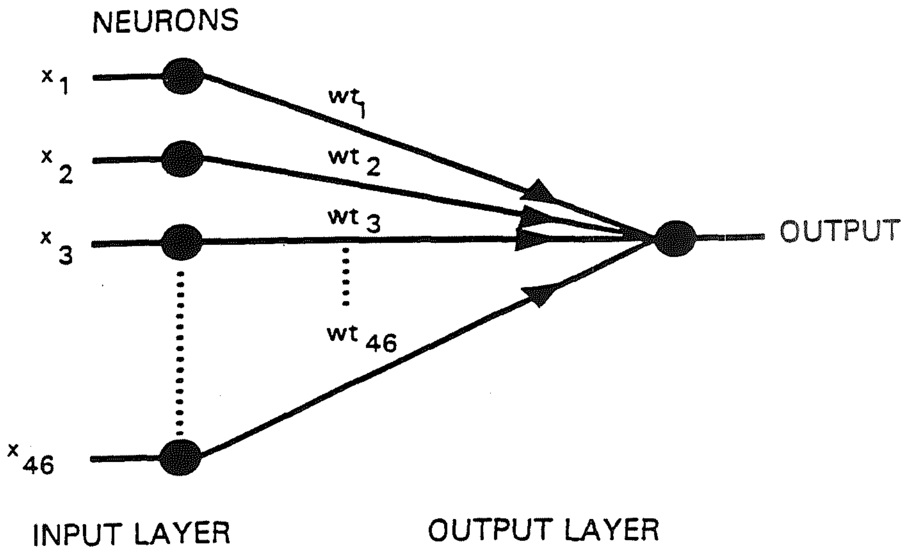


Figure 1: The block diagram of the 2-layer Neural network back-propagation algorithm used to analyze the Timing Control IC data.

The back propagation algorithm used in these experiments [1] was compiled for the PC

[1] The back-propagation algorithm software routine used in these experiments was written by Intelligent Reasoning Systems, Inc. and modified by the authors.

using Borland's Turbo C software. The i_{DD} Pulse Response Test was conducted on six digital integrated circuit devices provided by Sandia National Laboratories. The devices tested are functionally equivalent to the National Semiconductor NS32C201 Timing Control Unit 24-pin device. These particular parts were manufactured for Sandia by AT&T using a CMOS p-well process with a 1.25 micron feature size. The devices contain approximately 3000 transistors. These devices were pre-tested at Sandia using the Advantest IC tester. It was determined that four of the six devices contained gate-oxide shorts. Table 2 provides a summary of the six devices tested.

Device number	Description
1228	Passed gate-oxide (gox) test
1258	Passed gox test
1252	Failed gox test - 1 mA defect
1399	Failed gox test - 4 mA defect
1328	Failed gox test - 250 uA defect
1402	Failed gox test - 250 uA defect

Table 1: Timing Control Unit ICs

The devices were then tested using the i_{DD} Pulse Response Test method. The inputs to the device were set to $2.5V$. The V_{DD} and V_{SS} rails were pulsed from the $2.5V$ bias to $+5V$ and $0.0V$ at the 10 us pulse rate and approximately .3 us rise and fall times. Plots obtained from the tests of the six devices are shown in Figure 2. The plots show very little difference in the response for the defect free and defective devices.

The Timing Control Unit ICs contain a significantly large number of transistors. Therefore it can be concluded that single defects are not obvious in large transistor circuits. This result was expected and yet the data obtained from the pulse test can still be used to determine the condition of the circuit under test. To analyze the data, the waveforms for all six devices were downloaded to a PC from a Tektronix model 2221A 100 MHz Digital Storage Oscilloscope. The data analyzed was limited to the region shown in Figure 2 beginning at the START point and terminating at the END point. This portion of the waveform consists of 46 samples. The Start and End points were arbitrarily selected. The data was analyzed using a two-layer neural network back-propagation algorithm. A block diagram of the network is shown in Figure 1.

Two training patterns were used to test the effectiveness of the back propagation to detect defects. Training pattern (a) consisted of one defect free IC (1228) and the gate-oxide defect IC - 1252. Training pattern (b) consisted of the two defect free ICs (1228 and 1258) and the gate-oxide defect IC - 1252. The patterns were selected so that the neural network could be trained to distinguish a defect free (ideal) condition from a defective or irregular state (the gate-oxide defect). The neural network, using the back propagation algorithm, can be trained so that the weights will provide a specific numerical response for a given input. In this case, the back propagation neural network was trained to provide a response of .5 for the defect free circuits (1228 and 1258) and a .3 for the gate-oxide defect provided by (1252).

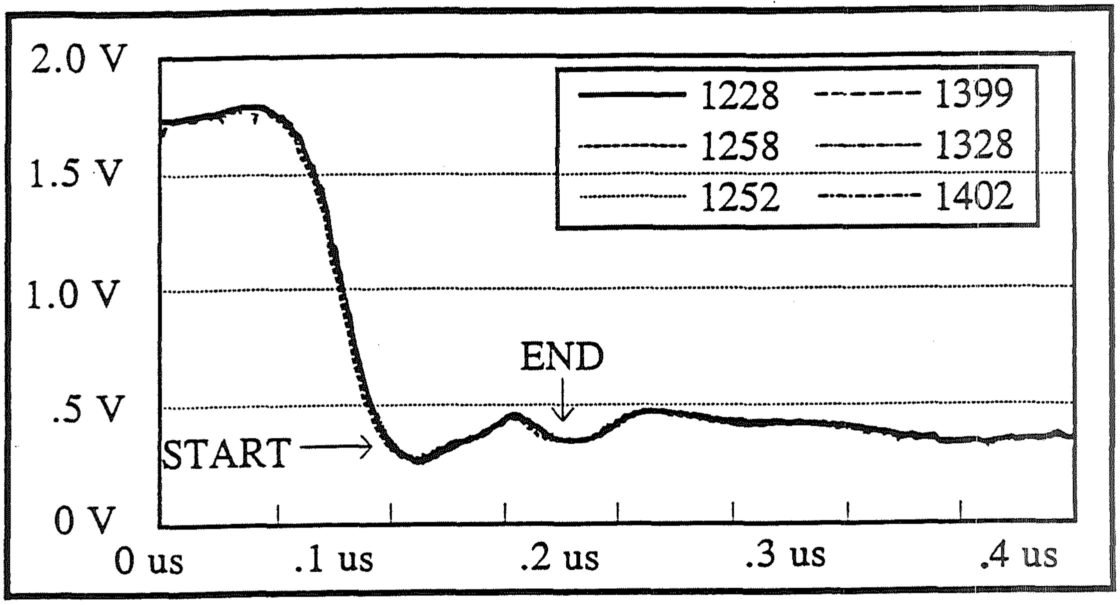


Figure 2: i_{DD} Pulse Response obtained for the six Timing Control Unit ICs.

The network was trained for an accuracy of 1of the training are provided in Table 2.

In both cases, the neural network provided the correct response for the circuit. Three groupings are provided in the response:

1. defect free [.495,.462]a and [.498,496]b
2. heavy gate-oxide defects [.325,.290]a and [.306,.265]b
3. light gate-oxide defects [.407,.372]a and [.401,.365]b.

The defect free responses were expected since they were part of the training phase but devices 1399, 1328 and 1402 provided responses consistent with the Advantest measurements without requiring any training.

4 Temporal Analysis Using a Transistor Model of an Axon Hillock

Current computational models of neurons, such as the non-linear perceptron used in back propagation networks, are oversimplified, lacking the temporal characteristics which are the true essence of analog computation [6]. A basic neural system consists of four main components: dendritic trees, synapses, axon hillocks and axons. The dendritic tree performs continuous time summations of the currents created by the synapses (inhibitory and excitatory). The dendritic tree termination is the axon hillock input, which produces a pulse or action potential at it's output whenever its input voltage surpasses a predetermined threshold level. A block diagram of the Neural Pulse Generator is shown in Figure 3.

Device#	Expected Response	Trained Responsea	Trained Responseb
1228[a,b]	.5	.495	.498
1258[b]	.5	.462	.496
1252[a,b]	.3	.325	.306
1399	undefined	.290	.265
1328	undefined	.407	.401
1402	undefined	.372	.365

[a]: This training phase only used ICs 1228 and 1252.

[b]: This training phase used ICs 1228, 1258 and 1252.

Table 2: Results of the Neural Network Training

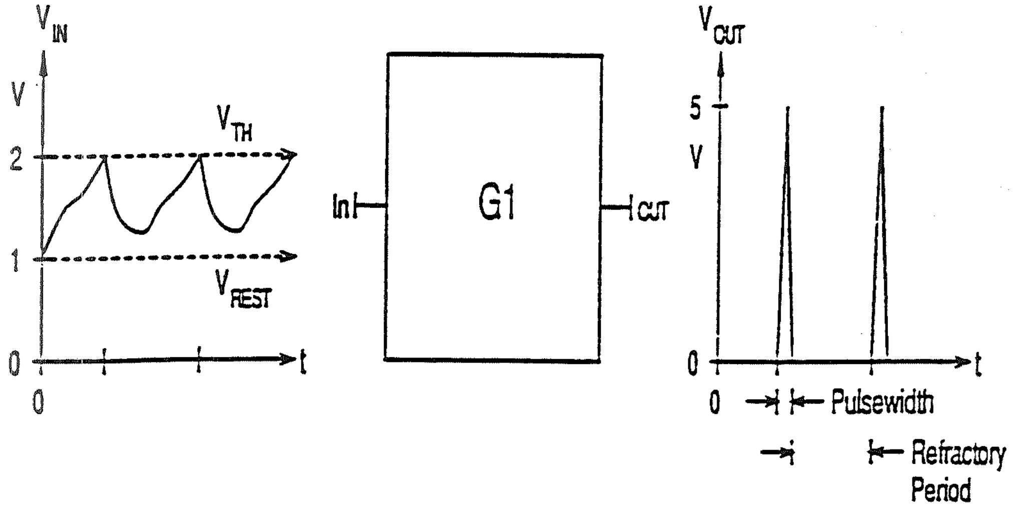


Figure 3: Neural Pulse Generator Block Diagram

This model exhibits behavior similar to those found in true electrophysical neurons. The axon hillock in its simplest form is a pulse generator in a neural system. An advantage with a hardware model is that it is a continuous time processor whose present state is a function of all the previous inputs. A block diagram of the Biologically Realistic Hillock hardware model is shown in Figure 4.

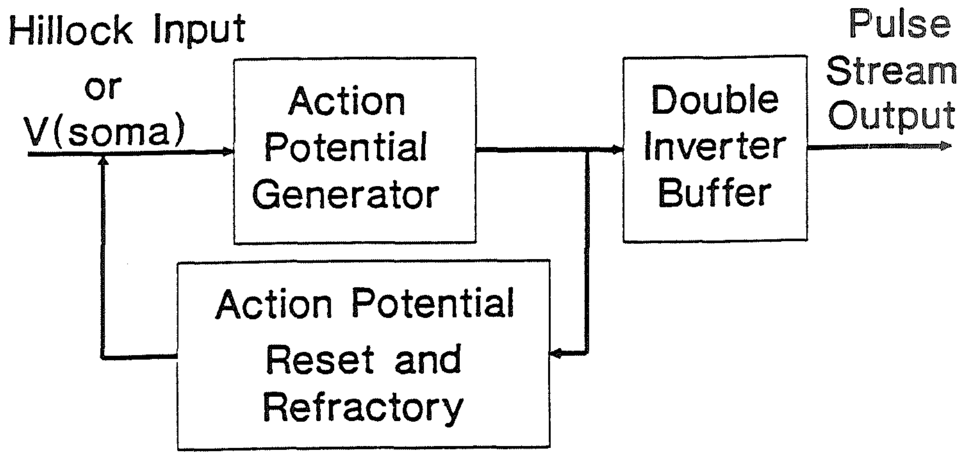

Hillock Input
or
V(soma)
Action Potential Generator
Double Inverter Buffer
Pulse Stream Output
Action Potential Reset and Refractory

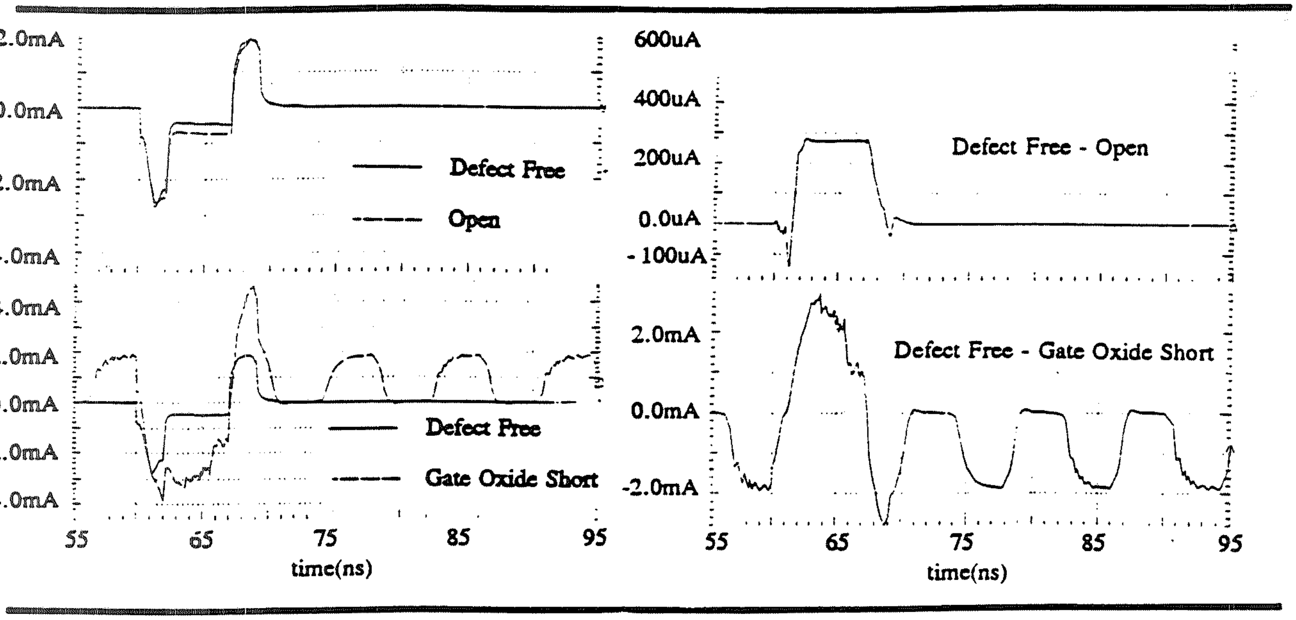
Figure 4: Biologically Realistic Hillock Hardware Model Block Diagram

The hardware model has three significant variable inputs: the Action Potential Generator (APG), the Action Potential Reset and Refractory circuit (APRR) and the Double Inverter Buffer (DIB). The hillock recieves an input, calculates the necessary updated potentials, and produces an output. The APG subcircuit monitors the soma potential (hillock input) for the condition $V_{soma} - V_{th} > 0$, at which time it produces the rising edge of an AP. This rising edge causes the APRR to become active and pull the soma potential towards ground, which causes the state $V_{soma} - V_{th} \leq 0$, thus resetting the APG which causes the falling edge of the AP. The double inverter buffer isolates the hillock from the load.

HPSPICE simulations using the hillock model and i_{DD} Pulse Response Test have produced favorable results. The results demonstrated the ability of the hillock model to distinguish between defect free circuits, open and gate-oxide faults. Figure 5(a) shows the pulsed transient power rail current produced by HSPICE. The circuit under test is a three stage inverter circuit with each stage consisting of six CMOS inverters. The defect is located in the first stage. Figure 5(b) shows the differences between the defect free circuit and the open circuit and the defect free and gate oxide defects.

Similar results have been obtained with the defect in various stages. With even the small difference shown in Figure 5b the hillock model is able to discern between the two conditions as shown in their distinctly different pulse streams. This pulse stream output for the hillock is shown in Figure 6.

A hardware circuit board of the Biologically Realistic Hillock was constructed. Test results conducted using the hillock on a set of digital ICs provide similar results to the



a) Transient rail current

b) The differences in the defect free and defective currents.

Figure 5:

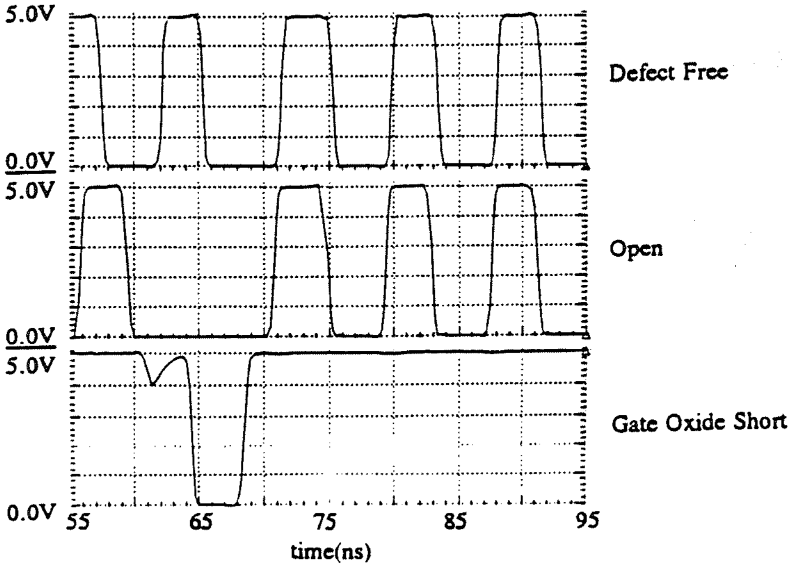


Figure 6: Hillock Output Pulse Streams

simulations. The i_{DD} Pulse Response Test was conducted on four 74HC04 14-pin Hex Inverter ICs. The data was obtained using a Tektronix 2221A 100 MHz Digital Storage Oscilloscope. The test consisted of three defect free ICs and one defective IC [2]. The defect free ICs demonstrate near identical pulse streams while the defective IC produces a distinquely different pulse stream at the output of the hillock. The results of this test are shown in Figure 7.

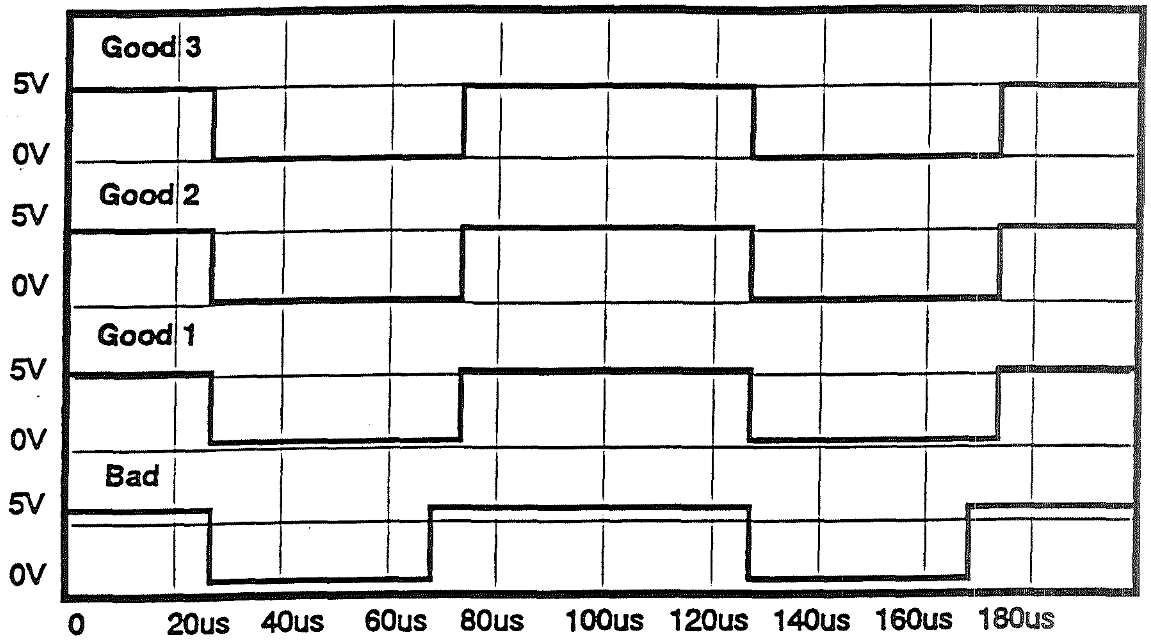


Figure 7: Hillock Output Pulse Streams Using the Transistor Model

5 Conclusions

This paper has shown that defects in VLSICs can be detected by performing temporal analysis on the data obtained from the i_{DD} Pulse Response of a circuit under test. Gate-oxide shorts and opens [Figures 5(a) and 5(b)] in CMOS circuits have been shown to be detectable by this method. An important benefit of the proposed technique is that it is essentially test vector independent. It has also been shown that the same test procedure can be used for both digital and analog circuits [2,3] and, hence, may be used in a mixed-mode circuit. Test time and complexity have been minimized to the point of requiring only a single pulse or a short series of pulses to the V_{DD} and V_{SS} rails to analyze the circuit under test. The application of the proposed method in more complex circuits and in mixed-mode VLSI circuits is currently under investigation.

[2] These devices were previously tested for functionality

6 Acknowledgements

This work was supported in part by grant HRD9253013 from the National Science Foundation. A special thanks is given to Keith Treece of Sandia National Laboratories (SNL) for providing the pre-tested Timing Control Unit ICs.

References

[1] L. Brust and M.S. Tsay, "Mixing signals and voltages on chip," IEEE Spectrum, August 1993.

[2] J. Beasley, H. Ramamurthy, J. Ramirez-Angulo and M. DeYong, "i_{DD} Pulse Response Testing of Analog and Digital CMOS Circuits," 1993 International Test Conference.

[3] J. Beasley, H. Ramamurthy, J. Ramirez-Angulo and M. DeYong, "Fault Detection in VLSI Circuits Using i_{DD} Spectrum Analysis," 1993 European Conference on Circuit Theory and Design.

[4] R.P. Lippmann, "An Introduction to Computing With Neural Nets," IEEE ASSP Magazine, April 1987, pp 4-22.

[5] R.C. Eberhart and R.W. Dobbins, Neural Network PC Tools: A Practical Guide, San Diego and New York: Academic Press, Inc., 1990.

[6] M. DeYong, "VLSI Hybrid Temporal Neural Networks for Signal Processing," Ph.D. Dissertation, New Mexico State University, 1992.

Test Pattern Generation for ILA Sequential Circuits[†]

Yu Feng
Allied Signal Inc.
General Aviation Avionics
Olathe, KS 66062-1212

James F. Frenzel
Department of Electrical Engineering
University of Idaho
Moscow, ID 83844-1023

Gary K. Maki
NASA Space Engineering Research Center
University of New Mexico
Albuquerque, New Mexico 87106

Abstract - **This paper presents an efficient method of generating test patterns for sequential machines implemented using one-dimensional, unilateral, iterative logic arrays (ILAs) of BTS pass transistor networks. Based on a transistor level fault model, the method affords a unique opportunity for real-time fault detection with improved fault coverage. The resulting test sets are shown to be equivalent to those obtained using conventional gate level models, thus eliminating the need for additional test patterns. The proposed method advances the simplicity and ease of the test pattern generation for a special class of sequential circuitry.**

1 Introduction

Iterative logic arrays (ILAs) are ideal as VLSI subsystems because of their regular structure, and ease of design, layout, and testing. CMOS binary tree structured (BTS) pass transistor networks provide the advantage of lower power consumption, regular circuit structure with high density, and a low transistor count [1]. Synchronous sequential machines have been designed using this design technology by Liu [2]. This motivated research into the important subject of ILA testing. This paper focuses on testing a special, one-dimensional ILA which implements the function of a synchronous sequential circuit, with each cell in the ILA comprised of combinational logic implemented with BTS pass transistor networks. The faults addressed by this work are gate level stuck-at faults and transistor stuck-open faults. The transistor level ILA testing provides the advantage of test simplicity and improved fault coverage, as well as real-time fault detection capability.

[†] This research was supported in part by NASA under Space Engineering Research Center Grant NAGW-3293.

y_1	y_2	y_3	y_4	y_5	y_6		I_1	I_2	I_3
1	0	0	0	0	0	S_1	S_2	S_5	S_4
0	1	0	0	0	0	S_2	S_1	S_3	S_4
0	0	1	0	0	0	S_3	S_1	S_3	S_4
0	0	0	1	0	0	S_4	S_1	S_3	S_6
0	0	0	0	1	0	S_5	S_1	S_5	S_6
0	0	0	0	0	1	S_6	S_1	S_5	S_6

Table 1: Sample Flow Table

2 Background

2.1 BTS pass transistor ILA design review

An ILA circuit consists of an array of identical cells. The parallel inputs to a cell are referred to as primary inputs (PIs) and the serial inputs to a cell are the outputs of the previous cell. Table 1 gives a flow table for a sequential state machine which will be used as an example throughout this paper. The states are encoded using a one-hot code and the next state equations are given as:

$$\begin{array}{lcllllllllllll}
Y_1 & = & 0 & + & y_2I_1 & + & y_3I_1 & + & y_4I_1 & + & y_5I_1 & + & y_6I_1 \\
Y_2 & = & y_1I_1 & + & 0 & + & 0 & + & 0 & + & 0 & + & 0 \\
Y_3 & = & 0 & + & y_2I_2 & + & y_3I_2 & + & y_4I_2 & + & 0 & + & 0 \\
Y_4 & = & y_1I_3 & + & y_2I_3 & + & y_3I_3 & + & 0 & + & 0 & + & 0 \\
Y_5 & = & y_1I_2 & + & 0 & + & 0 & + & 0 & + & y_5I_2 & + & y_6I_2 \\
Y_6 & = & 0 & + & 0 & + & 0 & + & y_4I_3 & + & y_5I_3 & + & y_6I_3
\end{array} \qquad (1)$$

To implement the sequential circuit in an ILA architecture the next state equations must be converted to pass logic functions, where each control variable passes a single pass variable or a constant, as shown in Equation 2.

$$\begin{array}{lcl}
Y_1 & = & y_2(I_1) + \overline{y}_2\,(y_3(I_1) + \overline{y}_3\,(y_4(I_1) + \overline{y}_4\,(y_5(I_1) + \overline{y}_5\,(y_6(I_1) + \overline{y}_6(0))))) \\
Y_2 & = & y_1(I_1) + \overline{y}_1(0) \\
Y_3 & = & y_2(I_2) + \overline{y}_2(y_3(I_2) + \overline{y}_3(y_4(I_2) + \overline{y}_4(0))) \\
Y_4 & = & y_1(I_3) + \overline{y}_1(y_2(I_3) + \overline{y}_2(y_3(I_3) + \overline{y}_3(0))) \\
Y_5 & = & y_1(I_2) + \overline{y}_1(y_5(I_2) + \overline{y}_5(y_6(I_2) + \overline{y}_6(0))) \\
Y_6 & = & y_4(I_3) + \overline{y}_4(y_5(I_3) + \overline{y}_5(y_6(I_3) + \overline{y}_6(0)))
\end{array} \qquad (2)$$

Each equation may be realized with a BTS pass transistor network, as shown in Figure 1 for Y_1. The circuit has a regular structure; each part surrounded by the dashed rectangle can be considered as a cell, with the preceding cell providing the serial input. Each BTS cell is a MUX and the circuit can be implemented in the form of an iterative logic array. In general, the MUX cell function is $Q_i = y_i(I_j) + \overline{y}_i(Q_{i+1})$ where $i = [1, \ldots, 6]$.

The schematic of the synchronous sequential circuit ILA network and the two types of cells, the MUX cell and the ZERO cell, are shown in Figure 2. The ZERO cell is defined to preserve the circuit regularity and simplify automatic layout. In this one-dimensional

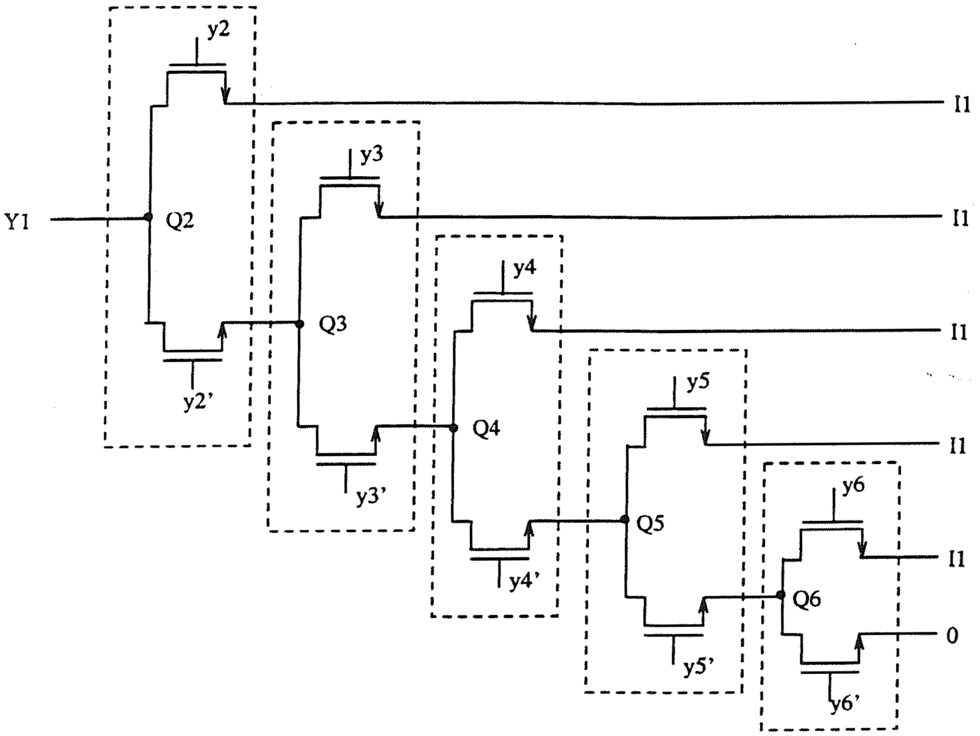


Figure 1: BTS circuit for State Variable Y_1

ILA architecture for sequential circuits, signals flow from right to left and each next state variable is generated by a slice of concatenated, combinational ILA cells. The outputs of the left most cells serve as D-inputs to the flip-flops and the flip-flop outputs are fed back to to generate the new next state variables. For the purposes of testing, the Y_i (or y_i) will be considered to be the primary outputs; there are no parallel cell outputs. This ILA architecture for synchronous sequential circuits simplifies the design and layout procedure by facilitating automatic placement of the two basic cells. Further details of ILA-based synchronous sequential circuit design have been discussed in [2].

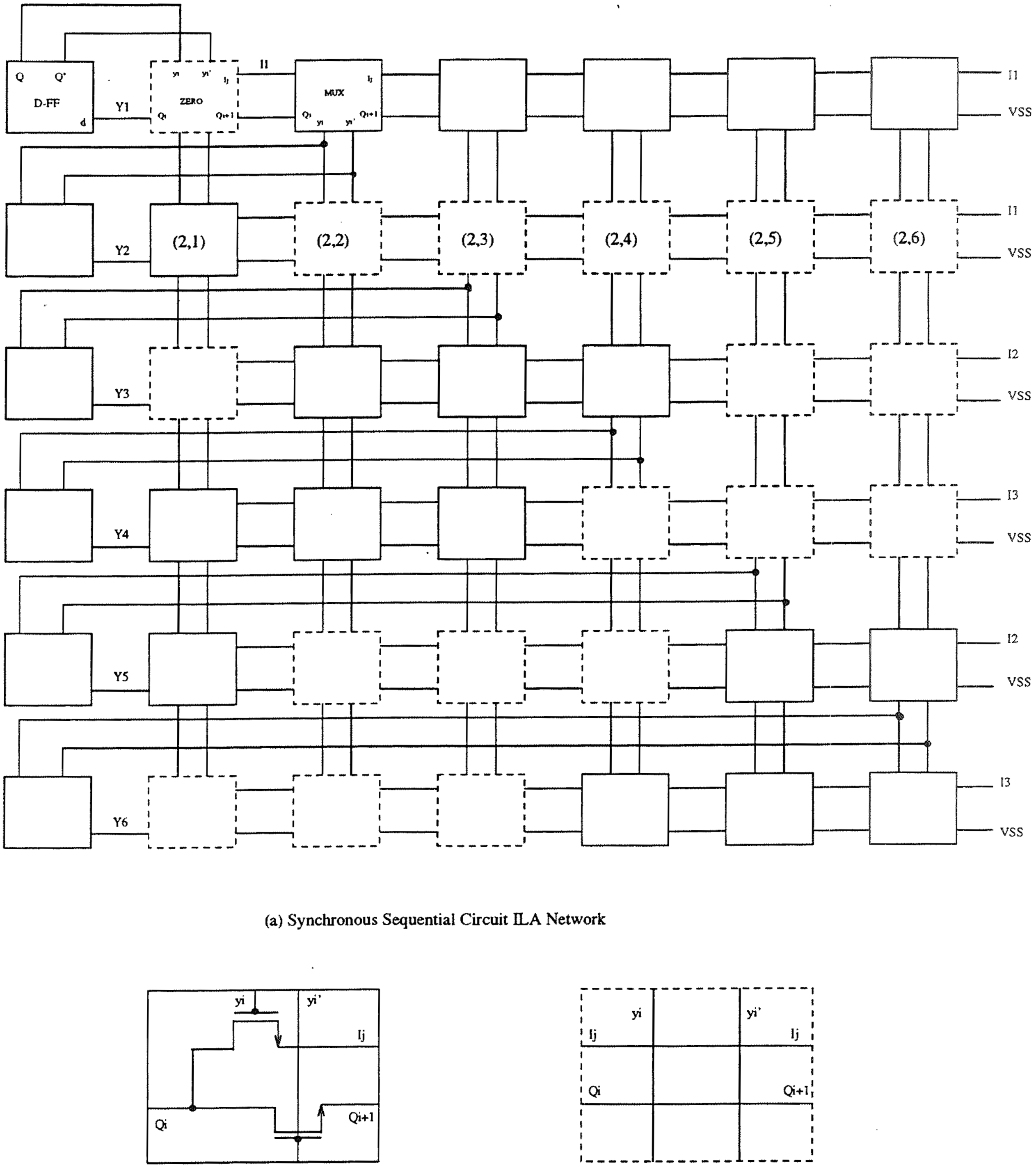
2.2 Fault Models

The majority of the circuit faults will be in the MUX cells. Two fault models will be used to represent these faults:

Stuck-at The stuck-at fault model is used to represent defects which result a gate input or output being fixed to either a logic 0 (s-a-0) or logic 1 (s-a-1).

Stuck-open The transistor stuck-open (s-op) fault model is used to represent defects which leave a FET permanently in a non-conducting state. For CMOS networks, the detection of a stuck-open fault needs two test vectors: an initialization test vector and an actual test vector. This is called a *two-pattern test* [4].

Throughout this work it will be assumed that there is at most one faulty cell in the array and that the circuit is time invariant (i.e., transient errors will not be considered).

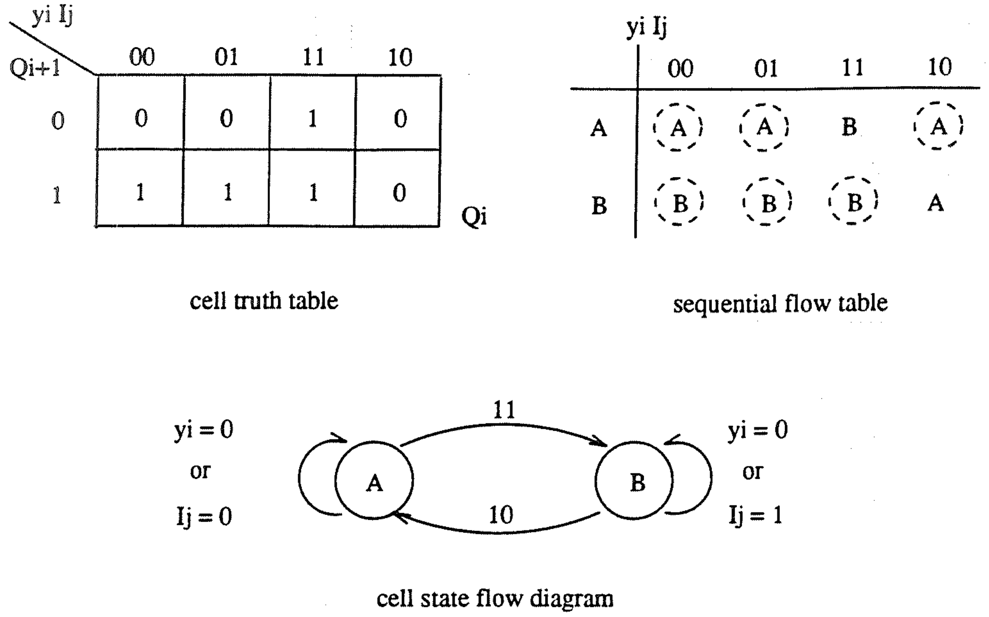


(a) Synchronous Sequential Circuit ILA Network

(b) ILA - MUX Cell

(c) ILA - ZERO Cell

Figure 2: The ILA network to Implement the Flow Table



yi Ij Qi+1	00	01	11	10
0	0	0	1	0
1	1	1	1	0

Qi

cell truth table

yi Ij	00	01	11	10
A	A	A	B	A
B	B	B	B	A

sequential flow table

cell state flow diagram

Figure 3: MUX Cell Function Description

3 Logic Gate level ILA fault detection

The truth table verification method developed by Friedman, Menon and Dias [5, 3] will be used to perform ILA fault detection of stuck-at faults at the logic gate level. In this method, a sequential circuit is used to represent the ILA cell. The flow table of the sequential circuit can then be used to describe a cell's behavior [5]. Figure 3 shows the cell's truth table, its corresponding sequential circuit flow table, and the state flow diagram.

Table 2 lists the potential stuck-at faults and the corresponding test patterns. The 'X' indicates a don't care on a cell input and $\hat{Q}_i$ represents the cell output under the occurrence of the fault. By combining faults which are detectable using the same patterns, a compressed fault list table is obtained as shown in Table 3. The complete test pattern is comprised of inputs that will activate the fault within the MUX cell and propagate it to the output. This will be shown with the benefit of an example.

Example Assume that the cell in Row 1, Column 3 is defective as shown in Figure 4 and the defect is modeled by Fault 1 in Table 2. First, to activate the fault the input for Cell (1,3) should be $Q_4 I_1 y_3 =$ X11. Second, to maintain a valid 1-hot-coding the y_i inputs to cells right of Cell (1,3) should be 0. Lastly, the cells left of Cell (1,3) must propagate the fault's effect; this is accomplished by selecting $y_i = 0$. The complete test pattern is shown in Figure 4.

By moving the inputs of Cell (1,3) to different cells, test patterns for testing the remaining MUX cells can be obtained. Following the same principles, the series of test inputs for other faults can be generated. The complete test set for testing all of the MUX cells under input I_j in the array is shown in Table 4. It is important to note that, with the exception of 13 and 14, each of the test patterns is consistent with the 1-hot-code state assignment. This allows the faults to be detected during normal operation without additional circuitry for forcing

	Fault	Test			Output	
		Q_{i+1}	I_j	y_i	Q_i	$\hat{Q}_i$
1	I_1 s-a-0	X	1	1	1	0
2	I_1 s-a-1	X	0	1	0	1
3	y_i s-a-0	X	1	1	1	0
4	y_i s-a-1	0	1	0	0	1
5	$\overline{y}_i$ s-a-0	1	X	0	0	1
6	$\overline{y}_i$ s-a-1	1	0	1	1	0
7	Q_{yi} s-a-0	X	1	1	1	0
8	$Q_{\overline{yi}}$ s-a-0	1	X	0	1	0
9	Q_{yi} s-a-1	X	0	1	0	1
10	$Q_{\overline{yi}}$ s-a-1	0	X	0	0	1

Table 2: MUX Cell Stuck-at Tests

Faults	Test			Output	
	Q_{i+1}	I_j	y_i	Q_i	$\hat{Q}_i$
1,3,7	X	1	1	1	0
2,9	X	0	1	0	1
5,8	1	X	0	0	1
4,10	0	1	0	0	1
6	1	0	1	1	0

Table 3: Compressed List

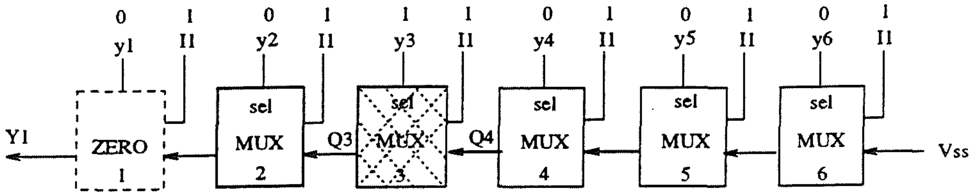


Figure 4: Test for Fault 1 in Table 2

unnecessary state transitions. Test patterns 13 and 14 are needed to test the cells in the rightmost column. Furthermore, Test 14 requires that a 1 be applied to the Q_{i+1} input of those cells to test for a s-a-0.

4 Transistor Level Fault Detection

Gate level fault models are limited to developing tests for stuck-at faults. It has been shown that CMOS circuits have failure modes which can't be modeled as stuck-ats. In this section a new method of test pattern generation is presented based on transistor level fault models. It turns out to be simpler and detects transistor stuck-open faults, as well as the stuck-at faults.

4.1 Pass Transistor Faults

In this class of sequential circuits the individual multiplexers are implemented using a BTS pass transistor network consisting of two NMOS transistors, controlled by complementary signals as shown in Figure 2. The function of a BTS pass network may be specified by a sum of product expression, with each product term consisting of a control variable (Pi) and a pass variable (Vi). Thus, the function of Cell (j, i) is given by the pass transistor expression

$$Q_i = y_i(I_j) + \overline{y_i}(Q_{i+1}),$$

Test	Control Inputs (present state)						I_j	Output
	y_1	y_2	y_3	y_4	y_5	y_6	I_j	Y_i
1	1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	1	1
3	0	1	0	0	0	0	0	0
4	0	1	0	0	0	0	1	1
5	0	0	1	0	0	0	0	0
6	0	0	1	0	0	0	1	1
7	0	0	0	1	0	0	0	0
8	0	0	0	1	0	0	1	1
9	0	0	0	0	1	0	0	0
10	0	0	0	0	1	0	1	1
11	0	0	0	0	0	1	0	0
12	0	0	0	0	0	1	1	1
13	0	0	0	0	0	0	X	0
14	0	0	0	0	0	0	X	1

Table 4: Complete test patterns for MUX cells with input I_j

where Q_{i+1} is the output of MUX Cell $(j, i+1)$, which is immediately to the right of Column i. The following theorems given by Maki and Peterson [6] are used to develop the necessary test patterns.

Theorem 1 *A stuck-at fault on a pass transistor's input or output can be detected by turning on the transistor (Pi = 1) and changing the pass variable input (Vi).*

Theorem 2 *A transistor stuck-open fault can be detected by letting the control variable be 1 (Pi = 1) and changing pass variable input (Vi).*

Based on these theorems, the complete test sequence for Cell (j, i) is obtained as shown in Table 5 and may be reduced to Table 6. The procedure may be summarized as follows:

1. Set $y_i = 1$, change input I_j between 0 and 1;
2. set $y_i = 0$, change input I_j between 0 and 1.

Certain characteristics of the network can be summarized by the following theorems:

Theorem 3 *A fault in Cell (j, i) will be masked when the transistor controlled by y_k in Cell (j, k) is turned on, for $0 < k < i$.*

Theorem 4 *Each MUX cell in the array is responsible for forcing a state variable to transition from a 1 to a 0 or from a 0 to a 1.*

Test patterns can be generated by identifying present states in which a cell is responsible for forcing a state variable to a 1 or 0. The test set consisting of all such states will constitute a complete test set.

	Fault	Input		Control		Output
		I_j	Q_{i+1}	y_i	$\overline{y}_i$	Q_i
1	I_j s-a-0	1	X	1	0	1
2	I_j s-a-1	0	X	1	0	0
3	Q_i s-a-0	1	X	1	0	1
		X	1	0	1	1
4	Q_i s-a-1	0	X	1	0	0
		X	0	0	1	0
5	trans.	0	X	1	0	0
	y_i s-op	1	X	1	0	1
6	trans.	X	0	0	1	0
	$\overline{y}_i$ s-op	X	1	0	1	1

Table 5: Faults and TPs for MUX Cell in ILA

	I_j	Q_{i+1}	y_i	$\overline{y_i}$	Q_i
1	0	X	1	0	0
2	1	X	1	0	1
3	0	0	0	1	0
4	1	1	0	1	1

Table 6: Compressed Test Set

4.2 Test Procedure

For a synchronous machine having m state variables, y_i, and n inputs, I_j, the ILA network will have n rows and m columns. Each state variable is realized by a regular BTS pass transistor network, as shown in Figure 1 for Y_1. To test Cell i, let $y_1y_2y_3\ldots y_iy_{i+1}\cdots y_n = 000\ldots 1X\ldots X$, which propagates the I_j input of Cell i to the output. Changing I_j then detects the presence of stuck-at and stuck-open faults. Specifically, the path from the input to the output passes through the transistor controlled by y_i and transistors controlled by $\overline{y}_1, \overline{y}_2, \ldots, \overline{y}_{i-1}$. Any stuck-at or stuck-open fault along this path is thus detected. The above steps can be summarized in the following procedure:

1. Turn on the transistor controlled by y_1 in MUX Cell 1; the other transistor in the MUX can be turned off. Toggle input I_j of the transistor controlled by y_1 to test the transistor.

2. Repeat Step 1 for transistors controlled by y_i, $i = 2, 3, \ldots, m$, with the input pattern $y_1y_2y_3\ldots y_i \ldots y_m = 000\ldots 1 \ldots 0$ and toggle input I_j, to detect faults in the transistors controlled by y_i and $\overline{y}_{i-1}$.

The complete test set is identical to that developed using a gate level model (Table 4). This shows that for this class of sequential circuits it is not necessary to use additional tests for detecting transistor stuck-opens. Tests 1 to 12 are patterns for testing the example synchronous sequential circuit with a ZERO cell in each row; Tests 13 and 14 are additional patterns needed for circuits without a ZERO cell in every row. If one row does not have a ZERO cell, the test of the transistor controlled by $\overline{y}_i$ in the rightmost cell can not be accomplished with a 1-hot-code; instead all state variables must be 0 for testing.

When the circuit is in the operational mode, the Q_{i+1} inputs for the MUXes in the rightmost column should be V_{SS}. This input can detect a stuck-at-1 fault but can not test for a stuck-at-0 fault and stuck-open fault. To test these two faults, a mode control signal should be given to switch the circuit to a test mode. In this mode, the Q_{i+1} inputs are assigned to be 1 to test for the stuck-at-0 and stuck-open faults.

4.3 Real Time Testing

A 1-hot-code is used in the state assignment of the sequential circuit, meaning that all present and next state codes shall be a 1-hot-code also. It is important to note that the test patterns are also a 1-hot-code. Assume there is a fault in Cell i for next state variable Y_j. Only when Cell i must force a 1 to 0 or 0 to 1 transition will the fault become detectable. When the fault becomes sensitized to affect the next state value, then the next state will no longer be a 1-hot-code and hence force the circuit to go to another valid state code word. A self-checking 1-out-of-n code checker is needed for real-time fault detection. A totally self-checking check circuit for a 1-out-of-n code can be built on-chip to do built-in-self-test and error indication [7].

5 Conclusions

This paper presents an efficient methodology for generating test patterns for a particular class of synchronous sequential circuits implemented as a BTS pass network ILA. The test pattern generation procedure is discussed and a complete test pattern set for all of the ILA cells is obtained. The proposed method was developed at the transistor level to cover both stuck-at and transistor stuck-open fault conditions, with the advantage of simplicity, ease of test procedure, and efficiency with respect to test generation time. The logic gate level testing using the conventional method of truth table verification is reviewed and compared with the new method. It is shown that the two test sets are equivalent, eliminating the need for additional test patterns to detect transistor stuck-open faults.

References

[1] D. Radhakrishnan, S. Whitaker, and G. Maki, "Formal Design Procedures for Transistor Switching Circuits," IEEE Journal of Solid State Circuits, vol, sc-20, pp. 531-536, April 1985.

[2] M.N. Liu, K.Z. Liu, G.K. Maki, and S.R. Whitaker "Automated ILA Design for Synchronous Sequential Circuits," 3rd NASA Symposium on VLSI Design Oct, 1991, pp.12.1.1-12.1.14.

[3] F.J.O. Dias, "Truth-table Verification of an Iterative Logic Array," IEEE Transactions on Computers, June 1976, pp. 605-613.

[4] Jae-Min Lee and Sung-Mo Kang, "Testable CMOS Design for Robust and Logical Testability of Stuck-open/Stuck-on Faults," Proceeding of the 34th Midwest Symposium on Circuits and Systems, May 623-628.

[5] A.D.Friedman and P.R.Menon *Fault Detection in Digital Circuits*, Prentice-Hall, Inc., 1971.

[6] G. Peterson and G. Maki "Binary Tree Structured Logic Circuit: Design and Fault Detection", International Conference on Computer Design, October, 1984, pp. 671-676.

[7] D. Anderson and G. Metze, "Design of Totally Self-Checking Check Circuits for m-out-of-n Codes," IEEE Transaction on Computers, March 1973, pp.263-269.

Faulty Behavior of Asynchronous Storage Elements [1]

Waleed K. Al-Assadi, Ding Lu
Anura P. Jayasumana, Yashwant K. Malaiya[†], and Carol Q. Tong
Department of Electrical Engineering
[†] Computer Science Department
Colorado State University
Fort Collins, CO 80523

Abstract - **It is often assumed that the faults in storage elements (SEs) can be modeled as output/input stuck-at-faults of the element. They are implicitly considered equivalent to the stuck-at faults in the combinational logic surrounding the SE cells. A more accurate higher level fault model for elementary SEs used in asynchronous circuits is presented. This model offers better representation of the physical failures. It is shown that the stuck-at model may be adequate if only modest fault coverage is desired. The *enhanced* model includes some common fault behaviors of SEs that are not covered by the the stuck-at model. These include *data-feed-through* behaviors that cause the SE to be combinational. Fault models for complex SE cells can be obtained without a significant loss of information about the structure of the circuit.**

1 Introduction

Functional fault modeling is an effective approach to handle the complexities of large digital circuits. A functional fault model hides the complex fault behavior and presents a way of considerably simplifying test generation [1], [2]. Higher level fault models are easier to use because they represent the fault behavior independent of detailed lower level description. However it has been shown in some situations that a simple functional model may not adequately represent a significant fraction of failures. When this is the case, tests based on such a model may not be significantly better than random testing. If the fault model is *adequate*, a functional test set will test for most faults, while at the same time considerably reducing the test generation effort. A fault model can be termed adequate if it explicitly covers (i.e. coverage is guaranteed for) a major fraction, say x%, of all likely faults [2]. The number x cannot be obtained by using any fundamental considerations, but would be based on a reasonable convention. The faults not explicitly covered may or may not be tested if the test vectors are obtained using a fault model. Thus a fault model with low *explicit coverage* is likely to be inadequate.

A good strategy is to obtain a functional fault model for logic blocks derived from the the physical structure of the circuit. This requires that accurate fault models for primitive blocks, such as elementary storage elements (SEs) be considered. Although test considerations at the low level can be computationally complex, an accurate fault model for complex logic blocks inferred from the physical structure of the circuit can reduce the test generation and fault simulation efforts significantly.

[1] This work was supported partly by a SDIO/IST funded project monitored by ONR.

The elementary SEs are the basic primitives in complex logic blocks like registers, finite-state machines, and static memory blocks. This paper examines the major transistor-level faults for two elementary SEs used basically in asynchronous circuits. The behavior of each cell under the above faults is analyzed to evaluate possible functional fault models. Results for elementary SE cells are extended to characterize complex SE cells. In section 2, the minimal stuck-at model and the proposed enhanced model are described. SE cells are examined in section 3 for all possible transistor-level faults to seek a fault model with high fault coverage.

2 Fault Modeling of Elementary SE Cells

The minimal (stuck-at) fault model assumes that internal faults in the SEs can be modeled as stuck-at-0/1 at the inputs or the outputs of the SEs. We examine below the effectiveness of the minimal fault model in representing physical failures. The results reveal the need for a more accurate fault model to better represent the physical failures at the transistor level of an elementary SE.

To examine a SE cell, in general, an input sequence is required rather than a single input vector. Let T=$\{t_1,....,t_n\}$ be the set of all possible input combinations and $R(s,t_i)$ be the response of the cell to the input vector t_i applied to the cell when the cell is at state s. The behavior of each cell under all possible transistor faults is examined for all input combinations and previous states. A multivalued logic representation is used to better represent voltage levels that are not exactly logic 1 (hard 1) or logic 0 (hard 0) [3]. Here *high* level (H) corresponds to both 'hard 1' and 'soft 1', and *low* level (L) corresponds to both 'hard 0' and 'soft 0' [4]. A fault that causes the SE output to be $L(H)$ for all $t_i \in T$, regardless the state of the SE, can be modeled as stuck-at-0/(1). Under some faults the output of the faulty cell cannot make a high to low (low to high) transition, and the corresponding behavior is represented by H$\not\rightarrow$L (L$\not\rightarrow$H). Such faults generally appear as stuck-at-1(stuck-at-0).

However, some faulty behaviors of the SE cell do not manifest as stuck-at-0/1. Such faults cause the SE cell to become *data-feed-through* as defined in [2].

Definition 1: A faulty SE cell is said to be *data-feed-through* when its behavior becomes combinational such that $R(s,t_i) = f(y)$ for each $t_i \in T$, where y is the data part of t_i.
For example, for a NAND pair latch, y is a double element vector corresponding to A and B.

Some recent papers address the detection of several physical failures in CMOS synchronous latch cells [2], [4]-[7]. A comprehensive faults model for such latches is presented in [2]. The proposed *enhanced* fault model is presented. It was observed that *data-feed-through* faults cause a *race-ahead* condition in sequential circuits, i.e. the circuit reaches a state one clock period too early [8]. In this paper, we investigate the *enhanced fault model* for SEs that are used in building asynchronous circuits. The enhanced model includes faults that cause *data-feed-through* and problems of *non-retention* of logic level behaviors as well as the stuck-at faults. Such faults can be detected by monitoring logical levels. Hence they are termed logically testable.

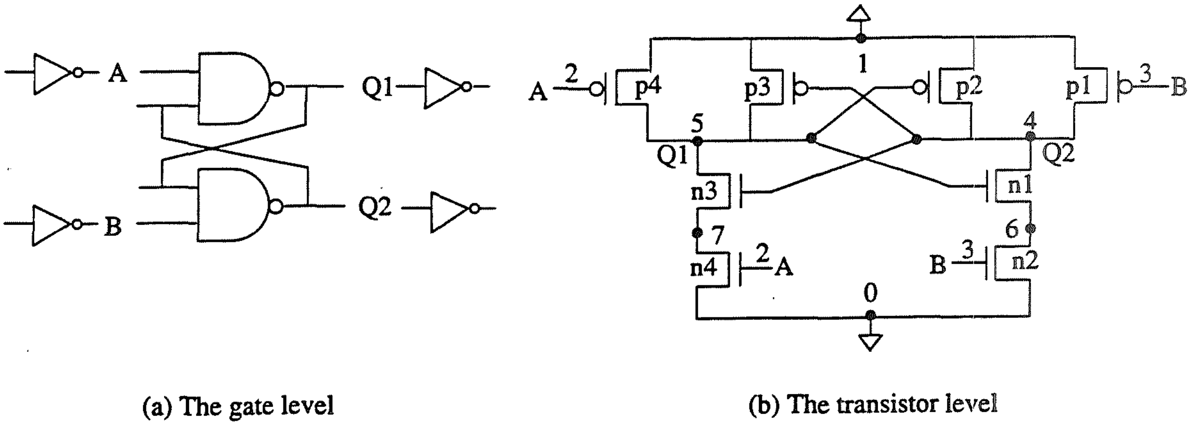
3 Detailed Examinations of the Elementary SE Cells

In this section, a detailed examination of two different elementary asynchronous SEs is presented. Each cell is examined for all possible transistor faults. Results obtained analytically based on a multivalued algebra have been verified by SPICE. A good functional fault model is sought such that the functional behavior of faulty SE cells can be adequately described. Both the *minimal* and the *enhanced* fault models are examined for the effectiveness in representing the functional faults. Because of the transistor sizing and technology used, '0' dominates if two nodes are bridged. All possible bridging faults between nodes in the same well are considered. We use (x, y) to indicate a bridging fault between nodes x and y. Bridging faults between internal nodes of different wells are not included because the probability of having such faults is very small. Analysis assumes that a bridging fault corresponds to a *hard short*. The analysis shows that many stuck-on and bridging faults change the conductance path between V_{dd} and V_{ss} nodes. This suggests that monitoring the supply current (I_{DDQ}), which can be many orders of magnitude higher in the presence of such faults, can be used for testing such faults. In the presence of stuck-open faults, a SE cell could turn from static to dynamic under some input vectors. This means that the logic value of the output of the cell is maintained due to the charge stored in the capacitance associated with the output node. This state may last only for a short time due to the leakage of the charge. However, at normal clock rates such faults can be detected only if they manifest as delay faults. Faulty behavior of two SE cells are summarized next.

3.1 The NAND pair latch

The cell is shown in Figure 1. A single-rail output is considered here and the output is observed at $Q1$. Similar but somewhat more complicated results for double-rail case can be obtained. In fault free cell, vector $AB = 11$ causes both $Q1$ and $Q2$ nodes to retain their previous logical values and the cell remains static, while vector $AB = 00$ is to be avoided because it causes race problem in the cell. However, only vectors $AB = \{01, 10\}$ make the cell in the transparent phase, i.e. can derive the cell to a known logic values.

Table 1 shows the faulty behavior of the cell for stuck-open/on faults. The results show that some of these faults cause enhancement in the supply current (I_{DDQ}) because a direct path between V_{dd} and V_{ss} under some test vectors is established. Consider stuck-open fault in transistors n1 or n2 or both. When vector $AB = 01$ is applied, a direct path between V_{dd} and V_{ss} is formed. The same observation is applied for stuck-open fault in transistor n3 or n4 or in both when vector $AB = 10$ is applied. The results show that only two stuck-open faults show fault free behavior. Consider stuck-open fault in p3. This fault causes the cell to exhibit dynamic behavior when vector $AB = 11$ is applied and the cell is initialized to logic 1. Due to this fault, node $Q1$ cannot keep its logical value of 1 indefinitely since the only way to refresh node $Q1$ if this vector applied is through transistor p3. Stuck-open fault in transistor p2 causes similar change when the same vector applied and the cell initialized to logic 0. These two fault are considered undetectable if tests are applied at normal rate. However, if tests are applied much slower than normal rate, which is unusual in testing, then when vector $AB = 11$ is applied, then the node $Q1$ capacitance can discharge to bring node $Q1$ to logic 0. This method of detection may also be unreliable because the voltage on floating nodes may settle down at an inderminate value rather than at logic 0. Hence these two faults are regarded undetectable. All stuck-on faults cause enhancement in I_{DDQ} compared with the fault free case.



(a) The gate level

(b) The transistor level

Figure 1: The NAND-pair cell

Testing the cell for bridging faults reveal the importance of the multivalued algebra. The results are given in Table 2. The indeterminate value is observed when both nodes $Q1$ and $Q2$ are shorted. This fault can cause charge sharing if vector $AB = 11$ is applied. Such a behavior can only be detected by observing the supply current I_{DDQ}. Bridging fault between nodes 7 and 6 shows stuck-at behavior depending on the initial conditions. Therefore this fault is modeled as stuck-at regardless of the logical values at nodes $Q1$ and $Q2$. Similarly bridging fault between input nodes A and B has consequences assuming 0 dominance. If input vector $AB = 01$ is the initialization vector, then $Q1$ is always at logic high (H) and $Q2 = A \oplus B$, however the observation is reverse-versa if vector $AB = 10$ is applied first. Therefore for the sake of fault coverage, we model this fault as stuck-at.

Tables 1 and 2 shows that only 18 faults out of 38 faults (i.e. 47%) are modeled as stuck-at and 17 faults (i.e. 45%) turn the cell combinational. Therefore the *enhanced* fault model would cover 92% of the logically testable faults. Testing this latch using robust tests is not trivial, because test patterns depend on both combination of primary inputs (A and B) and also on the state variables, which are not directly controllable and dependent on change of primary inputs. Consider stuck-open fault in transistor p4. In general to test for a stuck-open fault, a two -pattern test, the initialization pattern and the test pattern are required. To test for this fault, node Q1 has to be initialized to logic 0 by applying $AQ2 = 11$. $Q2$ however is a function of A and $Q1$ and can be driven to 1 by making either $B = 0$ or $Q1$=0. For this fault, only possible way is to make $B = 0$ always. Therefore to ensure the robustly of the test, a 3-pattern test; $AB = \{10, 11, 01\}$ instead of a two-pattern test is required. This shows the the test is not trivial robust because we have to take into consideration the state variables or outputs of the latch into consideration.

Table 1: Faulty behavior of the NAND pair latch under stuck-open/on faults

Transistor	Stuck-open			Stuck-on		
	$Q1$	$Q2$	Model	$Q1$	$Q2$	Model
p1	H$\not\rightarrow$L	L$\not\rightarrow$H	stuck-at	$\overline{A}$	H$\not\rightarrow$L	data-feed-through
p2	fault-free†	fault-free†	– – – –	$\overline{A}$	H$\not\rightarrow$L	data-feed-through
p3	fault-free†	fault-free†	– – – –	H$\not\rightarrow$L	$\overline{B}$	stuck-at
p4	L$\not\rightarrow$H	H$\not\rightarrow$L	stuck-at	H$\not\rightarrow$L	$\overline{B}$	stuck-at
n1	$\overline{A}$	H$\not\rightarrow$L	data-feed-through	$\overline{B}$	B	data-feed-through
n2	$\overline{A}$	H$\not\rightarrow$L	data-feed-through	H$\not\rightarrow$L	L$\not\rightarrow$H	stuck-at
n3	H$\not\rightarrow$L	$\overline{B}$	stuck-at	$\overline{A}$	A	data-feed-through
n4	H$\not\rightarrow$L	$\overline{B}$	stuck-at	L$\not\rightarrow$H	H$\not\rightarrow$L	stuck-at

†: The cell turns into dynamic under vector $AB = 11$

3.2 The C-element

The C-element shown in Figure 2 is the storage element used in self-timed asynchronous circuits. Such circuits like a pipeline interconnection circuit that controls data transfers between computation blocks, which is basically a half or full handshake circuit [9], [10]. This is a dynamic cell because there is no feedback in the cell and the output is observed at node C. Its logic function can be described by the Boolean equation $C = AB + AC' + BC'$, where C is the present state and C' is the previous state. Hence only two vectors $AB = \{11, 00\}$ make the cell in the transparent phase, while vectors $AB = \{01, 10\}$ make the cell latch its previous value.

The cell is examined to verify the effectiveness of the stuck-at model. It was observed that this model can not cover all the possible physical defects within the element. Actually, most of the defects within the cell like transistor stuck-on, transistor stuck-open and bridging between internal nodes have some other faulty behavior which can not be interpreted by the stuck-at fault model.

The results given in Table 3 show that some faults cause the cell to exhibit a behavior change in the latch phase, while still functioning properly in the transparent phase. *Conditional no-retention* of logic 1/0 ($CNR-1/0$) behaviors are observed. The definition is given below [2]:

Definition 2: Consider a SE cell in the state $Q = 1$ in the transparent phase, the cell exhibits *conditional non-retention* ($CNR - 1$) behavior if the cell fails the latch logic 1 and

Table 2: Faulty behavior of the NAND pair latch under possible bridging faults

Bridging fault	Logical behavior		Model
	$Q1$	$Q2$	
(1,5)	H	$\overline{B}$	stuck-at
(1,3), (6,0), (4,7),	H$\not\rightarrow$L	L$\not\rightarrow$H	
(2,7)	H$\not\rightarrow$L	$\overline{B}$	
(6,7), (2,3)	– – – –	– – – –	
(7,0), (1,2), (5,6)	L$\not\rightarrow$H	H$\not\rightarrow$L	
(2,5)	A	$\overline{A}+\overline{B}$	data-feed-through
(3,0), (3,6)	$\overline{A}$	H$\not\rightarrow$L	
(2,4), (3,7)	$\overline{A}$	A	
(5,7)	$\overline{A}$	A+$\overline{B}$	
(1,4)	$\overline{A}$	H	
(4,6), (3,5), (2,6)	B	$\overline{B}$	
(3,4)	$\overline{A}+\overline{B}$	B	
(4,5)	Indeterminate	Indeterminate	parametric

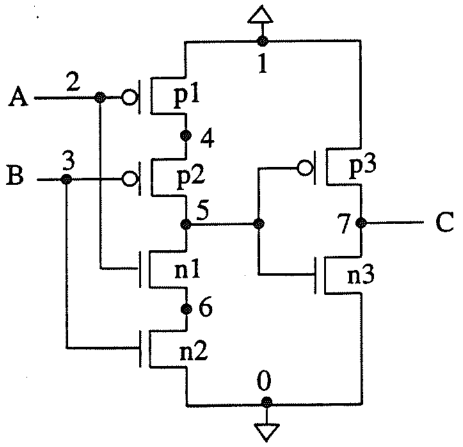


Figure 2: The C-element

instead Q turns to logic 0, i.e. $R(1,t_i) = 0$, where t_i is an input vector such that $t_i \in$ latch phase vectors. $(CNR - 0)$ is defined similarly.
As an example, consider stuck-on fault in transistor p1. If the cell is at logic 1 and vector $AB = 10$ is applied, then the cell is unable to latch logic 1 as in the fault free case, but this vector will turn the cell to logic 0. Therefore the behavior is modeled as $CNR - 1$. The tables shows also that the cell become combinational under several faults, and therefore they are modeled as *data-feed-through.* Some faults cause the cell to be *parametric*, where the logical value of the cell is indeterminate and logical testing cannot be used. Such faults can only be detected by monitoring the supply current (I_{DDQ}), which in the presence of the fault is many orders higher than the fault free current. This enhancement in I_{DDQ} is due to the forming of conducting paths between V_{dd} and V_{ss} under some faults.

Table 3 shows that among 33 possible defects within the C-element, only 17 (i.e. 52%) can be modeled by the stuck-at fault model, while the enhanced model covers 13 faults more (i.e. 91%). This means that 100% of the logically testable faults are covered by the enhanced fault model, while only 3 faults can be tested by monitoring the supply current I_{DDQ}..

Table 3: Faulty behavior of the C-element

Fault		**logical Behavior**	**Model**
Bridging:	(3,4), (1,7), (5,0)	stuck-at-1	stuck-at
Bridging:	(2,6), (7,0)	stuck-at-0	
Stuck-open: Bridging:	p1, p2, n3 (1,2), (1,3), (2,4)	H↛L	
Stuck-open: Bridging:	p3, n1, n2 (2,0), (3,0), (3,6)	L↛H	
Stuck-on: Bridging:	p1, p2 (1,4), (4,5)	CNR-1	CNR
Stuck-on: Bridging:	n1, n2 (5,6), (6,0)	CNR-0	
Bridging:	(2,3)	$C = AB$	data-feed-through
Bridging:	(2,5)	$C = \overline{A}$	
Bridging:	(3,5)	$C = \overline{B}$	
Bridging:	(2,7)	$C = A$	
Bridging:	(3,7)	$C = B$	
Stuck-on: Bridging:	p3, n3 (5,7)	indeterminate	parametric

4 Conclusion

The effectiveness of the minimal fault model for two basic SEs used in asynchronous circuits is evaluated. The enhanced fault model for the two cells is proposed, which provides higher explicit fault coverage compared to the minimal fault model. Higher level functional fault models for complex circuits using the two cells considered in this paper as primitives can be inferred from the proposed model, with higher fault coverage. This allows the testing of low level failures that cannot be characterized as stuck-at-0/1 at the functional level without the need to consider the physical implementation of the circuit. Thus the advantages of functional testing is retained with a higher coverage of low level failures. Test generation could be based on the change in the state-transition graph of the complex circuit due to such faults. This can be used to enhance the existing testing techniques for self-timed and asynchronous sequential circuits based on the changes in state transition graph which at present only consider stuck-at faults only.

References

[1] W. K. Al-Assadi, Y. K. Malaiya and A. P. Jayasumana, " Use of Storage Elements as Primitives for Modeling Faults in Sequential Circuits," *Proc. 6th. Int. Conf. on VLSI Design*, pp. 118-123, January 1993.

[2] W. K. Al-Assadi, Y. K. Malaiya and A. P. Jayasumana, "Faulty Behavior of Storage Elements and its Effects on Sequential Circuits," To Appear in *IEEE Transaction on VLSI Systems.*

[3] P. Banerjee and J. A. Abraham, "A Multivalued Algebra for Modeling Physical Failures in MOS VLSI Circuits," *IEEE Transaction on Computer-Aided Design,* vol. CAD-4, no. 3, pp. 312-321, July 1985.

[4] M. K. Reddy and S. M. Reddy, "Detecting FET Stuck-Open Faults in CMOS Latches and Flipflops," *IEEE Design and Test,* pp. 17-26, October 1986.

[5] D. L. Liu and E. J. McCluskey, "A CMOS Cell Library Design for Testability," *VLSI Systems Design,* pp. 58-65, May 4, 1987.

[6] R. Anglada and R. Rubio, "Functional Fault Models for Sequential Circuits," *Research Report DEE-3*, Electronic Engineering Department, Polytechnical University of Catalunya, Barcelona 1987.

[7] K. J. Lee and M. A. Breuer, "A Universal Test Sequence or CMOS Scan Registers," *Proc. IEEE Custom Integrated Circuits Conf.* pp. 28.5.1-28.5.4., 1990.

[8] W. K. Al-Assadi, Y. K. Malaiya, and A. P. Jayasumana, " Detection of Feed-Through Faults in CMOS Storage Elements," *Proc. NASA Symposium on VLSI Design*, pp. 7.2.1-7.2.5, October 1992.

[9] T. H.-Y. Meng, R. W. Brodersen and D. G. Messerschmitt, "Automatic synthesis of Asynchronous Circuits from High-Level Specifications", *IEEE Trans. Computer-Aided Design*, vol.8, pp.1185-1205, 1989.

[10] P. A. Beerel and T. H.-Y. Meng, "Semi-Modularity and Self-Diagnostic Asynchronous Control Circuits", *Advanced Research in VLSI* (MIT press 1991), pp.118-132.

Testing Self-Timed Circuits Using Scan Paths

Ajay Khoche and Erik Brunvand
Computer Science Department, University of Utah
Salt Lake City, UT 84112
E-mail: akhoche@cs.utah.edu, brunvand@cs.utah.edu

Abstract - **Testing is an essential part of any digital system design and one that can be made much easier if testing is considered during the design process rather than after the system is complete. There are a number of techniques for integrating testability into system design, but many of these traditional testing methods used for synchronous circuits are not directly applicable to non-clocked asynchronous circuits. As a result, many asynchronous circuits do not employ design for testability techniques. In this paper we present a method of using the familiar model of scan paths modified to test self-timed systems. Specifically, we consider circuits designed using a library of self-timed modules to assemble systems with two-phase transition control and bundled data paths. The method involves modifying these self-timed modules such that circuits designed from them have a built-in scan path.**

1 Introduction

As VLSI circuits become larger, faster, and more dense, the difficulties involved in building globally clocked synchronous systems also grow dramatically. One alternative is to use *asynchronous* or *self-timed* circuits which avoid timing problems by enforcing simple communication protocols between parts of the circuit. These protocols are insensitive to delays in circuit components or the wires that connect them. As building blocks for larger hardware systems, asynchronous and self-timed systems that synchronize locally and use no global clock have a number of compelling advantages over their synchronous counterparts [16, 17]. In particular, asynchronous and self-timed systems are free from clock-related timing problems, allow simpler system composition, show increased robustness in the face of process and environmental variation, can exhibit much lower power consumption, and even show increased performance when compared to globally synchronous systems.

Because of these properties, asynchronous circuits in general, and self-timed circuit in particular, are experiencing renewed interest by systems designers [3, 13, 14, 17]. One area where asynchronous circuits lag behind synchronous circuits, however, is in testing. Testing is a very important step in the development of reliable systems in general. When those systems are built using custom integrated circuits (ICs), testing for fabrication faults becomes vital. In order for such custom ICs to be tested effectively, testing must be integrated into the design of the circuits and not left as an afterthought. There is a large body of work related to testing clocked synchronous circuits. Unfortunately many of these techniques are not directly applicable to asynchronous and self-timed circuits and systems. Testing is also a computationally intensive activity, and the inherent concurrency typical of asynchronous designs makes it even worse. As a result, many of these systems are not tested rigorously, and have acquired an undeserved reputation of being difficult to test.

Traditionally, research into testing asynchronous circuits has focussed on testing networks of gates operating as asynchronous logic [15]. Recently, with the development of new methodologies for designing asynchronous circuits, these forms of testing are being revised and extended [9, 10, 11, 12]. In this paper we propose a method of testing a class of self-timed circuits for stuck-at faults under a single stuck-at fault assumption using the familiar model of scan paths. Our particular design style for self-timed circuits is based on a library of circuit modules used to build systems where control is based on handshakes between local modules. Our testing method consists of augmenting the self-timed circuit modules used in the design so that the resulting circuit will have a built-in scan path for testing.

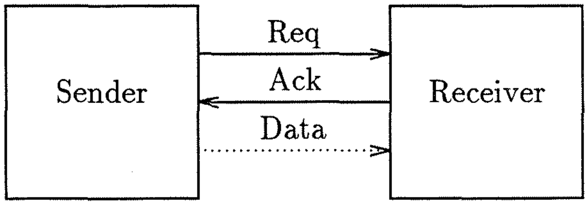


Figure 1: A Bundled Data Interface

Currently, our focus is on testing the control path only. However this approach can be easily extended to include the datapath in the scan chain. Although the cell library is available for field programmable gate arrays (FPGAs), CMOS, and GaAs implementation [6], we are evaluating the modified library using FPGAs for prototyping. As we gain more experience with the modified cells we plan to modify the other libraries as well.

In the following sections we will present an overview of our self-timed signaling style, present the library of cells, and show how they are used to assemble systems. We then describe our test strategy and show an example of how a module can be modified to include a scan path. We conclude with a discussion of overheads for the testable modules under current implementation.

2 Self-Timed Circuits

Self-timed circuits are a subset of a broad class of asynchronous circuits. Self-timed circuits are composed of parts which communicate using self-timed protocols. Self-timed protocols are often defined in terms of a pair of signals, one that requests or initiates an action, and another to acknowledge that the requested action has been completed. One module, the sender, sends a request event (*Req*) to another module, the receiver. Once the receiver has completed the requested action, it sends an acknowledge event (*Ack*) back to the sender to complete the transaction.

Although self-timed circuits can be designed to implement their communication protocols in a variety of ways, the circuits used in our library use two-phase transition signaling for control and a bundled protocol for data paths. Two-phase transition signaling [3, 16] uses transitions on signal wires to communicate the *Req* and *Ack* events described previously.

Only the transitions are meaningful; a transition from low to high is the same as a transition from high to low and the particular state, high or low, of each wire is not important.

A bundled data path uses a single set of control wires to indicate the validity of a *bundle* of data wires [17]. This requires that the data bundle and the control wires be constructed such that the value on the data bundle is stable at the receiver before a signal appears on the control wire. This condition is similar to, but weaker than, the equipotential constraint [16]. Two modules connected with a bundled data path are shown in Figure 1.

3 A Self-Timed Cell Library

Our design method builds control circuits using a variety of modules which communicate using two-phase transition signals. These modules, described in more detail elsewhere [2, 3, 17], are shown symbolically in Figure 2. Other modules in the library, such as transition-controlled latches, and completion-sensing adders, are used to build self-timed data paths. The functionality of the control modules is as follows:

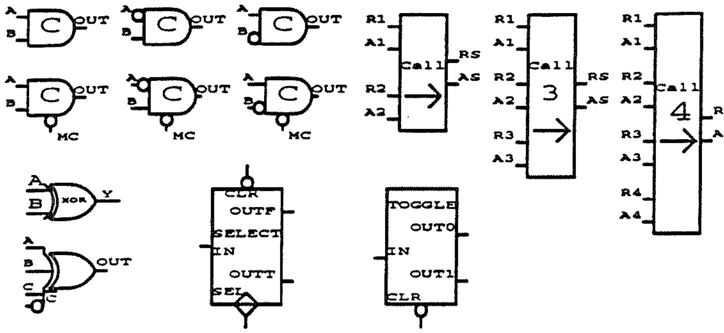


Figure 2: Control Modules for Self-Timed Designs

XOR An XOR behaves as an OR for transition signals. When a transition occurs on any of its inputs, the XOR generates a transition at its output.

C-Element A C-element is used as an AND function for transitions. A transition occurs at the output only when there have been transitions at both of its inputs. Note that the C-element must start in a state where both inputs are at the same value to behave in this way. A master clear signal to the control modules ensures this condition on system reset.

Select A two-way transition Select module steers the input transition to one of two outputs depending on the value of its select signal. The *sel* signal should be valid before the input transition arrives and must remain valid until after an output transition is generated at one of the outputs. In other words, *sel* is bundled with respect to the input transition.

Toggle A Toggle module routes an input transition alternately to its outputs. After initialization the first input transition causes a transition on *out0* and subsequent transitions are routed to alternate outputs.

Call A Call module acts as a hardware subroutine call allowing multiple access to a shared resource. The Call module routes the *Req* signal from a client (for example, either *R1* or *R2* in a two-way Call) to the subroutine circuit, and after the subroutine acknowledges, routes the *Ack* back to the appropriate client. The requests must be mutually exclusive.

This module library has been used in an OCCAM based automatic circuit compilation system [3, 7]. The software constructs of the OCCAM have been implemented using these library components. Thus the programs written in OCCAM are automatically translated into self-timed circuits using this library by this system. This library has also been used to build large circuits by hand using commercial schematic capture system from Viewlogic [4, 5].

4 Testing Strategy

Testing digital circuits usually consists of enabling a mode where the internal signals in the circuit may be set to known values and the internal results checked to see if they are consistent with those input values. In a synchronous circuit this is often achieved by using special latches for the state registers that can shift their contents in and out serially [1]. The various state holding latches of the system are connected into a serial scan chain which is used to apply test vectors and return test results. Our strategy is to adapt this testing style for use with self-timed circuits built using our library cells.

Some of the problems associated with testing self-timed circuits include:

- In the absence of a global clock the storage elements in self-timed circuits operate on local gating signals and a change at one place can propagate through many storage elements. This makes the testing difficult. In contrast, in synchronous circuits where the storage elements are controlled by global clock, a change in a signal can only propagate to the next storage elements.

- In a synchronous system it is possible to slow down the operation of the entire system simply by decreasing the clock speed to reason about noise related problems. Self-timed circuits react to local handshake signals and there is no analogous technique for slowing system operation.

- Using our library module approach, control is distributed throughout the system and is not centralized in a single controller with a convenient state register for the scan path. Each self-timed module is in fact a tiny state machine in itself. This increases the complexity of testing if functional testing of the entire circuit is desired.

4.1 Scan Design: General Approach

We propose adapting the scan design approach to address the above mentioned problems. Although control is distributed, the scan path connects the control modules in a way that allows them to be tested. In fact, the testing problem actually becomes somewhat easier because each control module, which is in itself a very small state machine, may be tested individually rather than testing the entire control path at once. With some additional extensions to the scan path the components can be partitioned and each partition can be tested for full functionality in isolation from other partitions.

A similar technique has been used by Carson [8] to test the special case of a self-timed counter. In that case, the Toggle module used to build the counter was modified to include a scan path in a similar way. We generalize this technique to include all the control modules in our self-timed library. The design of the scan chain has been organized at two levels:

Module level First, each module has been modified to incorporate a piece of the scan chain. This is accomplished by modifying the storage elements already used in the modules, and by adding some multiplexing of inputs to those storage elements. An example of this type of modification is shown in Figure 3. The extended modules operate in four modes controlled by two signals: *ns* (normal or scan) and *nx* (normal or external). In scan mode the storage elements of the module are arranged to become part of a 2-phase master-slave flip-flop with input from the module's request line and output on one of the module's acknowledge lines. The *nx* signal controls the gating (clocking) of this new master slave flip-flop. In normal mode the normal gating signal takes control of the gating of latches in the module. In external mode the two stages of master slave flip-flop formed out of two latches are gated by two external test clocks called *p1* and *p2*. Using a 2-phase flip-flop allows one to control the gating of each stage of flip-flop independently.

Circuit Level While in scan mode the module essentially becomes a master-slave flip-flop, so when a circuit is composed of such modules a scan path is automatically formed from the request line of the circuits to the acknowledge line. However due to the structure of certain modules, and the logic that may be embedded between these modules, it may not be possible to form scan chain on request/acknowledge lines. In such cases an alternative scan input is provided to the modules such that a single scan chain is formed. This means that the scan path lies along request/acknowledge line of the module as far as possible this allows to reduce routing overhead whenever possible.

4.2 Test Procedure

The testing is performed by shifting different patterns into the circuit using scan register. The circuits is put in the scan mode by setting $ns{=}nx{=}1$. The scan register is clocked by test clocks *p1* and *p2*. Three types of shifting are possible in this shift register. Clocking *p1* once, copies input of the module into one of the storage cell in the module. Clocking *p2* once causes the the above value to be copied into other storage cell. A full shift i.e. *p1* followed by *p2* causes the input to be stored in both the storage cells, thus in a full shift the storage cells in the module act as a master-slave flip-flop. The steps involved in testing a circuit composed of scan-modified modules are described below.

Scan Register Testing First the scan register itself must be tested. In this step the circuit is put into scan mode with external clocking by setting $ns{=}nx{=}1$. Once in this mode, 0's are shifted in and then out of the scan chain. This tests for stuck-at-1 faults on the portion of the circuit which is part of the scan path. After this 1's are shifted in and out to check for stuck-at-0 faults in the parts of the circuit that make up the scan chain.

Fault Testing In this step we test for stuck-at faults in the circuit components which are not part of the scan path. In our modules the storage elements (that are now connected into a scan chain) generally lie at the output of the module so setting these to a particular value results in applying that value to the combinational part of the following module. These combinational parts are tested as follows:

- Set *ns*=*nx*=1 and shift the test vector in.
- Set *ns*=0 and let the module operate in normal mode except that since *nx*=1, the storage elements are still gated(clocked) from the external *p1* and *p2* signals.
- After the internal signals have had time to stabilize, enable storage elements to latch new values using test clocks *p1* and *p2*.
- Set *ns*=1 and shift the new latched values out.

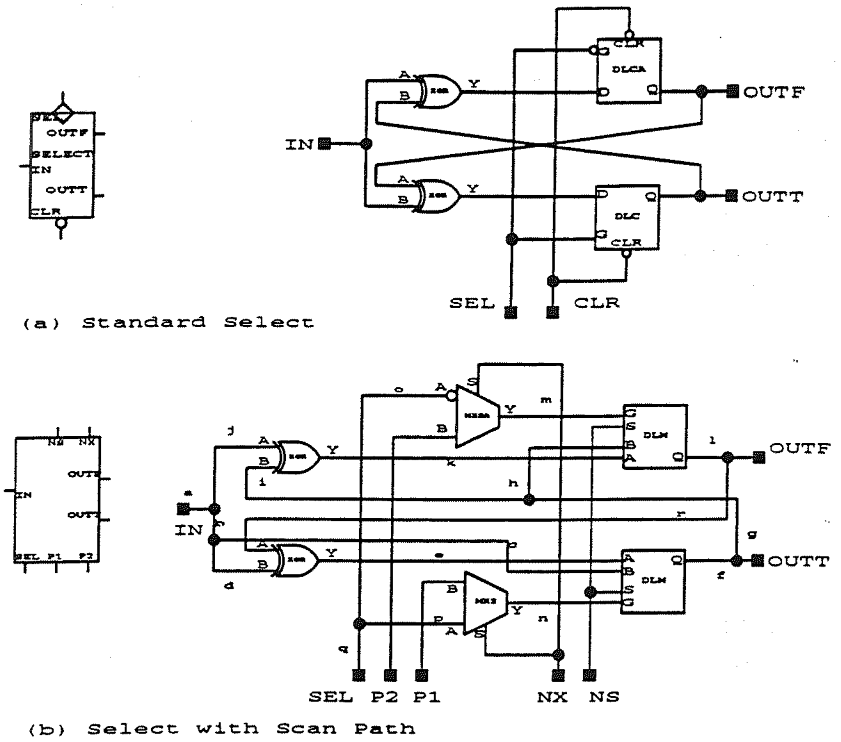
Testing the Gating Network In both of the above steps the gating network used in the normal operation of the module was not active and thus remained untested. Testing of the gating network is, in general, difficult because if the gating network is enabled then we loose our test-mode control over the storage elements. Any signal change on an internal storage element of the module can potentially propagate through several storage elements, which was not the case when external clocking was in effect. So, to test the gating network a special strategy is required. In order to test a module we have to make sure that inputs to it don't change when normal gating is enabled (*nx*=0). This can be done by shifting proper values such that the modules which provide input to it are in stable state.

- Set *ns*=*nx*=1 and shift in the values such that the modules feeding input to module under test are in a stable state. The module under test is kept in a state that fault in the gating network can be detected.
- Set *nx*=1 to enable the normal gating networks. Now since the modules which are providing input in a stable state, no change will occur in the input. However the module under test may undergo a change depending on the test vector.
- After the circuits have stabilized, set *nx*=1 to enable the external clocking and shift the results out. Now even if the change in output of module under test has propagated to following modules, it is of no significance as we are interested only in the values of the outputs of it, which can observed by shifted out.

5 Module Design

5.1 Modifications

The way in which the modules have been been modified to implement this testing scheme is described using a Selector as an example. The basic circuit is shown in Figure 3(a) and the



(a) Standard Select

(b) Select with Scan Path

Figure 3: Selector and Scan-Modified Selector

modified circuit is shown in Figure 3(b). These modifications are described below.

- The single-input latches (e.g. DLC, DLCA) used in (a) have been replaced with multiplexor latches(DLM) with two inputs in (b). These latches are part of the Actel FPGA component library and are constructed from a single Actel basic module. Selection between inputs is controlled by a global signal called *ns*. When *ns*=0 the input to the latches comes from the output of the XOR gate as in the original circuit. When *ns*=1 the input to the lower latch comes from the Select's *in* signal while input to the upper latch is the output of lower latch. Thus when *ns*=1, the latches are connected in series to form a scan chain.

- The second modification is made to the gating signals of the latches. In the modified circuit the gating signal comes from a multiplexor controlled by the *nx* signal. When *nx*=0, the gating signal comes from the *sel* line as in the original circuit, but when *nx*=1 the gating signals are provided from test clocks *p1* and *p2* such that the two latches in scan mode form a 2-phase master-slave flip-flop.

5.2 Testing Strategy

As part of the testing circuits built using scan-enhanced modules, each module may be tested for stuck-at faults. A detailed description of the test procedure for the Select module is given below:

7.5.8

- First the nets on the scan path are tested for stuck-at-1 fault by shifting in 0's and then reading back the values in the scan register. In this stage nets *a,b,c,f,g,h,* and *l* are tested for stuck-at-1. Similarly these nets can be tested for stuck-at-1 by shifting in 1's and then reading back. If any net on scan register has a stuck-at fault the shifted out values will not be same as one shifted in. This step also tests nets *m* and *n* for stuck-at-0 because if any of these lines is stuck-at zero then the corresponding latch will not store a new value and the chain will be broken at that module. In order to test *m* and *n* for stuck-at-1, an alternating pattern of 0's and 1's is shifted in and read out. If one of these nets is stuck-at-1 in any module then the output pattern will not be same as there will be a feed-through in the latch whose gating line is stuck-at-1.

- To test the network consisting of the XOR's, test vectors are applied by putting the circuit in the scan mode with external clocking by setting *ns*=*nx*=1. The circuit is then returned to non-scan mode by setting *ns*=0, allowing the input to the latches to come from the XOR nets. These values are latched using test clocks *p1* and *p2* which can be scanned out by putting the circuit back into scan mode. The following patterns are used to test the XORs:

  - *in*=1, *l*=0 and *f*=1 will test *a,b,d,j,i,e* and *f* for stuck-at-0 while *k* and *r* are tested for stuck-at-1.
  - *in*=0, *l*=1 and *f*=0 will test *a,b,d,j,i* and *f* for stuck-at-1 and *r* and *l* for stuck-at-0.
  - *in*=0, *l*=1 and *f*=1 will test *k* for stuck-at-0.
  - *in*=1, *l*=1 and *f*=1 will test *e* for stuck-at-1.

- The gating network consisting of nets *o, p* and *q* are tested in the following manner:

  - The circuit is first put into scan mode with external clocking by setting *ns*=*nx*=1. A test vector is scanned in such that *in*=0, *f*=1, *l*=1, and *sel*=1. After Shifting this vector, *nx* is set to 0 to allow *sel* to provide the gating signal to the latches. Now in the normal case (i.e. in the absence of any fault), the state of *l* and *f* will become *lf*=10, but if there is a stuck-at-0 fault at *p* or *q* then the lower latch will not be enabled and the state will remain the same. If *o* is stuck-at-0 then the state will become 00. These new states can be scanned out after setting *nx* to 1.
  - To test *p* or *q* stuck-at-1, a test vector is scanned in such that *in*=0, *f*=*l*=1 and *sel*=0. Now *nx* is set to 0. In the normal case the state of *l* and *f* will remain 11 but in case of *p* or *q* stuck-at-1, the state will become 00 as there will be a feed-through from *in* to *l*.
  - The net *o* is tested for stuck-at-1 by setting *in*=*f*=0, *l*=1 and *sel*=0 and then setting *nx*=0. In the normal case the values of *l* and *f* will be *l*=*f*=0, but in case of *o* stuck-at-1 the values will not change.

	Number of Actel Modules	
Library Module	**Non-Scan Version**	**Scan Version**
C-Element	2	3
2-way Call	5	7
3-way Call	10	14
Select	4	6
Toggle	2	4

Figure 4: Library Element FPGA Implementations

5.3 Circuit Overhead

As with any scheme for built-in testing, the scan patch technique adds overhead to circuits both in terms of area and in performance. The extra circuitry for multiplexing inputs to the state holding elements in the modules increases their area and degrades their performance. However, in a custom technology such as CMOS or GaAs, this overhead is usually small. For the FPGA version of the library, the area overhead is more important. This is because of the relatively coarse basic building block of the FPGA compared to the custom technologies. An additional multiplexor consumes an additional logic module on the Actel FPGA. A table of the number of Actel modules used for the control modules in the self-timed library is shown in Figure 4.

Performance effects due to the extra logic, on the other hand, appear to be negligible compared to the variation introduced by the placement and routing of the design on the FPGA. Custom technologies such as CMOS will allow a more realistic estimate of the effect of the extra scan logic on performance. The Actel FPGA is based on anti-fuse links, and each anti-fuse contributes a noticeable resistance to the wire that it forms ($\approx 500\Omega$). A placement that introduces more anti-fuses in a wire can easily be slower than a circuit with deeper logic but shorter connections. We have found that circuits built with the scan-path modules can be either slower or faster than their non-scan versions depending on the automatic placement and routing.

6 Conclusions

Self-timed circuits have a number of compelling advantages over their traditional synchronous counterparts. One area where asynchronous circuits lag behind, however, is in testing. As a result, many self-timed circuits are not tested rigorously, and have gained an undeserved reputation of being difficult to test. Although many testing techniques used for synchronous circuits do not map directly to self-timed circuits, simple extensions may make them more suitable.

We have described a technique for using the familiar notion of a scan-path to test self-timed circuits for stuck-at faults under a single stuck-at fault assumption. The circuits we consider are part of a library of self-timed modules used to build circuits that operate using two-phase transition control signals and bundled data paths. Although the library of modules has implementations in FPGAs, CMOS, and GaAs, we consider only the FPGA version initially. This allows rapid and inexpensive prototyping that would not be possible using other technologies. As we gain more experience with the technique, we plan to extend in a similar way the modules in the other libraries as well.

References

[1] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design.* Computer Science Press, 1990.

[2] Erik Brunvand. A cell set for self-timed design using actel FPGAs. Technical Report UUCS–91–013, University of Utah, 1991.

[3] Erik Brunvand. *Translating Concurrent Communicating Programs into Asynchronous Circuits.* PhD thesis, Carnegie Mellon University, 1991. Available as Technical Report CMU-CS-91-198.

[4] Erik Brunvand. Using FPGAs to prototype a self-timed computer. In *International Workshop on Field Programmable Logic and Applications*, Vienna University of Technology, September 1992.

[5] Erik Brunvand. The NSR processor. In *Proceedings of the 26th International Conference on System Sciences*, Maui, Hawaii, January 1993.

[6] Erik Brunvand, Nick Michell, and Kent Smith. A comparison of self-timed design using FPGA, CMOS, and GaAs technologies. In *International Conference on Computer Design*, Cambridge, Mass., October 1992.

[7] Erik Brunvand and Mike Starkey. An integrated environment for the design and simulation of self-timed systems. In *VLSI-91.* IFIP, August 1991.

[8] Gerald Carson and Getano Borriello. A testable asynchronous counter. *IEEE Journal of Solid State Circuits*, 25(4):952–960, Aug 1990.

[9] Ilana David, Ran Ginosar, and Michael Yoeli. Self-timed is self diagnostic. Technical report, Technion, 1990.

[10] Dong Sam Ha and Sudhakar M Reddy. On testable self-timed logic circuits. In *International Conference on Computer Design*, Cambridge, Mass., October 1984.

[11] Pieter Hazewindus. *Testing Delay-Insensitive Circuits.* PhD thesis, California Institute of Technology, 1991.

[12] Michiel Kamp. Testing delay-insensitive circuits: A case study. Technical report, Eindhoven Institute of Technology, September 1990.

[13] Alain J. Martin. Compiling communicating processes into delay insensitive circuits. *Distributed Computing*, 1(3), 1986.

[14] Cees Niessen, C.H. (Kees) van Berkel, Martin Rem, and Ronald W.J.J. Saeijs. VLSI programming and silicon compilation; a novel approach from Philips research. In *ICCD*, Rye Brook, NY, October 1988.

[15] Gianfranco R. Putzolu and J. Paul Roth. A heuristic algorithm for testing of asynchronous circuits. *IEEE Transactions on Computers*, c-20(6):639–647, June 1971.

[16] C. L. Seitz. System timing. In *Mead and Conway, Introduction to VLSI Systems*, chapter 7. Addison-Wesley, 1980.

[17] Ivan Sutherland. Micropipelines. *CACM*, 32(6), 1989.

Detection of CMOS Bridging Faults Using Minimal Stuck-At Fault Test Sets[†]

Nabeel Ijaz and James F. Frenzel
Department of Electrical Engineering
University of Idaho, Moscow, ID 83844-1023
Office: 208-885-6554 E-mail: j.frenzel@ieee.org

Abstract - **This paper evaluates the performance of minimal stuck-at fault test sets at detecting bridging faults. New functional models of circuit primitives are presented which allow accurate representation of bridging faults under switch-level simulation. The effectiveness of the patterns is evaluated using both voltage and current testing.**

1 Introduction

The diagnosis of digital circuits is an important problem which has attracted great interest in recent years. Most of the work has focused upon the detection of stuck-at faults and many algorithms have been developed that efficiently reduce the test generation and application time. However, the discovery of bridging faults as a major mode of failure [14] has added to the cost of testing VLSI circuits by requiring additional test patterns. As the number of potential bridging faults in a circuit can be quite large, it becomes computationally infeasible to generate tests for each one of them. Recently, work has been done on determining algorithms to generate test sets for detecting bridging faults.

Initial efforts focused on evaluating stuck-at fault tests for detecting bridging faults [1, 10, 11, 5]. Kodandapani and Pradhan showed that there are undetectable bridging faults even when a test set can detect all of the stuck-at faults in the circuits [8]. Recently, quiescent power supply current testing has gained popularity as an effective means of detecting bridging faults. Since bridging faults create a potential path between V_{dd} and V_{ss}, such faults may be detected by monitoring the power supply current [6]. In [15], Storey and Maly compared the effectiveness of stuck-at fault tests to that of current test vectors generated explicitly to detect bridging faults.

All of the efforts mentioned used stuck-at fault test sets with voltage or logic testing, i.e., detecting the fault by monitoring the logic levels at primary outputs. However, with the development of current testing one aspect remained untouched, namely the effectiveness of minimal stuck-at fault test sets as current test vectors; this issue is the focus of this paper. The coverage of bridging faults by stuck-at fault test sets using current testing is compared with voltage/logic testing of bridging faults.

[†]This research was supported in part by NASA under Space Engineering Research Center Grant NAGW-3293.

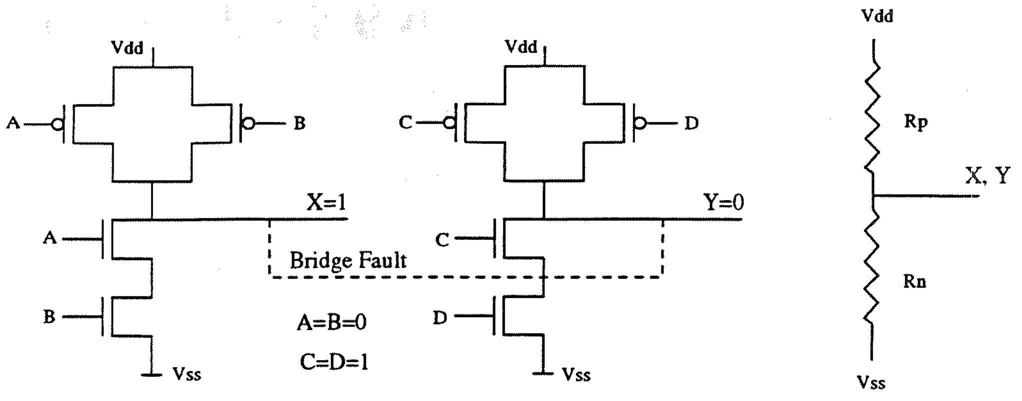


Figure 1: A two-NAND bridging and its equivalent resistance model.

2 The Bridge Fault Model

The behavior of the bridging fault is modeled using the voting model in [2]. In CMOS circuitry, the state of the inputs to a logic gate determines the drive strength of the gate. For example, if both inputs to a NAND gate are at logic 0, then because of the parallel configuration of the p-transistors, the drive strength of the pull-up network will be more compared to when only one input is at logic 0.

When two nodes are shorted together and are trying to drive opposite values, the situation can be approximated by a voltage divider as shown in Figure 1. Although CMOS transistors are non-linear devices, the resistor network representation can accurately model the drive strengths of the transistors involved in the bridging fault. The resultant signal value is determined by which resistor network has the least resistance. For the case shown in Figure 1, SPICE simulations indicated that when both p-transistors are ON, the pull-up network sourced more current than the two n-transistors could sink. The pull-up network thus wins the 'vote' and determines the logic value at the shorted nodes. On the other hand if only one p-transistor is ON and the two n-transistors are conducting, the pull-up network loses the vote and the n-network decides the logic value. Under both situations, the resultant node voltage is between 0 and 5 volts but typically within the ranges interpreted as valid logic signals. We will refer to the case of multiple p-transistors conducting in parallel as a "strong 1", and the case of a single p-transistor conducting as a "weak 1". If the relative strengths of the pull-up and the pull-down networks are very close, this voltage could be in the indeterminate range. However, work done by [2] have shown that bridging faults in CMOS circuits typically results in recognizable logic values.

3 Detection of Bridging Faults Using Voltage Testing

Voltage testing is the conventional way of detecting faults in a circuit. The detection of a bridging fault with this method requires that the test vector activate the fault by placing opposite logic values on the bridged nodes and then propagate the dominated node to a primary output. Figure 2 shows a NAND circuit with nodes I_0 and I_3 bridged together. This is defined as a *Non-Feedback Bridging Fault (NFBF)* since the two nodes are not connected

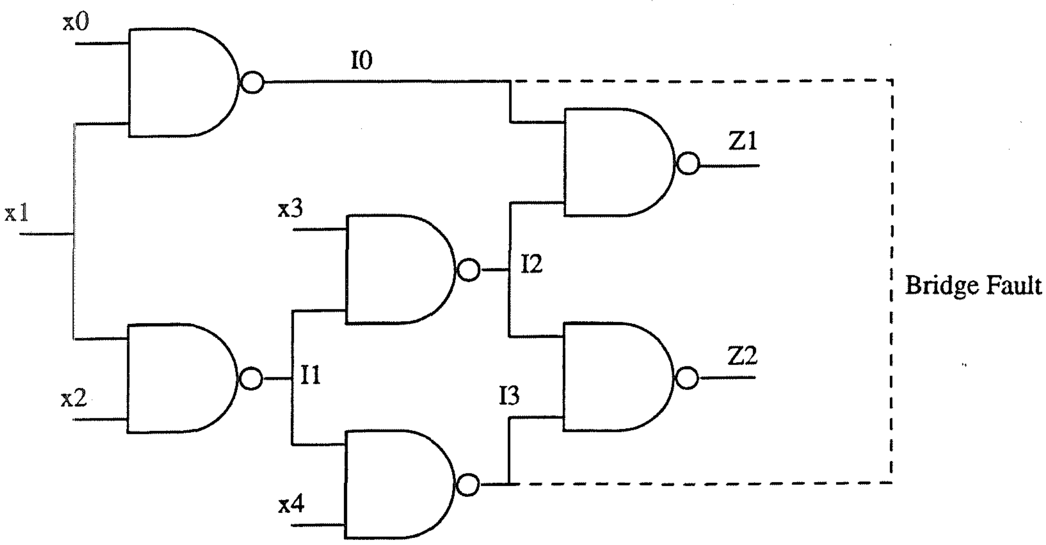


Figure 2: A Non-feedback bridging fault.

by any path through the circuit. In the absence of any information on the strengths of the gates, there are potentially four ways of activating the fault:

1. placing a strong 0 on I_0 and a weak 1 on I_3,
2. placing a weak 0 on I_0 and a strong 1 on I_3,
3. placing a strong 1 on I_0 and a weak 0 on I_3, or
4. placing a weak 1 on I_0 and a strong 0 on I_3.

A test vector that achieves any one of these conditions activates the fault. For the bridging fault to be detected the test vector must also propagate the weaker node to a primary output. For the circuit shown in Figure 2, the complete minimal stuck-at fault test vector set is:

$$<x_0x_1x_2x_3x_4>=<00100>,<01111>,<01010>,<11000>,<10101>$$

The test vector $<11000>$ places a strong 0 on I_0 and a weak 1 on I_3 by placing logic 1 on I_1, thus activating the fault. Furthermore, since x_3 is 0, I_2 is driven to a 1 which sensitizes Z_2 to I_3. The strong 0 on I_0 overrides the weaker 1 on I_3 and causes the output Z_2 to go high. The fault-free output for this vector is logic 0 and hence the fault is detected. Figure 3 shows examples of *Feedback Bridging Faults (FBF)*. Under a FBF, the two bridged nodes lie on the same path in the circuit. The number of inversions between the back node and the front node characterizes the FBF as 'odd' or 'even'[1]. In Figure 3(a), there is an odd number of inversions between the back node 'b' and the front node 'f', characterizing the fault as an odd-FBF; similarly, Figure 3(b) shows an even-FBF. The fault shown in Figure 3(c) cannot be characterized as either even or odd. This occurs in reconvergent fanout circuits when the reconverging paths have different inversion parities.

[1] The term 'front' node refers to the node furthest from the primary inputs.

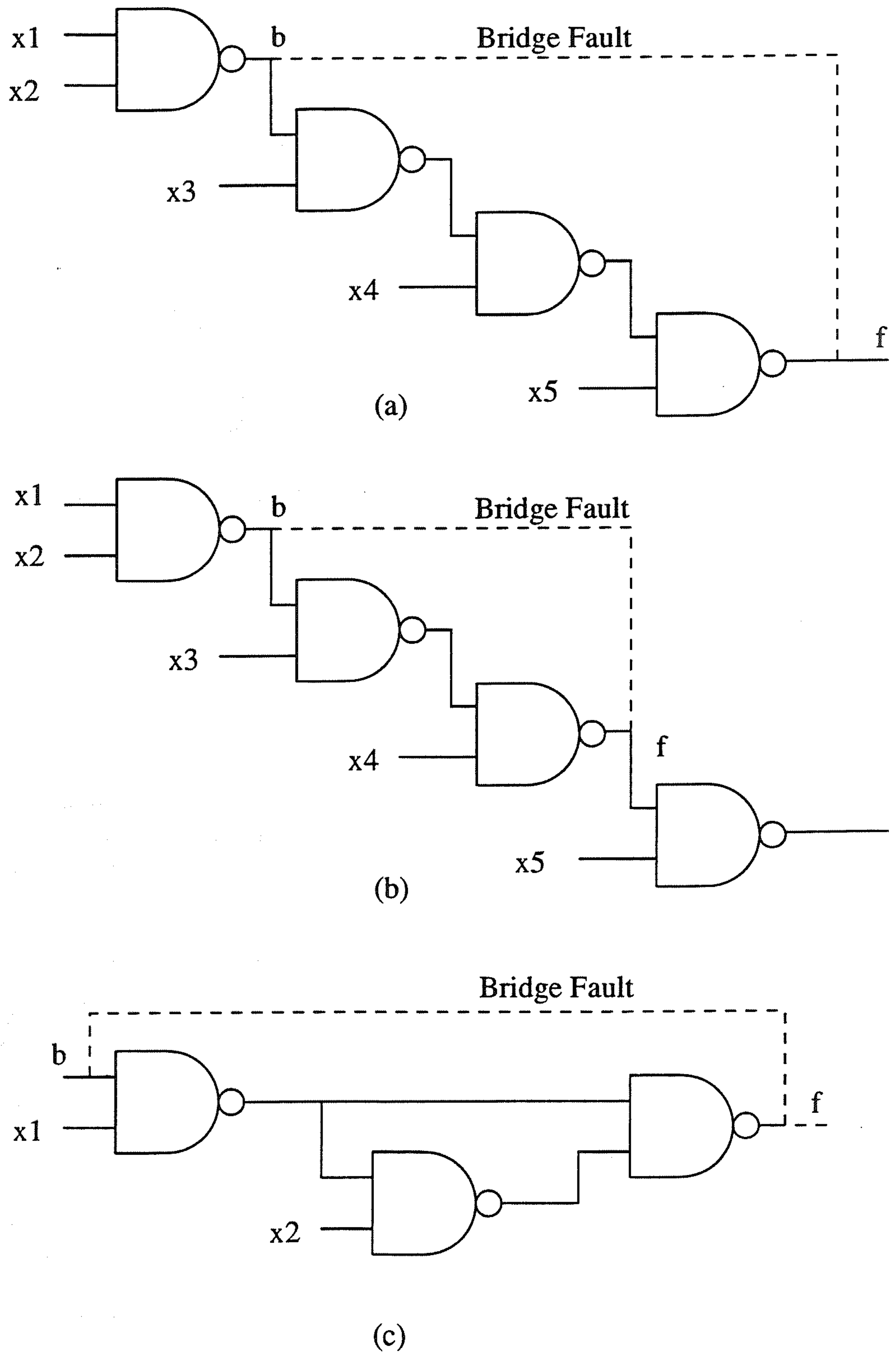


Figure 3: Feedback Bridging Faults: (a) Odd; (b) Even; (c) Neither.

A consequence of an odd feedback bridging fault is that oscillations may occur, depending upon the precise switching characteristics of the circuit [2, 9, 11, 10]. If the propagation delay of the circuit is greater than the rise time or the fall time of the back node, then the signal on the front node tends to follow the back node. This situation can be avoided if the path between the back node and the front node is desensitized. For the odd-FBF in Figure 3(a), the path between b and f can be desensitized by making x_3, x_4, or x_5 equal to logic 0. The fault can be detected by placing opposite values on the nodes and propagating the weaker node to a primary output.

However, SPICE simulations on CMOS26 1μm process gates showed that if one of the nodes is sufficiently strong, compared to the weaker node, oscillations do not occur. For the circuit in Figure 3(a), the fault is activated by $< 11100 >$, which places a strong 0 on 'b' and a weak '1' on 'f'. Since only one p-transistor is ON in the gate driving 'f', the logic 0 on 'b' outvotes the logic 1 and the shorted nodes are held at logic 0. The fault can then be detected by propagating the front node to a primary output. A FBF fault can also be activated by making the front node strong and the back node weaker. However, in some cases there may be only one path between the back node to the primary output, such as shown in Figure 3(a). In this case the back node cannot be propagated. In general, the front node is often closer to a primary output; therefore, it is desirable and advantageous to make the back node the stronger of the two when using voltage testing.

4 Power Supply Current (I_{ddq}) Testing

The voltage testing of bridging faults using minimal stuck-at fault test sets has some limitations. It was observed that most of the faults rendered undetectable by this method simply failed to propagate to the output. For the circuit in Figure 2, the complete stuck-at test set fails to detect the bridging fault involving I_0 and x_2. The only test vector that can activate this fault is $<01010>$, which places a logic 0 on I_2 and makes the value on Z_1 a logic 1 for both the faulty and the fault-free cases. As a result, the weaker node I_0 fails to propagate to the output and the fault is rendered undetectable. Compared to voltage testing, I_{ddq} testing only requires activation of a bridging fault. Activating the fault creates a direct path between V_{dd} and V_{ss}, which causes an elevation in I_{ddq}. The fault can therefore be detected by monitoring the static current.

Current testing shows an advantage over voltage testing when the bridging defect has a finite resistance. SPICE simulations of the two-NAND (CMOS26 1μm process gates) bridge in Figure 1 were used to determine the critical bridge resistance, above which the circuit was functionally correct. With all transistor lengths fixed at 1μm, when $W_p = W_n = 1\mu$m[2] the critical resistance was approximately 17KΩ. This value was reduced to 4KΩ when the widths were increased to 4μm. However, in both cases the current increased by an average factor of 15. This also indicated that the critical value of the bridge resistance is dependent on the resistance of the transistors in the pull-up and the pull-down networks.

[2] W_p and W_n are the widths of the p-channel and n-channel transistors, respectively.

5 Bridge Fault Simulation

Once a list of test patterns for a circuit was available, all possible bridging faults were simulated using the *NOVA* logic simulator to determine the coverage of the test vectors [13]. NOVA is a switch-level simulator with the capability of resolving multiple logic states. NOVA implements 21 logic states using 0, 1, and X (indeterminate) states and three strengths (active, resistive, and floating).

If two or more elements are trying to drive the same node, then NOVA determines the state with the greatest strength and assigns it to the node. For example, a node being driven by 1_{aa} and 0_{rr} will be assigned 1_{aa} because it is the stronger of the two. The subscripts represent the pull-down and pull-up strengths, respectively, of the gate driving the node. If two nodes driving the same node have opposite states but the same strength, then NOVA resolves the node to an X.

Left as is, the NOVA primitives (AND, OR, etc.) behave as wired-logic [1] under the simulation of bridging faults; this is now accepted as an incorrect model for CMOS circuitry [2, 9, 4]. Instead, "functional models" were written in the C programming language. These models were developed to implement the voting model of a bridging fault, taking into consideration the following points:

1. Relative strengths of the p-channel and the n-channel transistors must be known so that an accurate functional model could be developed.

2. CMOS devices are non-linear devices.

3. NOVA is a logic simulator and not an electrical level simulator and it was important to incorporate CMOS non-linearity in NOVA.

4. The voting model needed a resistive representation of the n-channel and the p-channel devices.

If the assumption is made that the transistors operate in the linear region such that

$$R_{c(linear)} = \frac{1}{k\frac{W}{L}(V_{gs} - V_t)} \quad \text{and} \quad V_{gs} - V_t = 4,$$

then problems 2, 3, and 4 simplify. SPICE simulations on CMOS26 (1μm process) transistors were used to develop a relationship between the relative strengths. Table 1 shows some results for different W_p and W_n. It was observed that a single n-transistor is 47% stronger than two n-transistors in series, as it sank 47% more current. Similarly, the current sourced by two p-transistors in parallel is twice that sourced by a single p-transistor. Comparing the results of single p-type and single n-type in the table, it can be deduced that a single n-type can sink twice as much current as a single p-type can source. Based on these observations, the functional models of the primitives were developed.

For the purposes of logic simulation, two states of logic 1 and logic 0 were used out of six states for each. 1_{aa} and 0_{aa} represented strong logic 1 and logic 0, 1_{ar}, 1_{rr}, 0_{ar}, and 0_{rr} represented weak logic 1 and logic 0. The other three states (for each) were not used because the combination of these signals with others resulted in a large number of indeterminate

W_p, W_n[a]	single n-type	series n-type	single p-type	parallel p-type
$1\mu m$	$172\mu A$	$110\mu A$	$64\mu A$	$128.2\mu A$
$4\mu m$	$1010\mu A$	$638\mu A$	$439\mu A$	$879.1\mu A$
$10\mu m$	$2690\mu A$	$1688\mu A$	$1180\mu A$	$600\mu A$

[a] $L_p = L_n = 1\mu m$.

Table 1: Strength Comparison of n-transistor and p-transistor

Circuit	*Patterns*	*total nodes*	*total BF*	*BF simulated*	*VD*	*ID*
C17	5	11	55	55	63.3%	100%
Full-Adder	4	12	66	66	51.51%	100%
C432	102	193	18528	2624	97.5%	99.96%
C1355	102	587	171991	3744	44.87%	99.91%

Table 2: Simulation Results.

states, even under fault-free conditions. Pseudo-code for a two input NAND functional model is shown in Figure 4.

6 Results

The minimal stuck-at fault test sets were generated using the *fault folding* [12, 7] and *PODEM* [1] algorithms. Fault folding was used for circuits whose layouts or schematics were available, whereas PODEM was used for ISCAS benchmark circuits C432 and C1355 [3]. The simulation results are shown in Table 2.

The column VD shows the percentage of faults that were detected using voltage testing. Faults were considered detected if an incorrect logic value was present at any primary output. If an indeterminate state (X, $, etc.) [13] appeared, the fault was considered undetected. The results of the voltage testing method were consistent with other's results and showed poor coverage of bridging faults.

The ID column shows the current-detected faults. A fault was considered detected if a test vector was successful in placing opposite logic values on the bridged nodes (activating the fault). For this simulation, the fault-free values of all internal nodes as well as the inputs and the outputs were recorded and then compared with the values when the fault was induced. As mentioned earlier, NOVA assigns an indeterminate state to the bridged nodes when they have opposite states but the same strength. Consequently, under current testing simulations, test patterns which resulted in indeterminate states like X, $, etc. on the bridged nodes were considered to detect the fault.

Since the fault folding method generates all possible minimal stuck-at fault test vector sets, the results shown here correspond to the test set that gave the best coverage of bridging faults. For C432 and C1355, PODEM generated only a single test vector set. For these two circuits it was not possible to test for all bridging faults because the circuit had to be recomplied and simulated for each fault. The 2624 bridging faults for C432 and 3744 faults for C1355 were randomly selected out of the total number of possible faults. For

```
if ( $X_0$ is a strong zero ) then
    if ($X_1$ is a strong zero) output a strong one;
    else if ($X_1$ is a weak zero) output a strong one;
    else output a weak one;

if ( $X_0$ is a weak zero ) then
    if ($X_1$ is a strong zero) output a strong one;
    else output a weak one;

if ( $X_0$ is a strong one ) then
    if ($X_1$ is a strong zero) output a weak one;
    else if ($X_1$ is a weak zero) output a weak one;
    else if ($X_1$ is a strong one) output a strong zero;
    else if ($X_1$ is a weak one) output a weak zero;
    else float the output;

if ( $X_0$ is a weak one ) then
    if ($X_1$ is a strong zero) output a weak one;
    else if ($X_1$ is a weak zero) output a weak one;
    else if ($X_1$ is a strong one) output a weak zero;
    else if ($X_1$ is a weak one) output a weak zero;
    else float the output;

if ( $X_0$ is indeterminate ) then
    if ($X_1$ is a strong zero) output a weak one;
    else if ($X_1$ is a weak zero) output a weak one;
    else float the output;
```

Figure 4: A Two-input NAND Functional Model.

C432, current testing failed to detect only 1 fault, whereas for C1355 only 3 faults were left undetected. The same faults were also undetected under voltage testing.

7 Conclusions

The primary objective of this work was to evaluate the effectiveness of minimal stuck-at fault test sets in detecting bridging faults. The results in Table 2 indicate that stuck-at test sets are extremely effective in detecting bridging faults when used as current test vectors. For example, in the case of C1355 the bridge fault coverage increased by 120% when current testing was used. This success is attributed to the test set's capability of toggling every single node in the circuit. It is also aided by the fact that current testing relates to the nature of a bridging fault and because of simpler detection requirements, enhances the performance of minimal stuck-at fault test sets.

8 Acknowledgments

The authors would like to thank the members of the MRC for their help and support.

References

[1] M. Abramovici, M.A. Breur, and A.D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[2] John M. Acken and Steven D. Millman, "Accurate Modeling and Simulation of Bridging Faults," *Proceedings of the Custom Integrated Circuits Conference*, San Diego, CA, May 1991, pp. 17.4.1–17.4.4.

[3] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN," *IEEE International Symposium on Circuits and Systems*, June 1985, pp. 663–698.

[4] F.J. Ferguson and T. Larabee, "Test Pattern Generation for Realistic Bridging Faults in CMOS ICs," *IEEE International Test Conference*, 1991, pp. 623–632.

[5] A.D. Friedman, "Diagnosis of Short-Circuit Faults in Combinational Circuits," *IEEE Transactions on Computers*, July 1974, pp. 746–752.

[6] C.F. Hawkins, J.M. Soden, R.R. Fritzmeier and L.K. Horning, "Quiescent Power Supply Current Measurement for CMOS IC Defect Detection," *IEEE Transactions on Industrial Electronics*, May 1989, pp. 211–218.

[7] Kilin To, "Fault Folding for Irredundant and Redundant Combinational Circuits," *IEEE Transactions on Computers*, Vol. C-22, Nov. 1973, pp. 1008–1015.

[8] K.L. Kodandapani and D.K. Pradhan, "Undetectability of Bridging Faults and Validity of Stuck-At Fault Test Sets," *IEEE Transactions on Computers*, Vol. C-29, No. 1, Jan. 1980, pp. 55–59.

[9] Y.K. Malaiya, A.P. Jayasumana and R. Rajsuman, "A Detailed Examination of Bridging Faults," *Proceedings of the International Conference on Computer Design*, Oct. 1986, pp. 78–81.

[10] K.C.Y. Mei, "Bridging and Stuck-At Faults," *IEEE Transactions on Computers*, Vol. C-23, No. 7, July 1974, pp. 720–727.

[11] Steven D. Millman and E.J. McCluskey, "Detecting Bridging Faults With Stuck-at Fault Test Sets," *IEEE International Test Conference*, 1988, pp. 773–778.

[12] Manjunath Shamanna, *Minimal Stuck-At Faults in VLSI*, Master's Thesis, University of Idaho, June 6, 1990.

[13] NASA Space Engineering Research Center for VLSI Systems Design, *NOVA Logic Simulator*, Idaho Research Foundation, University of Idaho, Moscow, ID, July 1991.

[14] J.P. Shen, W. Maly, F.J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design and Test*, Dec. 1985, pp. 13–26.

[15] T.M. Storey and W. Maly, "CMOS Bridging Fault Detection," *International Test Conference*, 1990, pp. 842–850.

Omit

Session 8
Architectures II

Chairman: Don Thelen

Robust Neural Classifier Circuits Using Asynchronous Design

John F. Hurdle, Peter R. Conwell, Erik Brunvand
Departments of Computer Science and Electrical Engineering
The University of Utah
Salt Lake City, UT 84112
Corresondence: hurdle@cs.utah.edu

Abstract - **Aerospace neural circuits must be adaptive, offer a practical size-performance ratio, and be environmentally robust. Our approach to building such circuits combines asynchronous design with a new fuzzy/neural classifier model. Asynchronous circuits offer many design advantages for neural hardware and our hybrid fuzzy/neural model, using mainly min and max operators, promises a low circuit complexity. We describe the general approach and conclude with a description of a novel use of rule-induction to further reduce circuit complexity.**

1 Introduction

Neural computation is interesting because it offers excellent performance across a broad set of tasks which are poorly served by von Neumann style processors, such as pattern classification/recognition, associative memory, symbolic reasoning, sensory fusion, and nonlinear control. Neural models come in many flavors, but they are distinguished by two common features: 1) they *learn* to map inputs to outputs (i.e., they are adaptive in the sense that they find a mapping function of interest without special knowledge of the specific underlying data), and 2) they *generalize* from what they have learned (i.e., they produce reasonable outputs when given inputs never seen before). In aerospace applications, where reliability is essential, hardware implementations are a natural way to exploit this useful style of computation.

There is a broad spectrum of neural hardware (for example, see [18]). In our work we have focussed on a class of neural devices tailored to specific applications, where low power consumption, ease of design/redesign, reliable performance, and high throughput are the cardinal design features. Application-specific neural design is an important design niche. Large, general-purpose neural chips are often predicated on a specific neural model or neural architecture and thus are locked into a specific set of computational tasks. Typically, they do not serve a specific application well, in terms of cost and performance. Our approach, outlined in detail in [6], relies on a choosing a neural model appropriate for the data of interest and then tailoring a circuit to suit both the model and the data. This results in an application-specific neural hardware implementation which is near-optimal in terms of size and speed but low in cost.

To achieve the design flexibility that such an approach requires, we use an asynchronous control paradigm. In an asynchronous circuit, the global clock found in synchronous design is discarded. Control is distributed throughout the circuit (i.e. , control is localized, not global) and strict notions of *timing* are replaced by strict notions of *sequencing.* Because there is no clock, problems like clock skew, clock distribution, and clock power are eliminated. This simplifies the design/redesign cycle because new circuit elements may be added or altered without regard to their impact on system timing. Asynchronous circuits are robust to environmental changes which can adversely affect the clock (i.e., temperature or supply voltage drift), they scale well because there is no clock to complicate the interfacing of modules, and they perform in an average-case fashion because the worst-case timing on the critical path no longer dictates system speed.

2 Neural Hardware for Aerospace Applications

The goal of neural hardware design is to map a neural model into hardware in an efficient form. Research in neural hardware falls broadly into three categories: model oriented, sensory oriented, or neurocomputer oriented. Model-oriented studies focus on implementing a particular model, such as multilayered feedforward nets, Hopfield Nets, Kohonen Maps, Adaptive Resonance Theory, and so on, in hardware form [18, 20]. Sensory-oriented studies, where a specific biological sensory organ is emulated in hardware form, have produced some interesting and practical circuits [4, 10]. Neurocomputer-oriented research explores ways of designing building blocks suitable for large-scale processors tailored to neural computation. These large-scale platforms are used to study learning algorithms or neural models requiring huge amounts of data, such as scene analysis or real-time objection recognition. The building block concept is viable because certain models rely on the massive interconnection of simple computational cells. Machines built of such blocks are essentially high performance, special-purpose computers dedicated to a subset of general neurocomputing. They have been implemented in custom and semi-custom VLSI, on field programmable gate arrays (FPGAs), and in microcontrollers [9, 12, 19].

The first solid-state neural circuits were built in the early 1970s [11]. In spite of the tremendous effort expended on building neural hardware over the past two decades, very few circuits have found their way out of the laboratory into real world devices. In particular, neurocomputer-oriented research has produced large systems, but these do not migrate easily to field applications (such as those found in aerospace) because of their expense and large power budget. They are excellent platforms for general studies, but rarely make a cost-effective solution to any specific problem. Likewise, custom chips that implement, in a general way, a given neural model usually are inefficient for specific applications.

We see a niche for small-to-medium scale neural circuits that are tailored to the specific needs of a given aerospace application. In this niche, low power consumption and environmental robustness are key, as is the need for a set of neural models which can adapt to the data set of interest. The idea is to provide, in a small and reliable package, "intelligent processing" tailored for a given job. Such jobs could include: qualitative or quantitative sensory data analysis, real-time control of nonlinear flight or fuel processes, adaptive remote vehicle or satellite processing, instrument diagnostics, etc. For such an application-specific circuit to be cost-effective, it has to perform as well as a general purpose circuit (e.g., microprocessor or microcontroller based circuits) and yet be cheaper to produce and operate.

In aerospace applications, cost is measured many ways: the dollar cost of design and fabrication, the energy cost of operation, the weight and volume costs of the final system package, and the cost of assuring reliable operation. We address these issues in two ways. First, we focus on neural models that use supervised training. This means that, for any given application, we possess a training set: a set of input vectors and their corresponding output values. This data is used to train the neural system while an independent test set is used to evaluate how well the system has learned and can generalize. To reduce the cost of design and fabrication we are developing a high-level neural synthesis system which preprocesses the training data, selects a model appropriate for the data, and synthesizes a custom circuit description [6]. Currently, this description is mapped into Actel FPGAs for testing and functional verification. Once verified, the FPGA-based system could be used directly in an single application, be factory masked for larger runs, or the underlying circuit description could be mapped to another technology, such as semi-custom standard CMOS VLSI or a space-hardened technology. Secondly, we reduce energy consumption, weight, and volume through a focus on special purpose parts which require a minimum of external support (e.g., disk drives, special data busses, etc). These chips are small and reliable because of the asynchronous design approach we use.

3 Asynchronous Neural Hardware Design

Unlike synchronous circuits, asynchronous circuits have no global clock. Where synchronous circuits use the clock and its underlying notion of strict timing to maintain control, asynchronous circuits use the notions of local communication and sequencing to maintain control. There are several ways to implement asynchronous designs [13]. We use a technique called self-timed design. In self-timed circuits, circuit modules inform each other when they are ready to send or receive data.

3.1 Implementing Two-Phase Self-Timed Circuits

The specific method we use relies on a simple 2-phase handshaking protocol and is illustrated in Figure 1. For a complete description of the protocol, see [3]. Our circuits are decomposed into modules which communicate by sending a transition (as opposed to a fixed level signal) to neighboring modules. Each transition signals an event and communication proceeds in a straightforward request-acknowledge (REQ-ACK) fashion. Typically, a module sets up data on a channel to a neighbor and issues a REQ (i.e., induces a transition, high or low, on the REQ line). When the neighbor is done with the data, it issues an ACK (i.e., induces a transition on the ACK line). The sender is then free to change data and assert another request.

Safe operation of this protocol demands that the data have arrived and settled at the receiver before the next event is signaled on the REQ line. The simplest way to do this is to model local timing delays, such as through combinational logic, by inserting a small number of buffers in the path of the REQ signal. These small gate delays restrain the REQ control signal until the data have arrived. By keeping the timing requirements local and short, we can still safely approximate delay-insensitive behavior (i.e., behavior where delays of any magnitude can be tolerated on wires and within circuit elements). Other, more elaborate

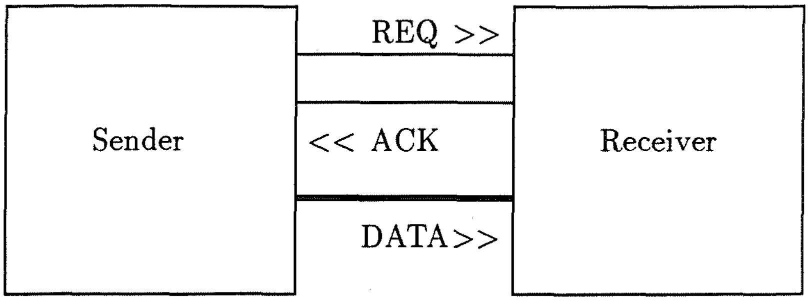


Figure 1: The Sender-Receiver Protocol

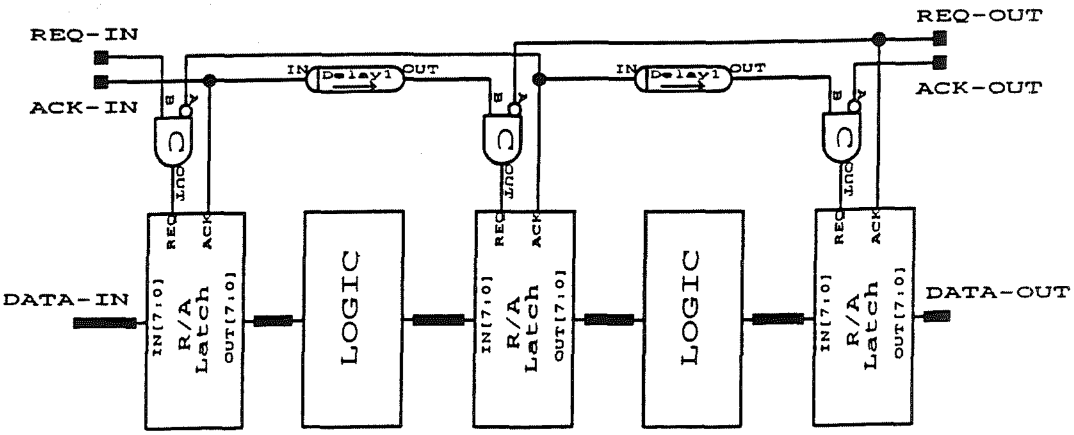


Figure 2: An Idealized Micropipeline

protocols have been designed which are truly delay insensitive, but they carry significant penalties in terms of extra data lines (such as with 4- phase, dual rail encoded data) and decoding circuitry [13]. We have chosen to conserve logic and wire resources by using the simpler 2-phase signaling convention. This results in smaller, more compact circuits than can be built with other self-timed methods. In practice, the 2-phase protocol turns out to be reliable and straightforward to implement, even with FPGA devices where the designer has little control over routing [1].

The neural models we study decompose nicely into a set of communicating subprocesses, and this decomposition, in turn, maps nicely to a pipelined operation. Sutherland has described an elegant use of 2-phase signaling in implementing novel pipeline structures which he termed micropipelines [17]. Figure 2 shows an abstracted micropipeline. The component labeled "C" (standing for c-element) provides circuit control. It issues a transition whenever both the sender has issued a REQ (signaling that new data is ready) and the receiver has issued an ACK (signaling its readiness for new data). It is the equivalent of an AND gate for transition signaling. Typically, the transition issued by the c-element controls a latch which forms the communication channel between stages. The delay elements model the delay through the logic blocks.

A virtue of micropipelines is that computations flow through the pipeline in a data-dependent fashion. For example, a micropipelined multiplier might detect at an early pipeline stage that an operand is zero or that all remaining leading bits are zero. In such a case, the computation is already complete and the result could be forwarded to the end of the pipeline without further processing. In a synchronous version of a pipelined multiplier, the clock would continue to clock this result through the pipeline, pausing long enough at each stage to allow (worst-case) processing to occur. In the micropipelined version, the result flows through to the end as quickly as the remaining stages can forward it. This results in average-case performance which, depending on the data and the underlying computation, can offer a substantial improvement over the worst-case performance typical in synchronous pipeline designs.

3.2 Self-Timed Neural Hardware

Because they have no clock, self-timed circuits can be used to good advantage in neural hardware design. We detail these points in depth in [7] and summarize them here. The self-timed approach confers several general important design advantages:

- Robust Performance: Because there is no clock, variations in temperature, supply voltage, and manufacturing process can alter the throughput of a self- timed device but the device will still continue to operate correctly (which may not be the case for a synchronous system in a similar, dynamic environment). In many applications, correct behavior is essential while throughput may be allowed to vary across an acceptable range.

- Low Energy Consumption: In a synchronous system, the clock consumes power even when the circuit is idle. Self-timed circuits can be designed to consume power only when data is being processed and then only in a data-dependent way. For systems that operate intermittently or which process data arriving asynchronously, this savings in energy consumption can be significant.

- Ease of Design/Redesign and Scaling: Since global clocking is not an issue with self-timed circuits, new modules may be added, or old ones updated, without concern for the impact on overall system timing. In a micropipelined circuit, this means that a component which is performing too slowly may be replaced with a newer, faster version and the system will still perform correctly. This also means that self-timed circuits can be easily scaled up to practically any size with a similar guarantee of correctness. Finally, the problems of designing a clock distribution network (clock routing, clock skew, etc) are eliminated, which simplifies the design/redesign process.

- Average Case Performance: A well designed self-timed system can exploit data dependencies to speed computation (e.g., the multiplier example mentioned above). This is possible because circuit components are allowed to consume and produce data as quickly as they can, as opposed to synchronous parts which have to wait for the slowest component on the critical path.

These general self-timed advantages dovetail well with the needs of neurocomputing, especially in aerospace applications:

- Neural Hardware Will be Used in Dynamic Environments: It makes sense to exploit the robustness of self-timed devices in aerospace systems because self-timed circuits provide correct operation even in environments where voltage and temperature parameters are dynamic and often unpredictable.
- Neural Hardware Will Process Intermittent, Asynchronous Data: Data rates in real applications vary widely and data typically arrives asynchronously. Self-timed hardware consumes power in a data dependent fashion, which results in minimal energy drain (i.e., on an "Ras needed" basis). Also, because this hardware is inherently asynchronous, it naturally interfaces to asynchronous data sources.
- Neural Algorithms Are Fluid: Being relatively new, most neural algorithms are still being reformulated or refined. Self-timed circuit implementations gracefully support redesign, so an extant neural circuit can be changed to reflect an algorithm refinement with minimum design overhead.
- Neural Models Are Arithmetic Intensive: Neural processing typically relies heavily on arithmetic processing. Owing to their average-case behavior, self-timed circuits excel in this style of processing.

As attractive as it is, using self-timed design to build neural hardware is not without its drawbacks. First, few commercial CAD tools can support this style of design, although this is an active area of research internationally in the academic community [5]. Second, since most digital hardware is predicated on synchronous design, any practical asynchronous circuit will inevitably be interfaced to a synchronous component. Interfacing synchronous and asynchronous parts is not trivial, mainly due to the possibility of metastability failure at the interface latches. However, practical interfaces have been proposed and designed [8]. Third, self-timed circuits tend to be larger than their synchronous counterparts. This results from the need to add circuitry to support the local handshaking protocol between modules. How much larger these circuits are depends on the specific implementation, but in our experience with micropipeline organizations it tends to be small (on the order of 10-20%). Lastly, there are no studies which examine how well self-timed circuits perform in high radiation environments. One might expect self-timed circuits to be more vulnerable than synchronous circuits, because synchronous circuits are mainly at risk only in a narrow window around the clock edge (for control errors). But the question is still open and merits further study.

4 An Example: A Self-Timed Fuzzy-Neural Classifier

We have reviewed many neural models to assess their suitability for application-specific circuit synthesis. Several lend themselves to our special need for architectural plasticity. One in particular, a classifier developed by Simpson and based on fuzzy min-max (FMM) network concepts, shows special promise [15]. We present in this section a brief overview of the model and some implementation issues under a self-timed circuit design approach.

4.1 Overview of the FMM Model

FMM offers several attractive features: 1) it trains with a time complexity which is fixed and linear in the number of input vectors and input vector dimensionality; 2) it relies primarily on min and max operators, which are straightforward and fast in hardware; 3) it exhibits excellent generalization; and 4) new training data may be added without retraining on previous data, which simplifies on-chip learning. Space limitations prevent a detailed description of the learning algorithm and architecture here. Briefly, the training vector set is comprised of a set of ordered pairs [input vector, class], where the input vector is a vector of real-valued elements of fixed length, and the class is the corresponding class index of this vector. FMM learns by building a set of fuzzy hyperboxes (defined simply by min and max values at each of the box's vertices) for each of the output classes. The algorithm consists of:

```
For each training input vector,
    For each fuzzy hyperbox of the corresponding output class,
        Compute the membership function of the vector in that box
        (the average of the difference of the min and max points
        which define the fuzzy hyperbox, dimension by dimension,
        and the input vector elements);
        If (membership function is sufficiently inside an existing hyperbox)
           adjust the min and max vertices to completely include the input vector
        Else
           build a new hyperbox which contains this input vector;
    If (the new or expanded hyperbox overlaps a hyperbox from another class)
        adjust (in a minimal way) the vertices of the overlapping boxes to
        eliminate the overlap.
```

The final network consists of the values of the min and max values for each hyperbox, organized by output class. To evaluate a new vector (in testing or operational mode), the degree of membership of the vector in each of the classes is computed and forwarded to a winner take all (WTA) module if a hard classification decision is required. If the desired output should be "soft" (e.g., a qualitative assessment of the input vector) the membership values themselves could be used to indicate the degree to which the vector falls into each class. Since each input vector is examined only once it is easy to see why FMM trains in linear time. New training vectors are accommodated at any time by simply passing them through the training algorithm, which gives FMM a tremendous advantage over other methods, like backwards error propagation training where the addition of new training vectors usually requires complete (and lengthy) retraining.

4.2 Implementation Issues: Self-Timed FMM

Since FMM constructs a network specifically adapted and minimized for a given data set, it is ideal for our application-specific work. Arithmetically, in both training and operational modes, FMM uses only multiplication, division, addition, and the two fuzzy operators min and max. Avoiding the transcendental functions (e.g., exponentiation) found in most other neural models greatly simplifies the final circuit. In Simpson's original formulation, all values were real and were mapped to the unit interval [0..1]. To avoid floating point calculations and thus simplify the hardware, we have extended the algorithm to an integer-based form, and our precision studies have shown that most models can be processed using integer-based arithmetic without loss of classification accuracy (typical bit precision varies from 8 to 16 bits).

To further simplify the circuit and speed calculation, we note that the multiplications embedded in the algorithm involve only two model parameters, *gamma* (which controls the rate at which the membership function falls off as the input vector moves farther from a hyperbox) and *theta* (which controls when to build a new hyperbox or expand an existing one). Simulations have shown that gamma's value is not critical. Theta's value, on the other hand, is highly important. When theta is very large, too few hyperboxes are built and FMM generalizes poorly. When it is very small, too many hyperboxes are built and FMM's throughput drops. One of the tasks of the synthesis system is to find an optimal value for theta which preserves both performance and generalization. In our integer version, the search for optimal values may be constrained to powers-of-two, as this results in multiplications that can be implemented as shifts in hardware, decreasing circuit complexity.

The hardware division is unavoidable, but its impact can be reduced. Division is used in FMM as part of the calculation of the membership function. It entails dividing a sum of fuzzy min-max operations by the number of dimensions in the input vector. To the extent that we can reduce the number of dimensions we can decrease the complexity of the divider, because the divider is customized to meet the needs of a given data set. Secondarily, by reducing the number of input dimensions we can also decrease the final number of hyperboxes, because we have simplified the complexity of the 'training space." One way to reduce input dimensions is to identify those dimensions which contribute little, in an information-theoretic sense, to the classification process. We use a technique described by Smyth and Goodman [16]. It relies on inducing a set of rules which describe a data set in a minimal way.

Although Smyth and Goodman derived this technique to explore the rule base underlying a given data set, we note that the rules can also be used to explore the dimension space implicit in the data. In particular, we use the rule induction technique to find rules on 2-fold cross validation subsets of the training set. If a given dimension is never used (or is used very sparingly) in the rules induced in the cross validation runs, then we eliminate the dimension from consideration by FMM. For example, we used this technique with ionosphere radar reflection data [14]. The original input vector had 34 elements, but we determined that only 12 dimensions held useful information using rule induction. The simulated run of FMM on the 12-dimension version of the data and the 34-dimension version yielded overall classification rates which varied by only a few percent. A neural circuit for the 12-dimension version would save two bits in the divider (4 bits versus 6 bits for the 34-dimension version) and the final number of hyperboxes was about 33% smaller, further increasing throughput of the final FMM design.

There are several ways we can exploit self-timed design in the FMM neural circuit. First, data dependencies in the arithmetic modules can used to signal early completion. For example, carry-completion sensing adders can be used to signal when additions are complete (rather than waiting for a worst-case propagation through a synchronous adder). Also, the divider described above can be built using a division algorithm (say, SRT) that signals when leading divisor bits are all zero, again avoiding the wait for worst-case delays. Second, if the environment is supplying input vector data asynchronously, knowledge of which input dimensions can be ignored can be embedded in the input buffer of the self-timed FMM circuit. This allows FMM to immediately request a new dimension, skipping over useless dimensions, and thus to consume the input vector more quickly than if it had to wait for the useless dimensions to be processed. In the case of radar data already mentioned, it could skip over 64% of the input data. Lastly, modules in the FMM circuit can be easily replaced with faster versions to increase throughput without regard to system timing issues. For example, the current bottleneck in our FMM design is the rate with which the external RAM holding the min/max hyperbox vertices can supply data. Substituting faster RAM would result in a faster circuit and again carries no penalty in redesign.

5 Conclusion

The self-timed design techniques outlined here have been used in a variety circuits, including a small RISC CPU [2]. Circuits for several neural models, including FMM, have been designed down to the layout level and are undergoing simulation. Unlike processor design, where the design tradeoffs are well understood, application-specific neural design is an immature field. The design tradeoffs are not obvious. The advantage of a rapid design/redesign cycle under a self-timed paradigm is particularly useful in these studies. Neural circuits are arithmetic-intensive and will likely be used in environments where data arrives asynchronously. A self-timed approach is a good match generally to neural computation because it affords average-case numeric processing, promises low power consumption, interfaces gracefully to asynchronous data sources, and is inherently robust in operating environments where temperature and supply voltages vary significantly. All of these points suggest that self-timed neural circuits could be used effectively in aerospace applications.

6 Acknowledgements

The authors would like to thank Patrick K. Simpson for his perspectives on the utility of FMM in hardware form and on dimensional reduction. We also thank the members of the Asynchronous Systems Design Group at the University of Utah, especially Luli Josephson, for several helpful discussions in the area of self-timed circuit issues.

7 References

[1] Brunvand, E. L., "Implementing Self-Timed Systems with FPGAs", in *FPGAs*, W. R. Moore and W. Luk (ed), Abingdon: EE and CS Books, pp. 312-323, 1991.

[2] Brunvand, E. L., "Using FPGAs to Prototype a Self-Timed Computer", to appear in *Journal of VLSI and Signal Processing, 1992.*

[3] Brunvand, E. L. and Sproull, R. F., "Translating Concurrent Communicating Programs into Delay-Insensitive Circuits", in *Int. Conf. on Computer- Aided Design*, 1989.

[4] Card, H. C. and Moore, W. R., "Silicon Models of Associative Learning in Aplysia", *Neural Networks*, vol. 3 (2), pp. 333-346, 1990.

[5] Gopalakrishnan, G. and Brunvand, E., "Minitrack on Asynchronous Design Methods", in *26th Hawaii Int. Conf. on System Science*, Volume 1, Maui HA, 1993.

[6] Hurdle, J. F., "Efficient Asynchronous Architectures for Neural Hardware Systems", Ph.D. Dissertation, The University of Utah, in preparation.

[7] Hurdle, J. F., Josephson, L., Brunvand, E. L. and Gopalakrishnan, G., "Asynchronous Models for Large Scale Neurocomputing Applications", in *Fifth Int. Conf. on Neural Networks and Their Applications*, Nimes (France), 1992.

[8] Josephson, L., "Mixed Asynchronous and Synchronous Circuits: Design and Evaluation", Masters Thesis, Dept. of Computer Science, The University of Utah, in preparation.

[9] Lehmann, C., Viredaz, M. and Blayo, F., "A Generic Systolic Array Building Block for Neural Networks with On-Chip Learning", *IEEE Trans Neural* Networks, vol. 4 (3), pp. 400-407, 1993.

[10] Mead, C., Analog VLSI and Neural Systems. Reading: Addison-Wesley, 1989.

[11] Nakano, K., "Associatron– a Model of Associative Memory", *IEEE Trans. on Sys., Man, and Cybernetics*, vol. 2, pp. 380-388, 1972.

[12] Pacheco, M., Bavan, S., Lee, M. and Treleaven, P., "Simple VLSI Architecture for Neurocomputing.", *Neural Networks*, vol. 1 (1 SUPPL), pp. 398, 1988.

[13] Seitz, C., "System Timing", in *Introduction to VLSI Systems* (chapter 7), C.A. Mead and L. Conway (ed), Reading, MA: Addison-Wesley, pp. 218-262, 1980.

[14] Sigillito, V. G., Wing, S. P., Hutton, L. V. and Baker, K. B., "Classification of Radar Returns from the Ionosphere Using Neural Networks", Johns Hopkins APL Technical Digest, 1989.

[15] Simpson, P. K., "Fuzzy Min-Max Neural Networks- Part 1: Classification", *IEEE Trans. Neural Networks*, vol. 3 (5), pp. 776-786, 1992.

[16] Smyth, P. and Goodman, R. M., "An Information Theoretic Approach to Rule Induction From Databases", *Trans. Knowledge Data Engineering*, vol. 14 (4), pp. 301-316, 1992.

[17] Sutherland, I. E., "Micropipelines", *Comm. of the ACM*, vol. 32 (6), pp. 720-738, 1989.

[18] Treleaven, P., Pacheco, M. and Vellasco, M., "VLSI Architectures for Neural Networks", *IEEE Micro*, vol. 9 (6), pp. 8-27, 1989.

[19] Wawrzynek, J., Asanovic, A. and Morgan, N., "The Design of a Neuro- Microprocessor", *IEEE Trans. Neural Networks*, vol. 4 (3), pp. 394-399, 1993.

[20] Zornetzer, S. F., Davis, J. L. and Lau, C., *An Introduction to Neural and Electronic Networks*, Academic Press, 1990.

High Speed Sequential Decoders

Jay H. Harris, Kurt Kloesel, and Brett A. Bernath
San Diego Univeristy
E-mail: harris@ecesun1.sdsu.edu

Abstract - **Design of high speed decoders for long constraint length convolutional codes with code rates of 1/2, 2/3, and possibly greater is described. The decoders implement the Z-J algorithm in single chip systems and perform the basic sort operation of the algorithm in one chip clock cycle. Pipelining and carry-select arithmetic operations are employed. Good performance at clock rates of 100 MHz and above and at associated symbol rates greater than half this value is expected. Results of performance simulations and low clock rate subsystem tests are described.**

Acceleration-Augmented LQG Control of an Active Magnetic Bearing

Joseph J. Feeley
Department of Electrical Engineering and Microelectronics Research Center
University of Idaho
Moscow ID 83843

Abstract - **A linear-quadratic-gaussian (LQG) regulator controller design for an acceleration-augmented active magnetic bearing (AMB) is outlined. Acceleration augmentation is a key feature in providing improved dynamic performance of the controller. The optimal control formulation provides a convenient method of trading-off fast transient response and force attenuation as control objectives.**

1 Introduction

Active magnetic bearings (AMB) are being considered for an increasing number of applications. In particular, their simplicity and reliability offer many advantages for space applications. For example, magnetic bearings and actuators are used exclusively in a space-borne cryogenic cooler developed by Philips Laboratories for NASA [1].

In many applications it is desirable that forces originating on the moving shaft be attenuated as much as possible before being transmitted to the bearing. In space applications this is especially important since these forces may cause undesirable motion of the host satellite. Careful mechanical design and various counterbalancing schemes have been used to achieve this objective. In addition, the flexibility afforded by AMB control can also play an important role in this regard. While the primary objective of the AMB control system is to keep the shaft centered in the bearing, active control can also be used to dynamically adjust bearing stiffness to attenuate force transmittal from the shaft to the bearing. In this context the AMB control system design becomes an optimization problem, trading-off transient response in shaft centering with force attenuation. The techniques of modern automatic control theory are well-suited to solving this design problem.

2 Acceleration-Augmented Design

At first glance, the most direct method of dealing with shaft forces appears to be to measure the acceleration of the shaft and use that signal for explicit force cancellation in the AMB controller. Unfortunately, it is seldom practical or economical to measure shaft acceleration directly; shaft position is usually the only measurement available for AMB control. It is also generally impractical to compute shaft acceleration by twice differentiating position because of the noise associated with the position measurement. Neither is it generally possible to use observer theory directly to compute acceleration because acceleration is not a state variable in traditional shaft dynamic models. However, if the shaft disturbance force is modeled as a random process, the resulting state variable model permits estimation of the shaft acceleration in the context of a Kalman filter. The estimated acceleration may then be used in a stochastic optimal control scheme based on the separation principle. This procedure is outlined below.

A complete set of model equations for a magnetic bearing with eddy currents in the actuator has been previously developed [2]. Only a simplified form of the equations of shaft dynamics are required to illustrate the acceleration-augmented control scheme suggested here. The simplified equations of motion of the shaft, in state variable form, are

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = \frac{1}{m}(u_c + x_3)$$

where

$x_1 =$ position
$x_2 =$ velocity
$x_3 =$ disturbance force
$u_c =$ control force
$m =$ mass of shaft

To a first order approximation, it is quite reasonable to model the disturbance force as a random walk

$$\dot{x}_3 = w$$

where

$w =$ white noise process with power spectral density Q

Assuming that the position measurement is corrupted by additive white noise, the measurement equation becomes

$$z = x_1 + v$$

where

$v =$ white noise process with power spectral density R

The foregoing equations can be written in standard state variable form as

$$\dot{x} = Fx + G_c u_c + G_d w$$

$$z = Hx + v$$

The usual LQG techniques employing the separation principle can now be used to find the steady state gains for the optimal controller and estimator.

The optimal control is found to be

$$u_c = -K_c \hat{x}$$

where

$\hat{x} =$ state estimate
$K_c =$ controller gain

The controller gain is given by

$$K_c = B^{-1} G_c^T S$$

where S is the steady-state solution to the matrix Riccati equation

$$\dot{S} = -SF - F^T S + SG_c B^{-1} G_c^T S - A$$

and

$$\begin{aligned} A = &\quad \text{performance index weighting on states} \\ B = &\quad \text{performance index weighting on control input} \end{aligned}$$

The controller gain and the bearing stiffness can be adjusted by changing the performance index weighting on the states and the control effort.

The optimal state estimates are the solutions to

$$\dot{\hat{x}} = F\hat{x} + G_c u + K_e(z - H\hat{x})$$

where the optimal estimator gain is

$$K_e = PH^T R^{-1}$$

and P is the steady state solution of the matrix Riccati equation

$$\dot{P} = FP + PF^T + G_d Q G_d^T - PH^T R^{-1} HP$$

To the extent that the controller model is accurate this control scheme provides optimal AMB control. The relative importance the controller places on centering control versus force attenuation is controlled by the relative magnitudes of the A and B matrices, respectively.

3 Results

The results of applying this acceleration-augmented design approach to the simplified magnetic bearing model are described below. It is instructive to use the simplified model because it yields closed form solutions [3] which provide considerable insight into the problem. Assuming unit mass and direct measurement of position only, the relevant matrices to characterize the system are

$$F = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$Gc = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$$

$$Gd = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

$$H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$Q = q$$

$$R = r$$

Under these conditions the Kalman gain becomes [3]

$$K_e = [2(\sigma)^{1/6} 2(\sigma)^{1/3} (\sigma)]^{1/2}$$

where $\sigma = \frac{q}{r}$ is a measure of the system signal to noise ratio. The estimator equations then become

$$\begin{aligned}
\dot{\hat{x}}_1 &= \hat{x}_2 + 2(\sigma)^{1/6}(z - \hat{x}_1) \\
\dot{\hat{x}}_2 &= \hat{x}_3 + u + 2(\sigma)^{1/3}(z - \hat{x}_1) \\
\dot{\hat{x}}_3 &= (\sigma)^{1/2}(z - \hat{x}_1)
\end{aligned}$$

Since the state is uncontrollable, the optimal control problem is solved by considering only states and in the performance index. With unit weighting on the position, zero weighting on the velocity and weighting on the control input, the weighting matrices become

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$B = \frac{1}{c}$$

The optimal gain is found to be [3]

$$K_c = [c\ (2c)^{1/2}\ 1]$$

A block diagram of the closed-loop system is shown in Figure 1.

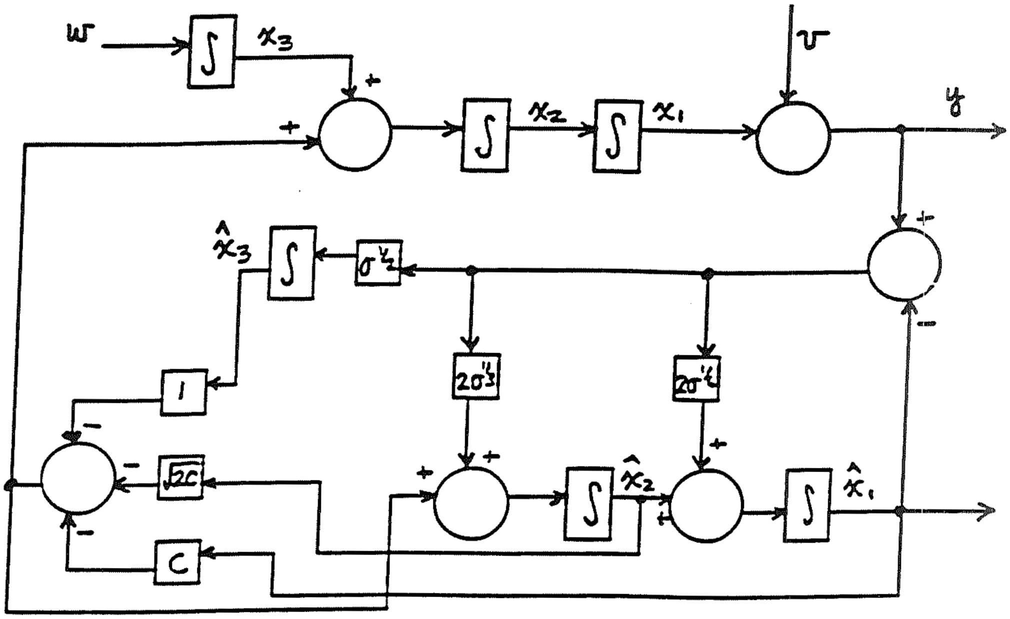


Figure 1: Block diagram of acceleration-augmented AMB

The closed-loop system has eigenvalues at

$$s_1 = 0$$

$$s_{2,3} = -\frac{(2c)^{1/2}}{2}(1 \pm j)$$

$$s_4 = -(\sigma)^{1/6}$$

$$s_6 = -(\sigma)^{1/6}\left(\frac{1}{2} \pm \frac{(3)}{2}\right)$$

It is clear from the above that the control effort weight c and the signal to noise ratio σ can be used as design parameters to adjust system performance.

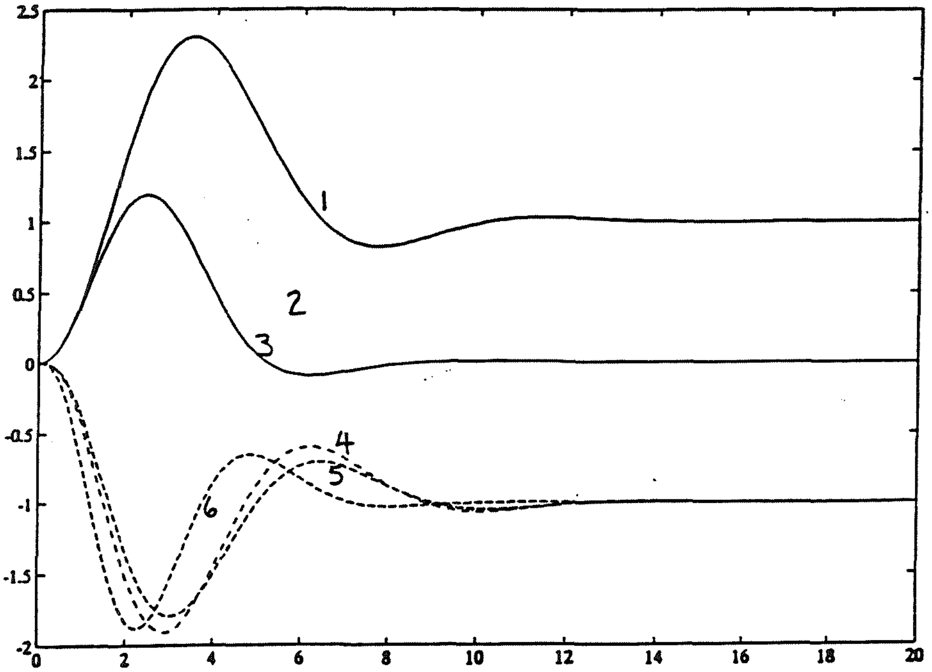


Figure 2: (1) Position response, no augmentation. (2) Position response with augmentation. (3) Position response with augmentation and large σ. (4) Control effort, no augmentation. (5) Control effort with augmentation. (6) Control effort with augmentation and large σ.

The effects of acceleration-augmented control and the use of the design parameters is illustrated in Figure 2. Traces 1,2, and 3 show the response of shaft position to an impulsive input disturbance (unit step increase in disturbance force). Trace 1 is for the system without acceleration-augmentation, while traces two and three show the response of the system with acceleration-augmentation included. Not surprisingly, acceleration- augmentation reduces the error in desired shaft position to zero since (somewhat surprisingly) for this simple case acceleration augmentation has the effect of introducing an integral control mode. Traces 2 and 3 show the effect of increasing the signal to noise ratio. Increasing σ to 10 moves the estimator poles away from the origin and reduces the response time of the system. The control effort associated with each of these responses is shown in traces 4,5, and 6, respectively. Notice that not only does the system respond faster with acceleration-augmentation and large σ, but it also uses less control effort as shown by trace 6.

4 Conclusions and Future Work

This paper shows that previously used "force control" schemes [4] are readily explained in terms of the stochastic control of a properly augmented linear dynamic model of a magnetic bearing system. Performance is readily modified to meet the desired trade-off between transient centering control and force attenuation by adjustment of control system design parameters. Nonlinear simulation studies are underway to quantify the performance improvement afforded by acceleration-augmented LQG AMB control. Efficient digital implementation of the controller is also under investigation.

References

[1] F. Stolfi et. al., "Design and Fabrication of a Long-Life Stirling Cycle Cooler for Space Application," Philips Laboratories Report, March, 1983.

[2] J.J. Feeley and D.J. Ahlstrom, "A New Eddy Current Model for Magnetic Bearing Control System Design," Fourth NASA Symposium on VLSI Design, October, 1992.

[3] B. Friedland, *Control Systems Design*, McGraw-Hill Book Company, pp. 451-455, 1986.

[4] H.M. Chen and M.S. Darlow, "Magnetic Bearing With Rotating Force Control," Transactions of the ASME, Journal of Tribology, Vol. 10, pp. 100-105, January, 1988.

Multiple-Fold Clustered Processor Mesh Array

Gerald G. Pechanek - IBM MWAVE Architecture & OS Developemnt, RTP, NC

Stamatis Vassiliadis - IBM Enterprise Systems, Endicott, NY and
Electrical Engineering Deptarment, State Univeristy of New York at
Binghamton, NY and Cornell University, Ithaca, NY

Jose G. Delgado-Frias Electrical Engineering Deptarment,
State Univeristy of New York at Binghamton, NY

Abstract - **The multiple-fold clustered processor mesh array is a triangular organization of clustered processing elements. This multiple-fold array maintains functional equivalence to the nearest neighbor mesh computer with uni-directional interprocessor communications, but with half the number of connection wires. In addition, the connectivity of the multiple-folded organization is superior to the standard square mesh due to the improved connectivity between the clustered processors. One of the primary application areas targeted is High Performance Architectures for image processing.**

1 Introduction

The processing of visual information can be considered to consist of three different processing domains: image processing, pattern recognition, and computer graphics [1]. The merger of these three domains is referred to as image computing and represents a capability required by the multimedia workstations of the future. Since image computing represents a high computational load for a processing system, special purpose processors have been proposed to off load the image processing task from the system processor. One of the special purpose processors used for image computing is the nearest neighbor mesh connected computer organization [2-4].

The massively parallel diagonal-fold array processor [5, 6] consists of a diagonally folded mesh computer which, with uni-directional interprocessor communications, provides the same functionality as a square mesh computer but with half the number of connection wires. By folding a mesh computer organization multiple times and organizing the processing elements into local clusters it is possible to enhance the connectivity of the original mesh and provide a simple method of scaling the array.

Two new machine organizations are presented utilizing the multiple-folding concept to emulate and extend the capabilities of the square four neighborhood and the hexagonal six neighborhood processor arrays. Also discussed is the method of enhancing the connectivity and how scaling is performed on the multiple-fold structures. A discussion of our further research and a summary concludes the paper.

2 Nearest Neighbor Mesh Organization

The nearest neighbor mesh connected computer [2-4] is shown in Figure 1A, where multiple Processor Elements (PEs) are connected to their north, south, east, and west neighbor PEs and all PEs are operated in a synchronous Single Instruction Multiple Data (SIMD) fashion. It is assumed that a PE can communicate with any of its neighboring PEs but only one neighbor PE at a time. For example, each PE can communicate to their east neighbor PE, in one communication cycle. It is also assumed that a broadcast mechanism is present.

In Figure 1A, the processing elements are labeled as $\mathrm{PE_{ij}}$ where i denotes the matrix column and j denotes the matrix row. Each $\mathrm{PE_{ij}}$ processing element contains four interface ports labeled North (N), East (E), South (S), and West (W). Figure 1A is an M by M mesh array with $M = 4$, i.e. an 4×4 array, resulting in M^2 PEs. With wraparound connections, i.e. a torus configuration, each row contains M interconnection wires and with M rows, there are M^2 horizontal interconnection wires. Each column contains M interconnection wires and with M columns, there are M^2 vertical interconnection wires. The total number of wires in the mesh connected computer with wraparound connections is $2M^2(K)$, where K is equal to the number of interprocessor interconnection wires.

3 Multiple Folded Mesh Array Processor

In the mesh connected computer organization of Figure 1A, it can be noted that during any uni-directional transmission of information in the structure only 50% of the PE connection wires are utilized. It is desirable to achieve the same functionality and performance as the mesh connected computer with half the connecting wires since this amounts to a savings of $M^2(K)$ wires with corresponding savings in chip real estate. The diagonal-fold computer organization achieves this capability. The first step in the process of creating the diagonal-fold organization of PEs is to fold a square mesh connected computer along the $\mathrm{PE_{ii}}$ diagonal elements, i.e. $i = j$. For example, folding the 4×4 mesh depicted in Figure 1A results in the diagonal-fold mesh [5, 6] shown in Figure 1B, where the symmetric PEs, $\mathrm{PE_{ij}}$ and $\mathrm{PE_{ji}}$ are placed together. The symmetric PEs share transmission and reception ports, symbolically indicated in Figure 1C where the top PE's, $\mathrm{PE_{ij}}$ W, S, N, and E ports are shared with the bottom PE's, $\mathrm{PE_{ji}}$ N, E, W, and S ports respectively. The total number of diagonal-fold mesh array wires was demonstrated to be $M^2(K)$ where K is equal to the number of interprocessor interconnection wires, which is exactly one half the number of wires used in the square mesh organization [5, 6].

The triangular mesh of Figure 1B can be folded again along the diagonal made up of PEs $(PE-14, PE-41)$ and $(PE-23, PE-32)$ resulting in the triangular array shown in Figure 1D. The triangular mesh of Figure 1D can be folded again along the imaginary line between the following two groups of processors: group-1 made up of $(PE-12/PE-21, PE-34/PE-43, PE-22$, and $PE-33)$ and group-2 made up of $(PE-13/PE-31, PE-24/PE-42$, and $PE-23/PE-32)$ resulting in the triangular array shown in Figure 1E. Figure 1F and 1G depict the symbols used for the PEs in the third fold organization. The 4×4 Mulitple-Fold Clustered Processor Mesh Array, shown in Figure 1E, is redrawn, Figure 2A, using simplified clustered processor symbols. To provide a simplified referencing notation for identifying the shared port lines of a clustered PE element, the port number corresponds to the dual symmetric PEs identified by the PE with an asterisk, Figure 2A and 3B. It can be noticed in Figure 2A that even with the multiple folds the wiring between the clusters of processors is very regular and that there are no long wires required to implement the torus wrap around wiring, just local wiring within processor clusters.

The Symmetric and Diagonal PEs are grouped together through the two additional folds of the structure. After three folds, the eight processors in a group, can be identified given one set of i, j values per eight processor cluster as follows:

$$PE_{ij}, PE_{ji}, PE_{i,M-j+1}, PE_{M-j+1,i}, PE_{j,M-i+1}, PE_{M-i+1,j}, PE_{M-j+1,M-i+1} \text{ and } PE_{M-i+1,M-j+1}$$

For a particular value of M, the choices of i, j for the eight processor clusters are restricted by $1 \leq i \leq j \leq M/2, 1 \leq j \leq M/2, i \neq j$ and $ij \neq ji$. After three folds, the four processors in a diagonal group with $\leq j \leq M/2$ are:

$$PE_{ij}, PE_{i,M-i+1}, PE_{M-i+1,i}, \text{ and } PE_{M-i+1,M-i+1}$$

4 Enhancing the Connectivity

Increased connectivity can be obtained in the three-times-folded structure by improving the communications means between the processors that were placed together due to the folding. This connectivity, which in the folded structure is provided locally within a PE cluster would not be easily added in the square mesh due to the number of wires and numerous wire crossings that would occur if equivalent wiring was attempted. Many approaches can be taken to incorporate the improved connectivity within clusters of PEs. One approach is to add additional I/O ports per processor while another approach combines the processors in the cluster with a multi-port register/storage array. An example of the additional wires that would need to be added in a square mesh to provide equivalent logical connectivity for just one cluster is shown in Figure 2B. For Figure 2B, the additional wires added provide logical interconnectivity equivalence to the single clustered processor ($PE-1,2, PE-2,1, PE-1,3, PE-3,1, PE-3,4, PE-4,3, PE-2,4$, and $PE-4,2$) shown in Figure 2A.

5 Scalability of the Multiple-Fold Array

By examining Figure 1E and comparing the location of the processors in reference to the square mesh shown in Figure 1A it can be noted that the first column of PEs in Figure 1E contain the N,S,E,W edge PEs of the square mesh. In general, a concentric ring of processor elements in the 2-D mesh becomes a column of clustered processor elements in the multiple-fold array. The consequence of this can be seen in relation to the wrap-around wires which become local connections within the 1st column of PE clusters. An additional consequence concerns the scalability of the triangular array. A $Z \times Z$ mesh array, for example the 4×4 mesh of Figure 1A, can be expanded to a $(Z+2) \times (Z+2)$ array, 6×6 array Figure 3A, by adding a concentric ring of processors, indicated by PEs with asterisks, to the original mesh and reconnecting the now longer wrap-around wires accordingly. In the multiple-fold clustered processor mesh array, the same expansion can be obtained by adding a new column of PEs, breaking the $Z \times Z$ local wrap-around wiring and connecting the previous wrap-around wire ports to the added column of PEs' appropriate I/O ports, Figure 3B. An example of the local wraparound wiring for a 4×4 array is shown in Figure 2A. The new added column of PEs contain the $(Z+2) \times (Z+2)$ wrap-around wiring for the new mesh array. In both cases the PEs are renumbered.

6 Multiple-Fold Hexagonal Array

In this section the folding concepts developed in the previous sections are applied to the six neighborhood hexagonal processor array, Figure 4A. The processor nodes in Figure 4A are labeled Cxx for nodes in the outer concentric ring of processors, and Dxx for the next concentric ring of processors, continuing until the central processor node E01 is reached. The hexagonal processor array of Figure 4A is folded along the diagonal made up of nodes C01, D01, E01, D04, and C07 which results in the processor organization depicted in Figure 4B. In Figure 4B the processor nodes which overlapped due to the folding are combined into dual processor nodes, for example C02/C12 and D03/D05. The interconnecting wires depicted, in general represent multiple communication paths some of which may be shared for unidirectional communication applications to reduce wiring complexity.

The folded hexagonal array of Figure 4B is next folded along two diagonals resulting in two separate fold operations. The first fold is along diagonal C03/C11, D02/D06, and E01. The second fold is along diagonal C05/C09, D03/D05, and E01. Both folds are inward with processor nodes folding over the triangular array that is bounded by the stated diagonals. The resulting organization is depicted in Figure 4C, made up of clusters of processing elements. Three processing elements in a cluster node are used on the two edges and six processing elements in a cluster node make up the rest of the array. For example, C03/C07/C11, D02/D04/D06, D01/D03/D05, and C01/C05/C09 make up the four three cluster nodes. A single processing element E01 makes up the last processing element. Within a three or six cluster processor group local connectivity is enhanced through additional I/O ports or a multi-port storage area. The enhanced connectivity provides reduced communications distances that could not be easily obtained in the typical six neighborhood hexagonal processor array, Figure 4A. .il 5 In a similar manner to the method for scaling the multiple-folded square clustered processor mesh array, the multiple-folded hexagonal processor array can be scaled. A concentric ring of processors surrounding an original hexagonal processor array, Figure 5A PEs with asterisks, can be added to the multiple-folded hexagonal array by adding an expanded edge of processors to the triangular processor array, Figure 5B PEs with asterisks.

7 Further Research

Research into folded array processors is continuing. Examples using the diagonal-fold array processor for a parallel 2-D convolution algorithm for image processing multi-media applications and for a finite difference method of solving differential equations is presented in reference [5]. In other applications, we have applied the folded array principles specifically for the digital emulation of completely connected and multi-layer neural networks [7, 8] creating a scalable massively parallel machine organization. In addition, we have generalized the Multiple-Fold Clustered Processor Mesh Array processor elements and provided an example using the new organization in a bitonic sorting application [9].

8 Summary

The Multiple-Fold Clustered Processor Mesh Array represents a new machine organization obtained by first folding an array multiple times, second by clustering the processor elements, and third by providing improved connectivity among the clustered processor elements. The new organization achieves improved connectivity, reduced wiring complexity, and ease of scaling to larger arrays as compared to the original square or hexagonal array structures. This organization is potentially important to parallel algorithm execution in image processing, multi-media, and general purpose applications.

References

[1] R. J. Gove, W. Lee, Y. Kim, and T. Alexander, "Image Computing Requirements for the 1990s: from Multimedia to Medicine," Proceedings of the SPIE Vol.1444 - Image Capture, Formatting, and Display, pp. 318-333, 1991.

[2] K. E. Batcher, "Design of a Massively Parallel Processor," IEEE Transactions on Computers Vol. C-29, No.9, pp.836-840, September 1980.

[3] L. Uhr, *Multi-Computer Architectures for Artificial Intelligence.* New York, New York: John Wiley & Sons, chap. 8, p. 97, 1987.

[4] R. Cypher and J. L. C. Sanz, "SIMD Architectures and Algorithms for Image Processing and Computer Vision," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 37, No. 12, pp. 2158-2174, December 1989.

[5] G. G. Pechanek, J. G. Delgado-Frias, and S. Vassiliadis, "A Massively Parallel Diagonal-Fold Mesh Array Processor," IBM Corporation Technical Report - TR 01.C420, pp. 1-24, Endicott, New York, September 1992.

[6] G. G. Pechanek, J. G. Delgado-Frias, and S. Vassiliadis, "A Massively Parallel Diagonal-Fold Array Processor," Accepted for Publication: Proceedings of the 1993 International Conference on Application-Specific Array Processors, Venice, Italy: IEEE Computer Society Press, October 25-27, 1993.

[7] G. G. Pechanek, "PLAN: Pyramid Learning Architecture Neurocomputer," Ph.D. Dissertation, pp. 1-342, T. J. Watson School of Engineering and Applied Science at the State University of New York at Binghamton, 1991.

[8] . G. Pechanek, S. Vassiliadis, and J. G. Delgado-Frias, "Digital Neural Emulators Using Tree Accumulation and Communication Structures," IEEE Transactions on Neural Networks, pp. 934-950, November 1992.

[9] G. G. Pechanek, S. Vassiliadis, and J. G. Delgado-Frias, "Massively Parallel Multiple-Folded Clustered Processor Mesh Array," IBM Corporation Internal Technical Report - TR 29.1655, pp. 1-41, Research Triangle Park, North Carolina, May 1993.

PRIMARY CONTACT: Gerald G. Pechanek, Ph.D.
IBM Corporation
P.O. Box 12195
203 Silicon Drive, Research Triangle Park, N.C.
27709
(919) 543-4706 (O), (919) 254-6963 (FAX)
e-mail: GPECHANEK@VNET.IBM.COM

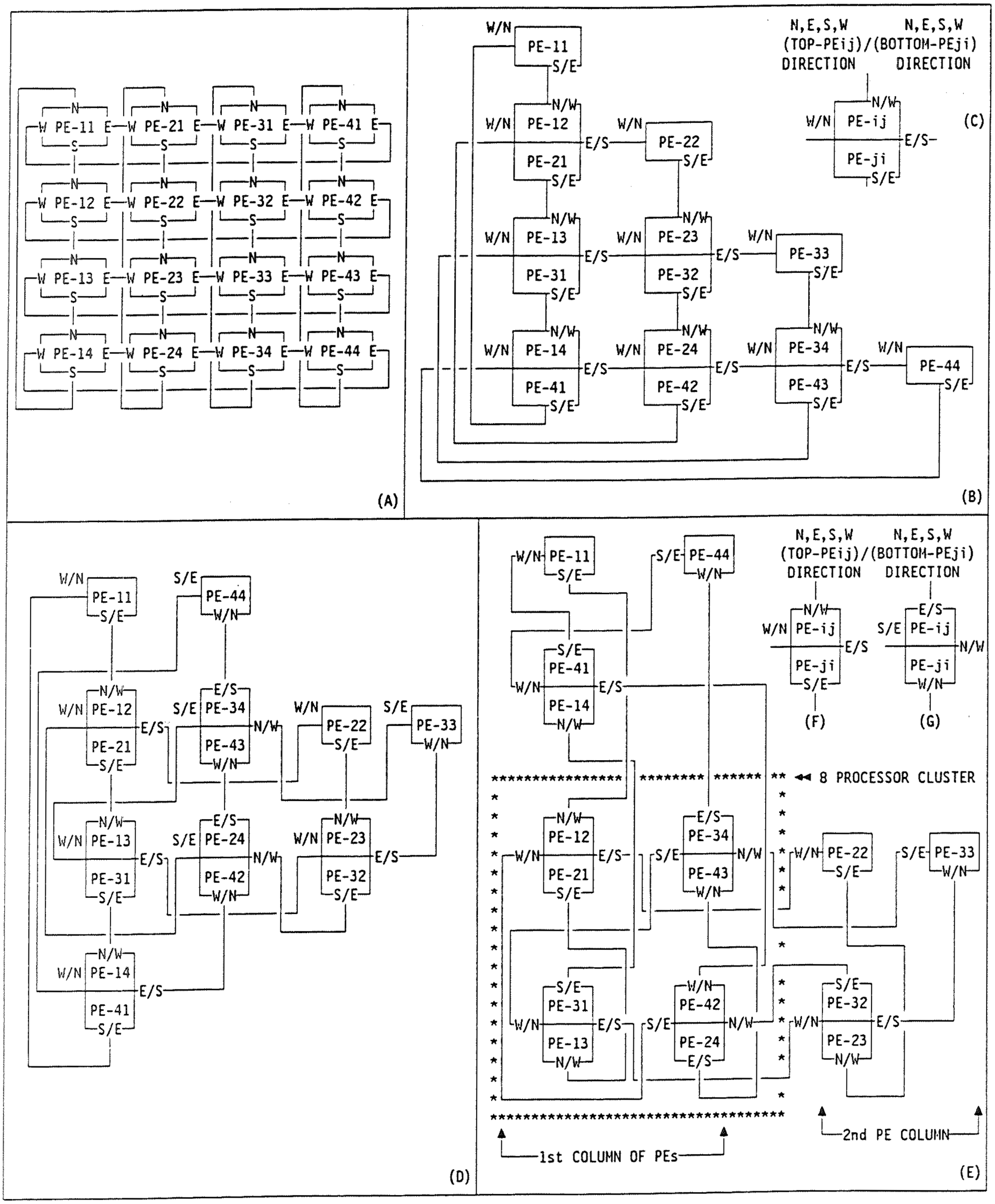


Figure 1. (A) 4 x 4 MESH, (B) 1st FOLD, (D) 2nd FOLD, (E) 3rd FOLD CLUSTERED PROCESSOR

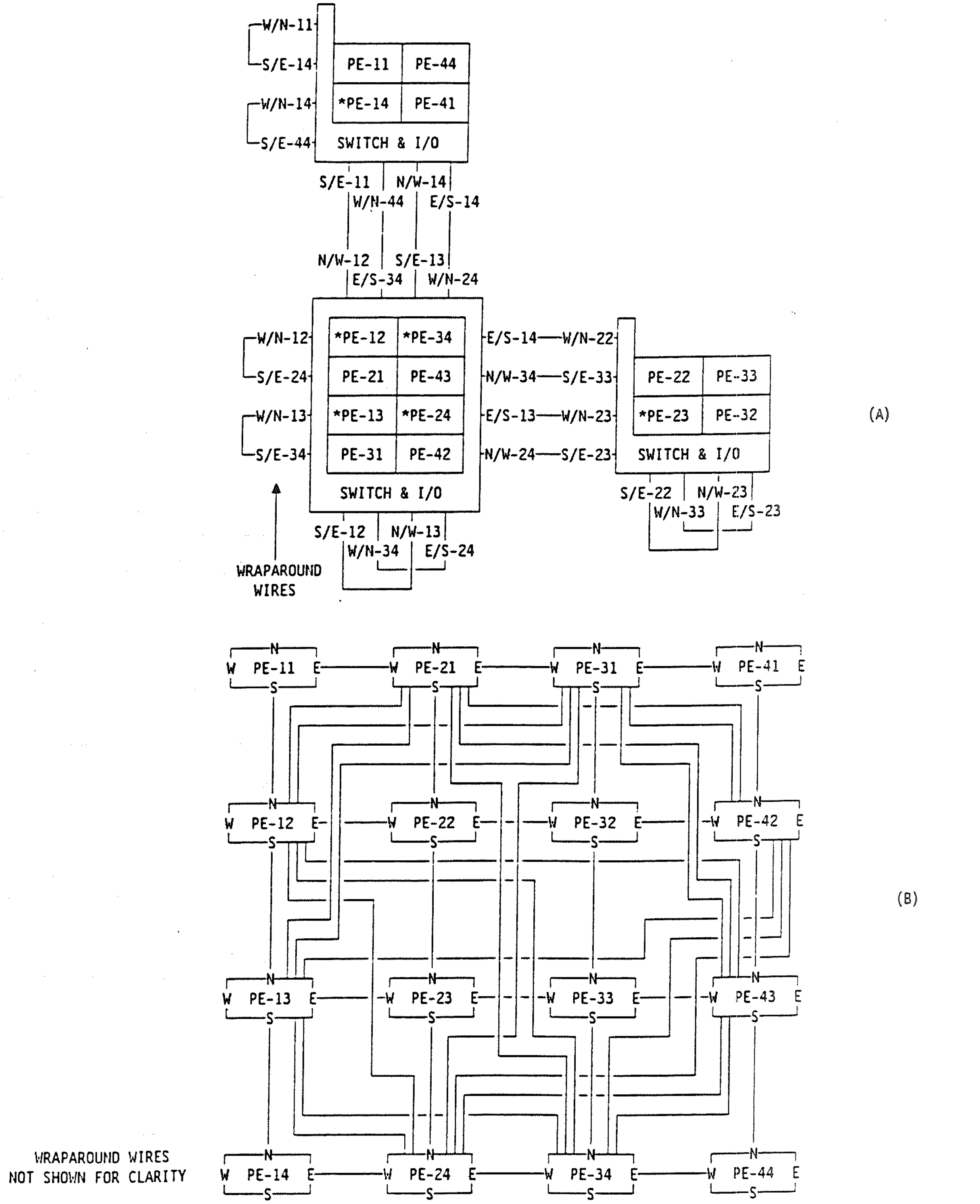


Figure 2. A) 4x4 THIRD FOLD CLUSTERED PROCESSOR MESH ARRAY and B) 4x4 SQUARE MESH WITH EQUIVALENT WIRING FOR THE ONE EIGHT PE CLUSTER

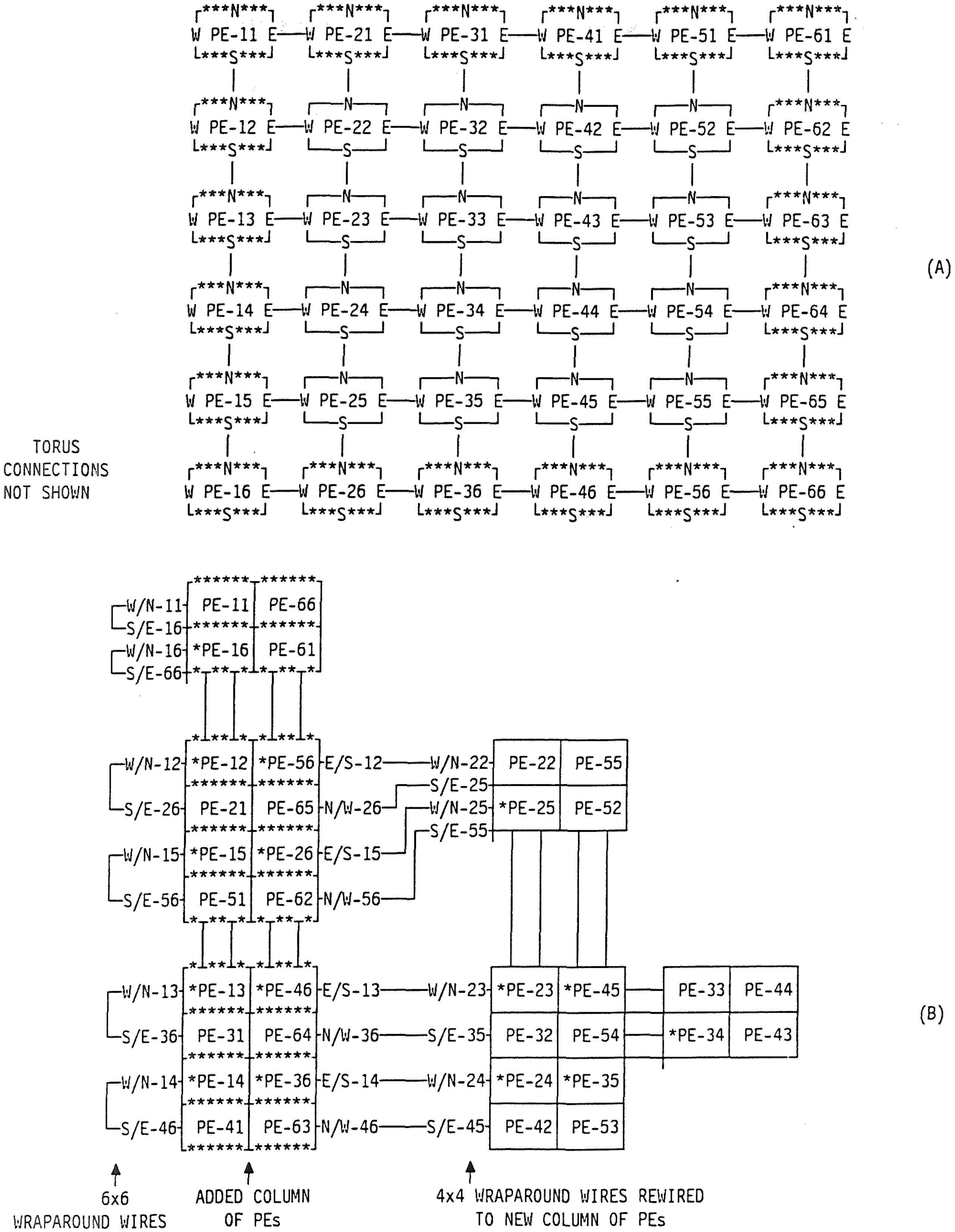

W PE-11 E — W PE-21 E — W PE-31 E — W PE-41 E — W PE-51 E — W PE-61 E
W PE-12 E — W PE-22 E — W PE-32 E — W PE-42 E — W PE-52 E — W PE-62 E
W PE-13 E — W PE-23 E — W PE-33 E — W PE-43 E — W PE-53 E — W PE-63 E
W PE-14 E — W PE-24 E — W PE-34 E — W PE-44 E — W PE-54 E — W PE-64 E
W PE-15 E — W PE-25 E — W PE-35 E — W PE-45 E — W PE-55 E — W PE-65 E
W PE-16 E — W PE-26 E — W PE-36 E — W PE-46 E — W PE-56 E — W PE-66 E
N
S
TORUS CONNECTIONS NOT SHOWN
(A)
W/N-11
S/E-16
W/N-16
S/E-66
PE-11
PE-66
*PE-16
PE-61
W/N-12
S/E-26
W/N-15
S/E-56
*PE-12
*PE-56
PE-21
PE-65
*PE-15
*PE-26
PE-51
PE-62
E/S-12
N/W-26
E/S-15
N/W-56
W/N-22
S/E-25
W/N-25
S/E-55
PE-22
PE-55
*PE-25
PE-52
W/N-13
S/E-36
W/N-14
S/E-46
*PE-13
*PE-46
PE-31
PE-64
*PE-14
*PE-36
PE-41
PE-63
E/S-13
N/W-36
E/S-14
N/W-46
W/N-23
S/E-35
W/N-24
S/E-45
*PE-23
*PE-45
PE-32
PE-54
*PE-24
*PE-35
PE-42
PE-53
PE-33
PE-44
*PE-34
PE-43
(B)
6x6 WRAPAROUND WIRES
ADDED COLUMN OF PEs
4x4 WRAPAROUND WIRES REWIRED TO NEW COLUMN OF PEs

Figure 3. A) 4x4 TO 6x6 MESH EXPANSION and B) 4x4 TO 6x6 MULTIPLE-FOLD MESH EXPANSION

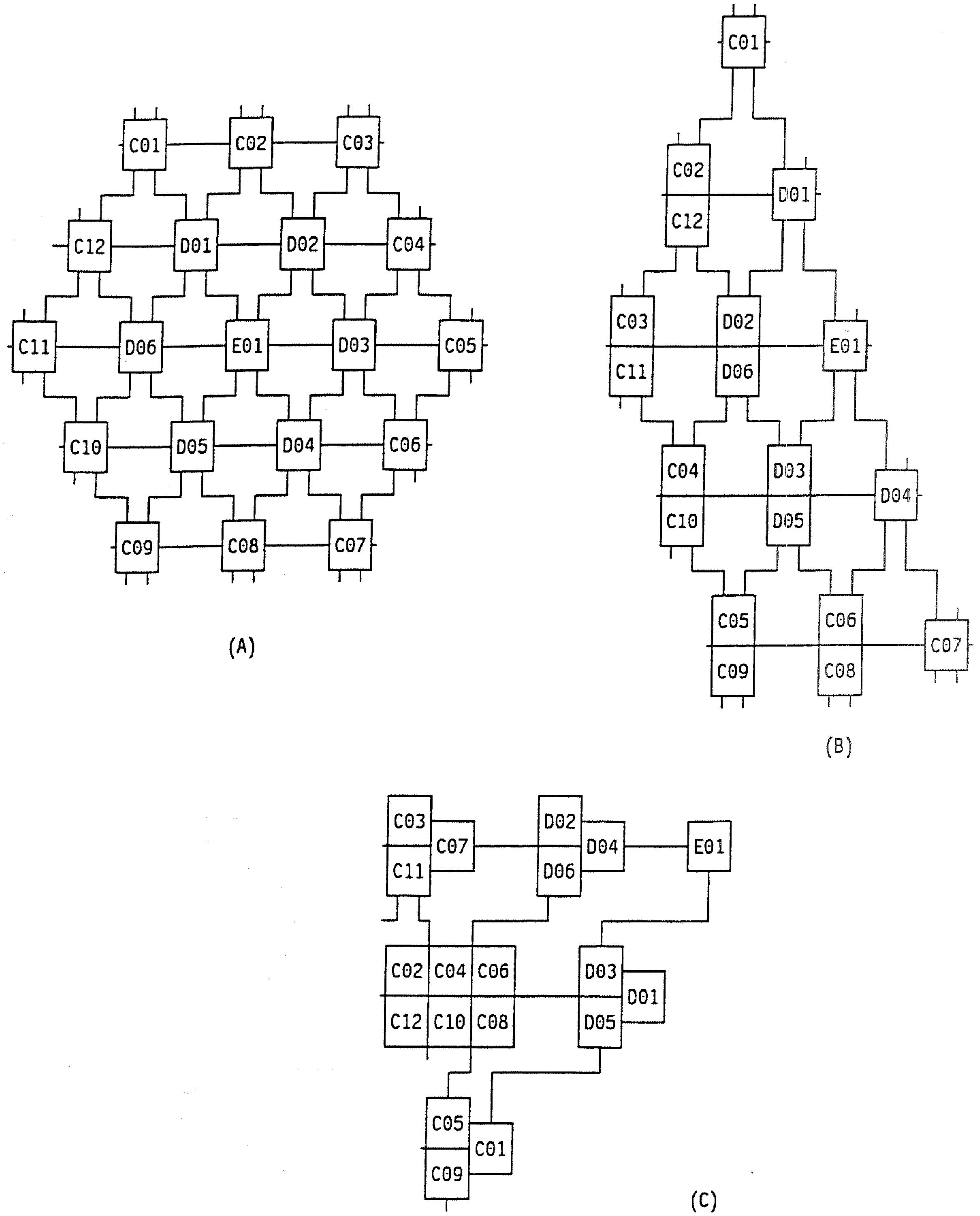


Figure 4. A) HEXAGONAL 6-PE NEIGHBORHOOD ARRAY, B) 1st FOLD, and C) 2nd & 3rd FOLD

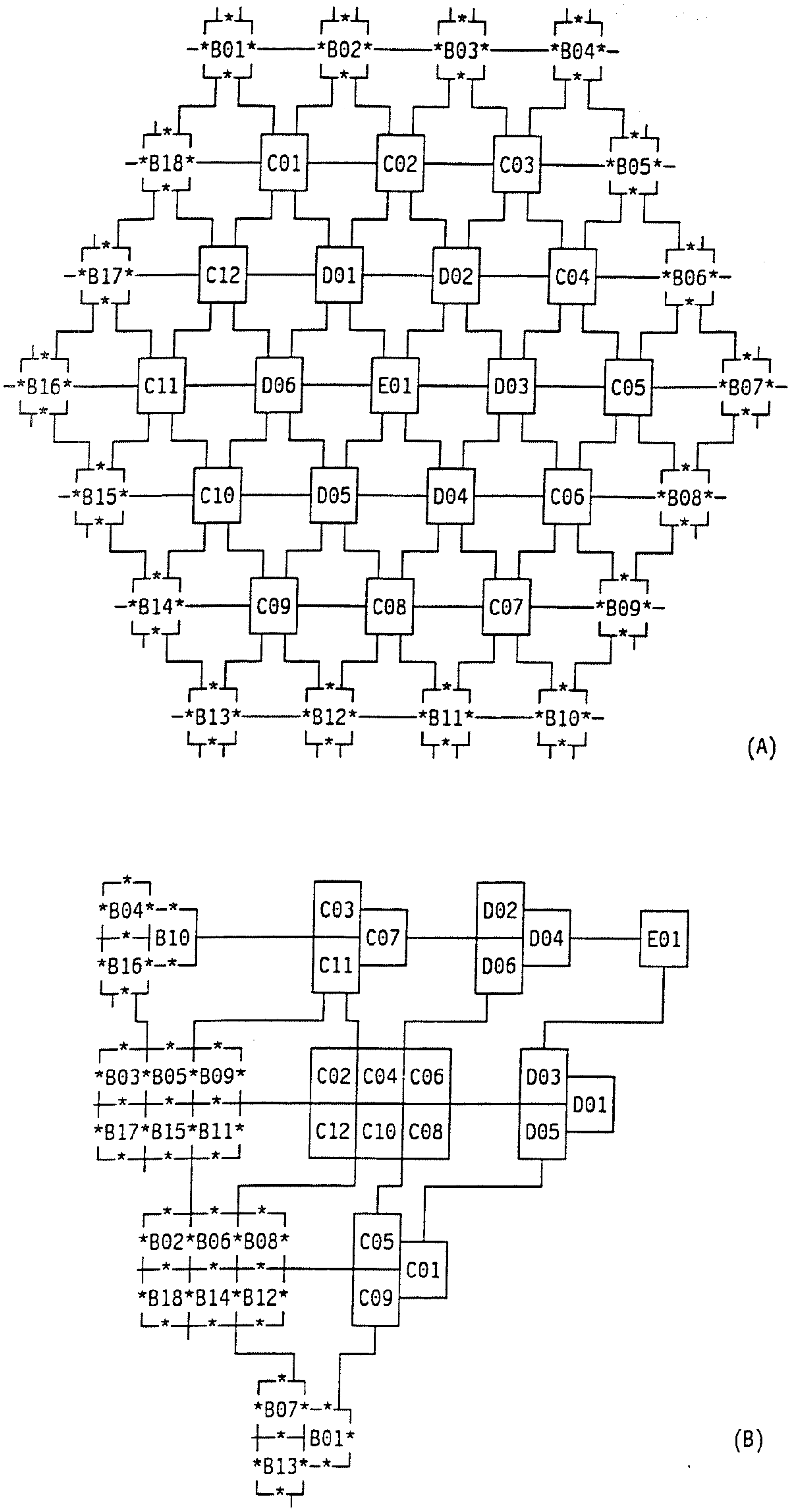


Figure 5. A) HEXAGONAL ARRAY EXPANSION and B) MULTIPLE-FOLD HEXAGONAL ARRAY EXPANSION

Practical Issues in Implementing Roving Spares in VLSI Systolic Arrays

Rajendra Bandekar and Amitava Majumdar
Department of Electrical Engineering
Southern Illinois University
Carbondale, IL 62901-6603
e-mail: majumdar@zeus.c-engr2.siu.edu

Abstract - **Roving spares as a methodology for on-line testing and reconfiguration of systolic arrays was first proposed in [5]. In this paper we address several unsolved problems in implementing roving spares. An algorithm for distinguishing between permanent faults and intermittent/transient faults is proposed. A methodology for determining different parameter values for this algorithm is presented. Results characterizing an optimal switching structure are presented. A judicious partitioning of information (for distributed and central processing) allows design of simple switch controllers for implementing roving spares operation as well as reconfiguration in the presence of failures. Testability and self-checking capability is incorporated into these controllers using a few extra gates.**

1 Introduction

Systolic arrays find use in a wide variety of applications ranging from simple sorting to matrix arithmetic and other complex signal processing computations [11]. Due to their versatility they can now be found in several critical and demanding environments such as space, medical and military applications.

One of the most stringent demands on circuits deployed in such environments is that of operational reliability. Whereas post-manufacturing (production) tests (carried out on all manufactured IC chips) are sufficient for ensuring quality of commercial circuits, in critical applications however, it is also necessary to monitor their performance during operation. Sources of run-time failures include alpha-particle bombardment, electron migration and general wearout of components. Monitoring of circuit performance during operation is called *on-line* testing. When a failure is detected using on-line testing, the machine can either be shut off for repairs or be reconfigured in such a way that it can operate correctly in the presence of the failure. The latter is called fault tolerance.

In this paper we present a methodology for on-line testing of a general class of systolic architectures. We also develop a switch design that allows automatic reconfiguration of the array to by-pass faulty modules. In both these tasks we exploit the regularity of the architecture, a feature common to all systolic arrays.

1.1 Past Effort

A universal characteristic of on-line testing and fault tolerance techniques is *redundancy* in the form of either 1) extra *hardware* (also called space redundancy) or 2) extra computational *time.*

Among methodologies employing hardware redundancy are dual-rail self-checking techniques [2] and modular redundancy techniques [9]. In dual-rail checking, hardware is added at a low level of circuitry while for modular redundancy high level components are added for testing purposes. An example of modular redundancy used in systolic arrays can be found in [17]. Time redundancy techniques offer a way of trading off extra hardware by increasing computation time [18, 16, 10, 13]. While space redundancy techniques incur high costs in terms of hardware overhead, time redundancy techniques suffer from speed and throughput limitations.

Another form of on-line testing, namely *algorithm based fault tolerance*, exploits characteristics of the particular computation being performed by a systolic array. Among these are general error detection/correction coding schemes [20]. More recently (see [1]) certain specialized techniques have been developed for a class of regular systolic algorithms based on concepts such as checksums. These are dependent on computations as well as data flow within the array.

After a fault has been detected using an on-line testing scheme, fault tolerance is usually achieved by by-passing the faulty module. This requires extra switching hardware. The simplest form of by-passing is logical removal of the whole row or column of modules containing the faulty one. Consequently there is some performance and/or reliability degradation. This degradation can be reduced by increasing the complexity of switches which allow by-passing of only the faulty module. A large amount of literature is available on reconfiguration startegies for systolic arrays. The interested reader may refer to [19, 6, 8].

1.2 Motivation and Our Work

In space, military and certain medical applications (which are the main areas of our focus) performance takes priority over hardware costs. Hence, modular redundancy is preferred over time redundancy techniques. The well known modular redundancy techniques do not offer much flexibility at assembly time for trading off hardware cost with reliability.

A generalization of modular redundancy that does allow this flexibility was first proposed by Breuer and Ismail in [4] and later adapted for systolic arrays by Breuer, Majumdar and Raghavendra in [5]. This technique, called *roving spares*, combines on-line testing with automatic reconfiguration. The switching structure allows a spare module to be logically moved from one physical module to another (hence the name roving spare). The same switching structure that makes *roving* possible also allows reconfiguration to by-pass a single faulty module. Depending on the number of spares available, the number of modules tested simultaneously can be varied[1]. Whereas the conceptual design of such a system was first proposed in [5] much of the implementational details were left open.

In this paper we address all practical issues in implementing roving spares and propose solutions for them. In the process we also generalize roving spares to a large class of systolic arrays. Section 2 describes the architecture and operation of roving spares in 2-dimensional systolic arrays. A simple algorithm for filtering out transient and intermittent failures is

[1] The well known 2-modular redundant system is a special case where the number of spares equals the number of active modules

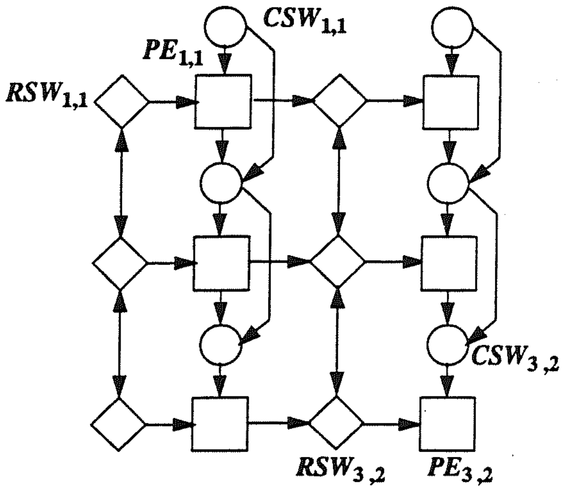


Figure 1: Architecture

presented in Section 3. Section 4 deals with switch design and characterization. Some concluding remarks are given in Section 5.
Note: Due to space limitations, all proofs of results are omitted from the text and may be found in [3].

2 Architecture and Operation of Roving Spares

2.1 Architecture

The systolic array as shown in figure 1 is a rectangular grid of $(m + 1) \times n$ processing elements (PEs). Each PE in this array, is connected to its immediate neighbours, namely $PE_{i+1,j}$, $PE_{i-1,j}$, $PE_{i,j+1}$ and $PE_{i,j-1}$. Data flow from $PE_{i,j}$ is unidirectional to $PE_{i+1,j}$ and to $PE_{i,j+1}$. A spare PE is associated with each column of PE's. Such architectures and data flow characteristics are very common in signal processing applications (e.g wave-front processing arrays etc.; see [12]).

Communication between PEs is established with the help of programmable switches and links. Associated with each $PE_{i,j}$ is a set of two programmable single-track switches, a row switch $RSW_{i,j}$ and a column switch $CSW_{i,j}$. Each switch can be configured by a switch control unit (SCU). The architechture of the switches and their control unit are described in the following section.

The SCU is designed to generate a faulty bit f for any *single stuck-at fault* (ssf) in the switch control ciruitry. Fault bit f is used to drive the current switch state to a failsafe state. It is proved that this failsafe state of the switch prevents erroneous routing of data in case of control logic failure, thus avoiding malfunction of the systolic array.

2.2 Operation

The spare PE in a column is switched, in a cyclic manner, to all the physical PE's in that column. The logical spare PE and a neighboring PE are tested together in *duplex* mode, i.e. they perform the same computations on the same input data and their outputs are checked by a comparator. In particular, $PE_{i,j}$ and $PE_{i+1,j}$ form a duplex pair for a test period of T clock cycles. Their inputs are tied together by the switching structure.

If the PE's in a duplex are non-faulty their outputs are same. Therefore, outputs from any one of the PE's in a duplex can be used in the next stage of computation. For the sake of proper operation we differentiate between the logical function of the two PE's. We assume that $PE_{i+1,j}$ (in general the PE with the higher row index) is the *spare* and $PE_{i,j}$ the *active* PE (consequently $PE_{1,j}$ can never be a logical spare). Outputs of only the active PE (in this case $PE_{i,j}$) are used in the next stage. As will be clear this assumption does not affect our results.

Outputs of the two duplex-mode *PEs* are compared in every clock cycle by a comparator circuitry. A mismatch indicates a fault in one of the two PE's in the duplex. Diagnosis can be done either off-line or on-line using sequential diagnosis techniques described in [5]. In this paper we assume an off-line diagnosis procedure. After the faulty PE is identified it is by-passed by the switching structure. This involves configuring the switches in a way that the faulty PE is a permanent logical spare[2] implying that its outputs are unused. When normal operation is resumed, the column containing the faulty PE operates without any on-line testing until it is replaced.

3 Fault Classification using Test Window Renewal

Faults can be classified as one of three types, namely 1) *permanent*, 2) *intermittent* and 3) *transient*. For practical purposes we distinguish between permanent and intermittent faults *only by the frequency* with which they manifest themselves as erroneous outputs. By this classification, both permanent and intermittent faults are caused by permanent defects in the circuit, the difference being that permanent faults have high detectability ($\geq \mu$) and intermittent faults have low detectability ($< \mu$). On the other hand transient faults may have high or low detectability depending on where in the circuit they occur. Their distinguishing characteristic is that they are shortlived.

We assume that lower level fault tolerance methods are used to tolerate intermittent and transient faults. Therefore, the purpose of roving spares is to detect permanent failures in PEs and *filter out* errors caused by intermittent and transient faults. Whereas filtering of intermittent faults requires estimation of its detectability, filtering of transient faults requires estimation of its lifetime.

We assume that the causes for transient errors (usually spurious charge due to α-particle bombardment) vanish within one clock cycle. Further, its effect may be stored in memory elements and remain for at most ν clock cycles (where ν is the sequential depth of the PE). Based on these assumptions we claim that the maximum lifetime of a transient fault is ν.

[2] In the rest of the paper the term spare PE is used to mean either a logical spare or a faulty PE and the meaning will be clear from the context.

```
Algorithm PE_test($T, M$)
window_num = 1; test_clock = 1;
While (test_clock $\leq T$)
{   If (NO_Error)  test_clock = test_clock +1;
    Else
    {  window_num = window_num +1;
       If (window_num $> M$)  break;      /* Permanent fault */
       Else  test_clock = 1;              /* Start new window */
    } }
If (window_num $< M$)  Switch to next duplex.
Else  Indicate PE failure.
```

Figure 2: Test-window renewal algorithm for filtering out intermittent and transient faults.

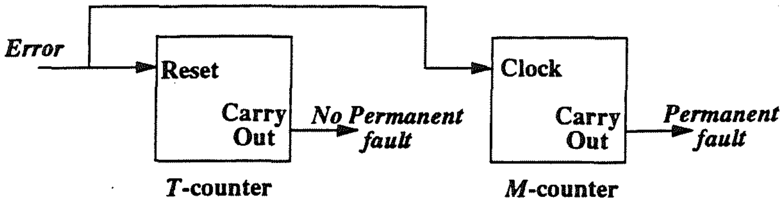


Figure 3: Circuit for implementing Algorithm *PE_test.*

We propose an algorithm (shown in Figure 2) that performs both filtering functions. The main idea is to test a PE in duplex mode for a window of T test clock cycles. If no error is detected, the spare switches to the next PE (starting a new duplex test). If an error was found in the first window (say in the kth test cycle) a new window of T cycles is started from the $k+1$th cycle[3]. A maximum of M such windows are allowed. If errors occur in M windows then the fault(s) causing them is(are) classified as permanent. If no error is detected in any of the windows the fault is considered either transient or intermittent and the two PEs in the duplex are declared *healthy.*

3.1 Determining Algorithm Parameters

Value of T is usually determined by fault coverage requirements. A suitable test length T^* can be found for a given circuit using methods outlined in [14, 15]. Given T^*, our aim is to classify permanent faults correctly. Let ϵ be a specified confidence value for correctly classifying permanent faults. For a fault with detectability p, the probability that it is classified as permanent is expressed as $f(p) = \left(1-(1-p)^T\right)^M$. We see that f is monotonically increasing in p. Therefore, among permanent faults, ones with detectability μ have the lowest probability of being classified as permanent for a given T and M. Letting $f(\mu) \geq \epsilon$ yields

[3] This is viewed as a window renewal process, hence the name test-window renewal.

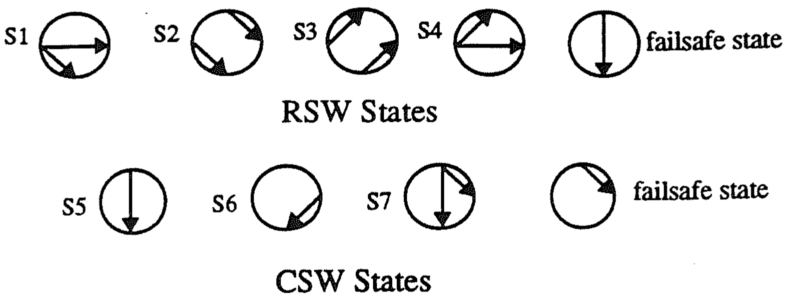


Figure 4: Row and column switch states

$$M^* \leq \left\lfloor \frac{\ln \epsilon}{\ln\left(1-(1-\mu)^{T^*}\right)} \right\rfloor . \tag{1}$$

However, to filter out transient faults we need $M^* \geq \nu + 1$. Thus, there may be cases where the set of constraints are not satisfied. In general, there exists a *region of feasibility* where specified values for μ, ν and ϵ can be satisfied by T^* and M^*. Other means of constraining these factors are also being currently studied.

Example 1 *A typical test length for medium sized circuits is* 10000 *vectors (clock cycles). Setting* $T^* = 10000$ *and* $\mu = 0.0007$, *we get* $M^* \leq 11$ *for confidence value* $\epsilon = 0.99$, *i.e. with 99% confidence a permanent fault will be classified correctly. If the sequential depth of the given circuit is* $\nu = 8$ *(a typical value), setting* $M^* = 9$ *satisfies all constraints.* □

4 Switch Characterization and Control

Reconfiguartion is necessary for two tasks, namely 1) to bypass a faulty *PE* and 2) during normal operation for routing data correctly for roving spare operation. The switching and routing structures developed for this purpose have low area usage and high reliability. The following results characterize our switching structure.

Theorem 1 *Four states, as shown in figure 4, are necessary and sufficient for RSW's.*

The proof of this theorem follows from the two propositions stated below. Let spare *PEs* in columns $j-1$ and j be in rows p and q respectively at a given time (i.e. $PE_{p,j-1}$ and $PE_{q,j}$ are spares). Further, let $RSW_{i,j}$ be a row switch that routes data from a column $j-1$ *PE* to a column j *PE*.

Proposition 1 *If* $q \leq p$ *then* $RSW_{i,j}$ *can either be in state S1 or state S2.* □

Proposition 2 *if* $q > q$ *then* $RSW_{i,j}$ *can either be in state S3 or state S4.* □

Proofs of these propositions can be found in [3].

Proof of Theorem 1: From Propositions 1 and 2 we see that four states are *necessary* for row re-routing. Furthermore, any topological distribution of the spare *PEs* in two adjacent columns maps to one of the above cases. Hence the four states are also *sufficient*. This proves theorem 1. □

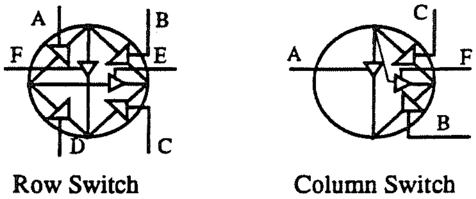


Figure 5: Design of row and column switches using tri-state buffers.

Theorem 2 *Three states, as shown in figure 4, are necessary and sufficient CSW's.*

Proof: (see [3]) □

A consequence of Theorems 1 and 2 is stated in the following corollary.

Corollary 1 *Single Track* RSWs *and* CSWs *are sufficient for row and column data routing.* □

The above results allow us to design switches that are simple and hence highly reliable. Row and column switches, made of tri-state buffers, are shown in Figure 5.

4.1 Switch Control Design

States of switches are determined by a controller associated with it. Two aspects of controllers are studied, namely 1) tradeoff between *distribution* of control and *complexity* of the circuit and 2) *testability* and *fail-safe operation* of controllers.

4.1.1 Row Switch Control Design

Three levels of control distribution are considered. At one extreme are completely centralized controllers which process information about the state of all switches associated with a column. The resulting circuit is highly complex and consequently has low reliability. Failure of such a controller results in malfunction of the entire systolic array. At the other extreme are completely distributed controllers, one such controller associated with each switch. However the number of states in such a controller (although much less than that required in a centralized controller) is still quite large. Given that $(m+1) \times n$ of these circuits are required, the overall area requirements and reliability may be unacceptable[4].

A feasible compromise between these extremes is achieved by judiciously partitioning the amount of information processed in a distributed manner and that processed by a central circuit. This results in a low complexity controller design. A block diagram of a partially distributed row switch control unit (P-*RSCU*) is shown in Figure 6a. Each $RSW_{i,j}$ has a P-$RSCU_{i,j}$ associated with it which processes most of the information required for normal operation. The rest of the information for a column j is processed centrally by a controller M_j.

Operation: It gets local inputs $n_{i,j-1}$ and $t_{i,j}$ from $PE_{i,j-1}$ and P-$RSCU_{i-1,j}$ respectively. It also gets a global input p from controller M_j. Input $n_{i,j-1}$ goes high if the spare is at

[4] A design of a completely distributed controller for row switches is given in [3]

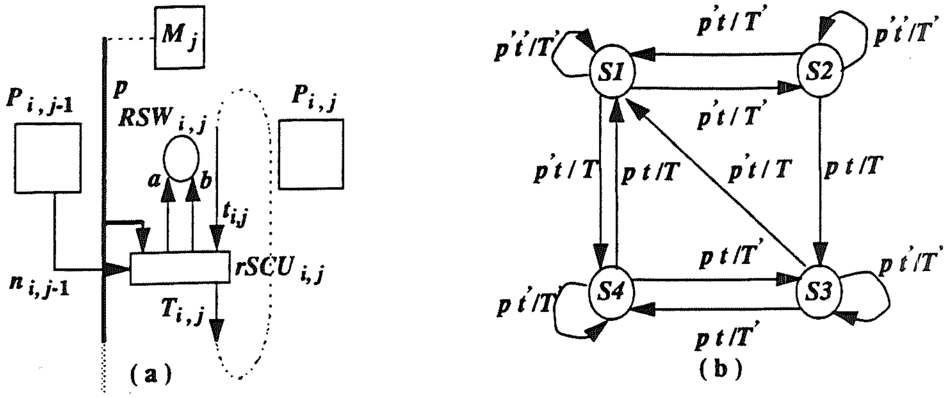


Figure 6: a) Row Switch Control Unit Block Diagram. b) State diagram of a partially distributed row switch controller.

$PE_{i,j-1}$ and moves to $PE_{i+1,j-1}$. The signal p indicates the relative position of spare PEs in columns j and $j-1$. p is high if and only if the row index of the spare in column $j-1$ is greater than the row index of the spare in column j. For $j=1$, p is always low.

Trigger output $T_{i,j}$ of P-$RSCU_{i,j}$ is the trigger input $t_{i+1,j}$ to P-$RSCU_{i+1,j}$. $T_{m+1,j}$ is fed back to $t_{1,j}$. When the logical spare moves from $PE_{i,j}$ to the next PE, $t_{i,j}$ goes high. Based on the current state of $RSW_{i,j}$ and the input $n_{i,j}$, P-$RSCU_{i,j}$ determines the next state of $RSW_{i,j}$ and sets $T_{i,j}=1$. A chain reaction follows till no more trigger signals are set. This indicates that $RSWs$ in column j have reached their stable states.

Outputs a and b of P-$RSCU_{i,j}$ determine the next state of $RSW_{i,j}$. The manner in which these signals are set and reset becomes clear from the state diagram of P-$RSCU$ shown in Figure 6b. We see that the global input (signal p) provides information required for the controller outputs to satisfy conditions of Propositions 1 and 2. Signals a and b are used as inputs to a decoder-encoder circuitry (Figure 7). Outputs of the encoder are used to control the tri-state buffers of the switches.

The circuit for a P-$RSCU$ is shown in Figure 7. Part A of the circuit implements the state diagram. Parts B and C are required to handle an exception where normal triggering is not enough to determine the states of all the switches.

4.1.2 Column Switch Control Unit

CSW control (unlike RSW control) is completely distributed. Each column switch $CSW_{i,j}$ has a control unit called T-$CCSU_{i,j}$, associated with it (see Figure 8a). The main component of this circuit is a decoder-encoder combination whose outputs control the tri-state buffers of a CSW (see Figure 8b).

Operation: Each T-$CCSU_{i,j}$ recieves input u, from $PE_{i,j}$ and input v, from $PE_{i-1,j}$. These signals are low if the corresponding PEs are *active* and high if they are spare. Input u to T-$CCSW_{1,j}$ is permanantly set high. Table 4.1.2 shows truth table of column switches[5].

[5] Both row switch controllers and column switch controllers have been designed using 3-*micron* technology. Plots and source files are indicated in [3].

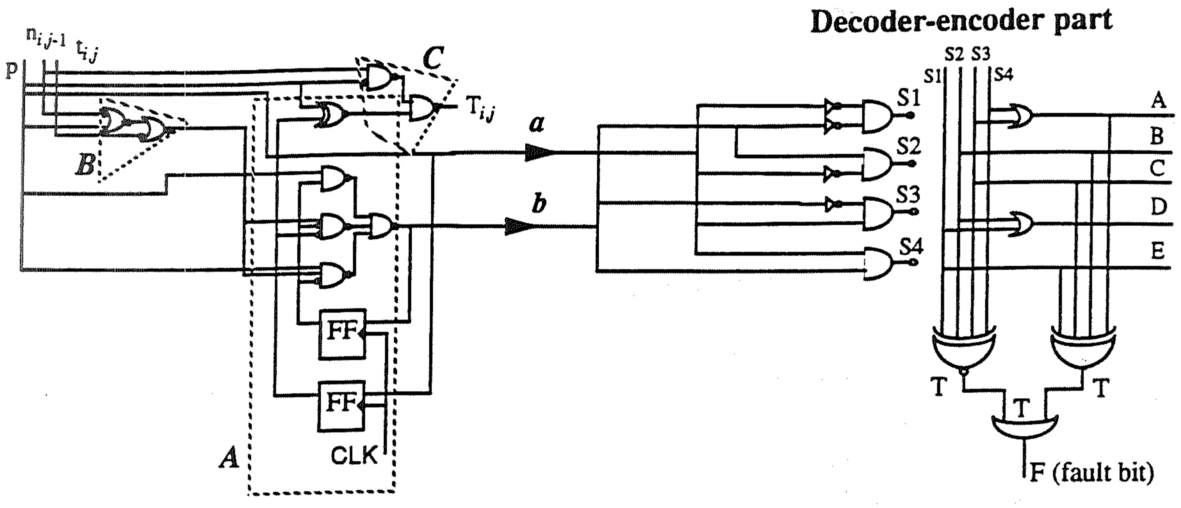
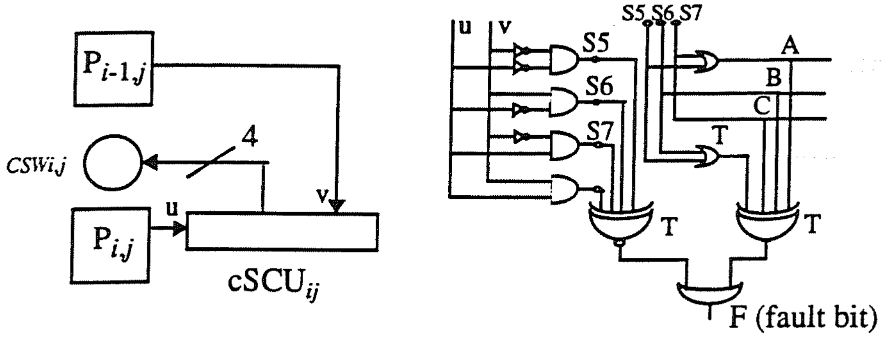


Figure 7: Row Switch Control Circuit



Column Switch Control Block Diagram

Decoder-Encoder circuit

Figure 8: a) Column Switch Control Unit Block Diagram. b) Column Switch Control Circuit.

$PE_{i-1,j}$ status (u)	$PE_{i,j}$ status (u)	Switch state
Active (0)	Active (0)	S5
Spare (1)	Active (0)	S6
Active (0)	Spare (1)	S7
Spare (1)	Spare (1)	Illegal

Table 1: Truth table for Column Switch Control

4.2 Testabilty and Failsafe Design

A testable design for RSCUs is achieved by adding a small amount hardware. Flip-flops $y1$ and $y2$ (Figure 7) are made scannable. All primary inputs (p, t, n) are made controllable and outputs (a, b, T) are made observable. Following fault collapsing, *FAN* [7] is used for test pattern generation. The complete stuck-at test set for this circuit is

$$\{11011, 10101, 00001, 10010, 10111, 01001\}.$$

The decoder-encoder circuits of RSW controllers and CSW controllers are made *self-checking* using a set of XOR, XNOR and OR gates (marked T) as shown in Figure 7. For any input combination only one of the outputs of the decoder (*S1,S2, S3, S4* for RSW controllers and *S5, S6, S7* for CSW controllers) is high. This condition is checked by the XNOR gate. Further, only two of the outputs (A through E in RSW controllers and A through D) of the encoder are high for any input combination. This is checked by the XOR gate.

Fault bit f is used to drive the corresponding switch (either RSW or CSW) to a failsafe state as shown in Figure 4. When $RSW_{i,j}$ fails $PE_{i,j-1}$ and $PE_{i,j}$ become *inaccessible*. Their functions are assumed by the spares in columns $j-1$ and j respectively and the array can resume operation. When $CSW_{i,j}$ fails only $PE_{i,j}$ becomes *inaccessible* and its function is assumed by the spare in column j. Correct data flow is maintained by *row* and *column* reconfiguration techniques.

5 Conclusion

We have addressed the problem of on-line testing and reconfiguration of systolic arrays using roving spares. Introduced first in [5], roving spares offers a generalization of modular redundancy techniques allowing flexibility in trading off reliability and hardware overhead.

In this paper we address several issues in implementing roving spares testing. We present a test window renewal algorithm (Figure 2) for distinguishing between permanent, intermittent and transient faults. Next we develop a switching structure and its control circuitry for carrying out roving spares operation as well as for reconfiguration in the presence of failures. Results presented in this paper ensure the minimum number of states required for roving spares testing. Further, by proper partitioning of control (i.e. partly distributed and partly centralized) we ensure high reliability of switch control circuits. Test vectors for complete stuck-at testing of switch controllers are provided. Self-checking capability is added to part of each switch controller ensuring fail-safe operation.

Future work in this area include evaluation of our testing algorithm in its ability to filter out intermittent and transient faults. Further, reliability and fault latency estimation of such systems also needs to be carried out. Work in these directions is already underway.

References

[1] J.A. Abraham, C-Y. Chen, P. Banerjee, W.K. Fuchs, S-Y. Kuo, and A.L.N. Reddy, "Fault tolerance techniques for sytolic arrays," *IEEE Computer*, 20(7):65–74, July 1987.

[2] Abramovici, M., M.A. Breuer and A.D. Friedman, Digital SystemsTesting and Testable Design, *Computer Science Press*, 1990.

[3] R. Bandekar, "Testing and Reconfiguration in Fault Tolerant Systolic Arrays," *Master's thesis*, Dept. of Elect. Engineering, Southern Illinois University, 1993.

[4] M.A. Breuer and A. Ismaeel, "Roving emulation as a fault detection mechanism," *IEEE Trans. on Comp.*, C-35(11):933–939, Nov. 1986.

[5] M.A. Breuer, A. Majumdar, and C.S. Raghavendra, "Fault tolerance and testing aspects of an architecture for a generalized sidelobe cancellor," In *Proc. ICCD*, 514-519, N.Y., Oct. 1988.

[6] J.A.B. Fortes and C.S. Raghavendra, "Gracefully degradable processor arrays," *IEEE Trans. on Comp.*, 15(1):37–46, Jan. 1982.

[7] Fujiwara, H. and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Trans. on Computers*, Vol. C-32, No. 12, Dec. 1983, pp. 1137-1144.

[8] R. Gupta, A. Zorat and I.V. Ramakrishna, "A Fault-Tolerant Multipipeline Architecture," In *Digest of 16th Ann. Int. Symp. on Fault Tolerant Computing Systems*, 350-355, 1986.

[9] B.W. Johnson, Design and Analysis of Fault Tolerant Digital Systems, *Addison Wesley*, 1989.

[10] B.W Johnson, "Fault-tolerant microprocessor based systems," *IEEE Micro*, 4(6):6–21, Dec. 1984.

[11] H.T. Kung, "Why systolic architectures?" *IEEE Computer*, 15(1):37–46, Jan. 1982.

[12] S.Y. Kung, VLSI Array Processors, *Prentice-Hall*, 1988.

[13] A. Majumdar, C.S. Raghavendra, and M.A. Breuer, "Fault tolerance in linear systolic arrays using time redundancy," *IEEE Trans. on Comp.*, 39(2):269–276, Feb. 1990.

[14] A. Majumdar and S. Sastry, "On the Distribution of Fault Coverage and Test Length in Random Testing of Combinational Circuits," *Proc. of 29th ACM/IEEE Design Automation Conf.*, pp. 341-346, 1992.

[15] A. Majumdar and S. Sastry, "Random-Test Length Estimation: Analysis and Techniques," *Tech. Rep. SIUC/DEE/TR-93-2*, Dept. of Electrical Engineering, Southern Illinois University-Carbondale, 1993.

[16] J.H. Patel and L.Y. Fung, "Concurrent error detection in ALUs by recomputing with shifted operands," *IEEE Transactions on Computers*, C-31(7):589–595, July 1982.

[17] R.K. Gulati and S.M. Reddy, "Concurrent error detection in vlsi array structures," In *Proc. ICCD*, 488–491, Oct. 1986.

[18] D.A. Reynolds and G. Metz, "Fault detecting capabilities of alternating logic," *IEEE Trans. on Computers*, c-27(12):1093–1098, Dec. 1978.

[19] R. Negrini, M.G. Sami, and R. Stefanelli, "Fault tolerance techniques for array structures used in supercomputing," *Computer*, 19(2):78–87, Feb. 1986.

[20] J.H. Wakerly, Error Detection Codes, Self Checking Circuits and Applications. *Elsevier-North Holland*, NewYork, 1978.

A Novel VLSI Processor Architecture for Supercomputing Arrays

N.Venkateswaran, S.Pattabiraman, R.Devanathan Ashraf Ahmed,
S.Venkataraman and N.Ganesh
Dept. of Computer Science and Engineering
Sri Venkateswara College of Engineering
Pennalur, Sriperumbudur - 602 105
Tamilnadu, INDIA

Abstract - **Design of the processor element for general purpose massively parallel supercomputing arrays is highly complex and cost ineffective. To overcome this, the architecture and organization of the functional units of the processor element should be such as to suit the diverse computational structures and simplify mapping of complex communication structures of different class of algorithms. This demands that the computation and communication structures of different class of algorithms be unified. While unifying the different communication structures is a difficult process, analysis of a wide class of algorithms reveals that their computation structures can be expressed in terms of basic IP, IP,OP,CM,R,SM & MAA operations. The execution of these operations is unified on the PAcube macro-cell array. [1] [2] [3].**

Based on this PAcube macro-cell array we present a novel processor element called GIPOP processor having dedicated functional units to perform the above operations. The architecture and organization of the these functional units are such to satisfy the two important criteria mentioned above. The structure of the macro-cell and the unification process has led to a very regular and simpler design of the GIPOP processor. Being designed on high performance mask programmable PAcube arrays the production cost of the GIPOP processor gets drastically reduced.

Index Terms - IP-Inner Product, OP-Outer Product, CM-Chain Multiplication, R - Reciprocal, SM - Scalar Multiplication, MAA - Massive array addition, PAcube - Programmable array of adders, Unification, GIPOP - Generalized Inner Product Outer Product.

1 Introduction

Here we propose and discuss certain important architectural and organizational features in the design of processor element meant for a general purpose massively parallel supercomputing array.

Functional Units: The functional units of the processor must be of higher complexity rather than simple Adder and Multiplier units. Presence of such complex functional units aids in mapping complex algorithms of large problem size. Deciding the types of functional units is a design criteria and should be based on the analysis of the computational structures of a wide class of algorithms. Also, it is enough that these functional units are of fixed point arithmetic type.

Organization: In mapping complex algorithms, it is preferred to take as much complexity as possible to within the processor. This involves assigning each processor a sub-graph of suitable complexity. While this reduces the communication complexity at the array level, it increases the processor complexity. Organization of the functional units therefore plays an important role in executing a combination of functions in an integrated manner. Maintaining the pipelining rate under such complex organization is however difficult and requires special hardware schemes, discussed later.

Parallelism and Pipelining: In executing different complex combination of functions in the processor, the organization of the functional units should be made suitable for dynamic reconfiguration. Dynamic reconfiguration of multifunctional units present two major problems i)matching the pipelining rate of different functional units and ii)managing the control complexity. This demands that the design of functional units be based on expandable macro-cell array in which the data flow of different functional units can be made identical.

Instruction Set: Instruction set design is a major criteria in fully utilizing the architectural capabilities of the processor. Branch and floating point instructions affects the pipelining rate. Such operations should be handled by the array controller, as this will greatly reduce the processor complexity. Removal of such complex instructions simplifies the control design of the processor.

Based on the above design criteria, we present a novel architecture called GIPOP processor. Section I deals with functional units of the GIPOP processor. In section II and III we discuss about the organization of these functional units and their level of pipelining and parallelism in solving complex functions. Section IV deals with instruction set of the GIPOP processor. The control design of the GIPOP processor is discussed in section V. In the final section we present a few examples of mapping on the GIPOP processor array.

2 GIPOP Processor Architecture

The GIPOP processor architecture is shown in Figure 1. **Functional Units**

IP and CM execution unit:
Analysis of a wide class of algorithms reveal that their computational structures can be expressed in terms of basic IP,OP,CM,SM,R and MAA operations. The OP and R operations can in turn be expressed as CM operation [2]. The execution of IP and CM can be unified by a process of massive array generation and reduction [3]. Unification means that the IP and CM operation has uniform dataflow on the macro- cell array and can therefore be executed in the same hardware structure. The GIPOP processor has therefore, a single functional unit for executing IP/CM operation. It can execute 4-bit 4-term IP / 4-bit 3-term CM operation.

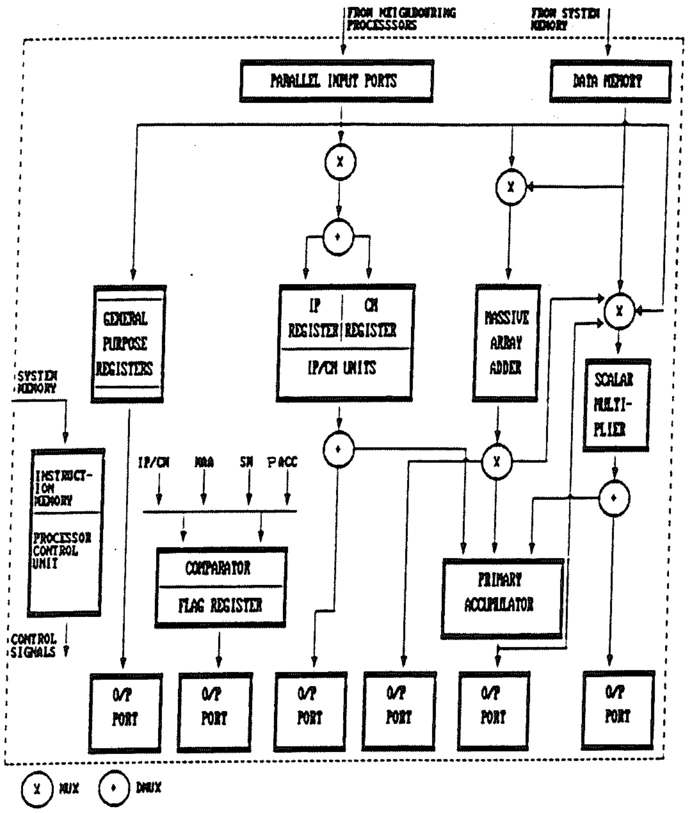


Figure 1: GIPOP Processor Architecture

Massive Array Adder:
There are several applications involving massive array addition. Conventional pipelined accumulation of numbers using two input adders proves highly inefficient in adding a very large array of numbers. GIPOP processor therefore includes a massive array adder unit. It can add eight 4-bit numbers simultaneously. Scalar Multiplier unit :

Operations of the type $S * \int_{i=1}^{n} K_i$
is another common function present in a majority of numeric algorithms. This if executed in IP/CM unit will be highly inefficient. The addition operation is performed in the MAA unit and a separate functional unit is provided in the GIPOP processor for performing scalar multiplication. The SM unit performs 4-bit by 7-bit(result of adding eight 4-bit numbers in the MAA unit) multiplication.

Primary Accumulator:
Problem size of the algorithm will be very much larger compared to the processor array size. In such cases serial operations at the array level will increase the communication complexity. The best approach is to have more serial operations in the processor and maintain maximum parallelism at the array level. For this purpose the GIPOP processor includes a special primary accumulator unit for this purpose. Moreover to accumulate operands of different weights, it has provision for appropriate shift. The shift logic is made hardwired instead of conventional clock controlled type to maintain the pipelining rate.

Comparator:
It can compare output form any of the above functional units and set the corresponding flags. The status flag will be read by the array controller to initiate the necessary branching operations.

General Purpose Registers:
After the initial loading of the array from the system memory, both the operands and the partial results will have to be pipelined through the array. Pipelining of the operands through the processor memory will increase the latency due to memory access time. Synchronization of the operand flow is a must to maintain the pipelining rate at the array level. Synchronization in the flow of partial results can be effected by introducing proper delay in the switch lattice of the processor array. But this will increase the switch complexity drastically. In order to avoid these difficulties, the GIPOP processor includes a set of general purpose registers primarily for pipelining data deep into the processor array.

The design of the above functional units are based on the PAcube macro-cell array [2] [3]. The architecture of the macro-cell and its expandability [4] with respect to wordlength and problem size highly simplifies the design implementation of the above functional units. Based on the grain size of the functional units, the wordlength and the problem size are partitioned accordingly. These functional units can be so mapped on the PAcube macro-cell array as to achieve the required pipelining rate [3]. In the present configuration the pipelining rate is fixed as four bits per cycle to reduce the processor complexity.

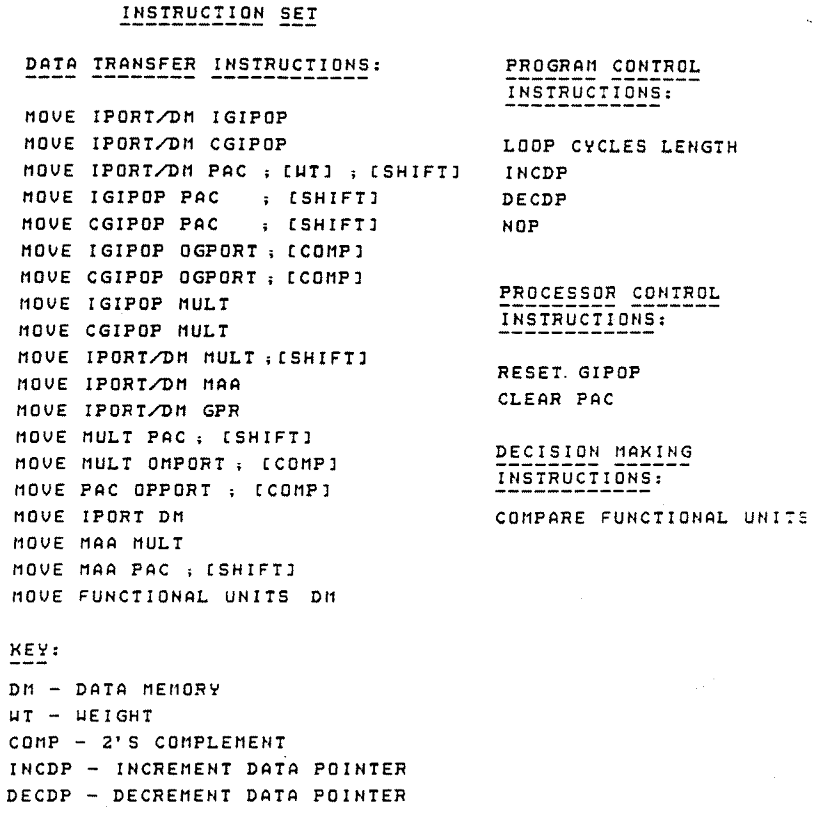
Floating point functional units:
All the above functional units are of fixed point arithmetic type only. In an array processor, each element of the array will be handling only a partial wordlength of the operands. Control operations like normalization in the different processors of the array has to be synchronized by the array controller. Leaving such operations to the array controller will not add much to its complexity, but will greatly simplify the processor design. Floating point operations can therefore be mapped at the array level, with no such special functional unit in the array.

Processor Memory and I/O Ports:
To facilitate parallel fetching of data required for the different functional units, the GIPOP processor memory is divided into three separate banks accessed independent of each other. These banks can be reconfigured to form different sizes as required. Each bank is a multiport memory and the number of ports depending upon the functional unit to which it is attached. The memory banks of all the functional units have 8 ports each except the SM unit which has 2 ports. Each port is of 4-bit length. The output port size of the functional units corresponds to their complexity.

3 Functional Units Organization

Organization of the functional units in executing a combination of functions plays an important role in increasing the performance of the processor at the array level. Maintaining the pipelining rate in such a complex organization requires handling of partial results from different functional units as and when they arrive. The control becomes even more difficult when the functional units involved are more complex than the conventional adder, multiplier, etc. This necessitates a simpler and regular data flow in the different functional units.

Being built on the expandable PAcube macro-cell array, the various functional units of the GIPOP processor has a regular data flow among them. Hence they can be easily integrated to operate on partial result outputs from different functional units in executing a combination of these functions (Figure 3). In such a complex organization, the multiple hardwired shift feature of the primary accumulator helps in maintaining the pipelined data flow.

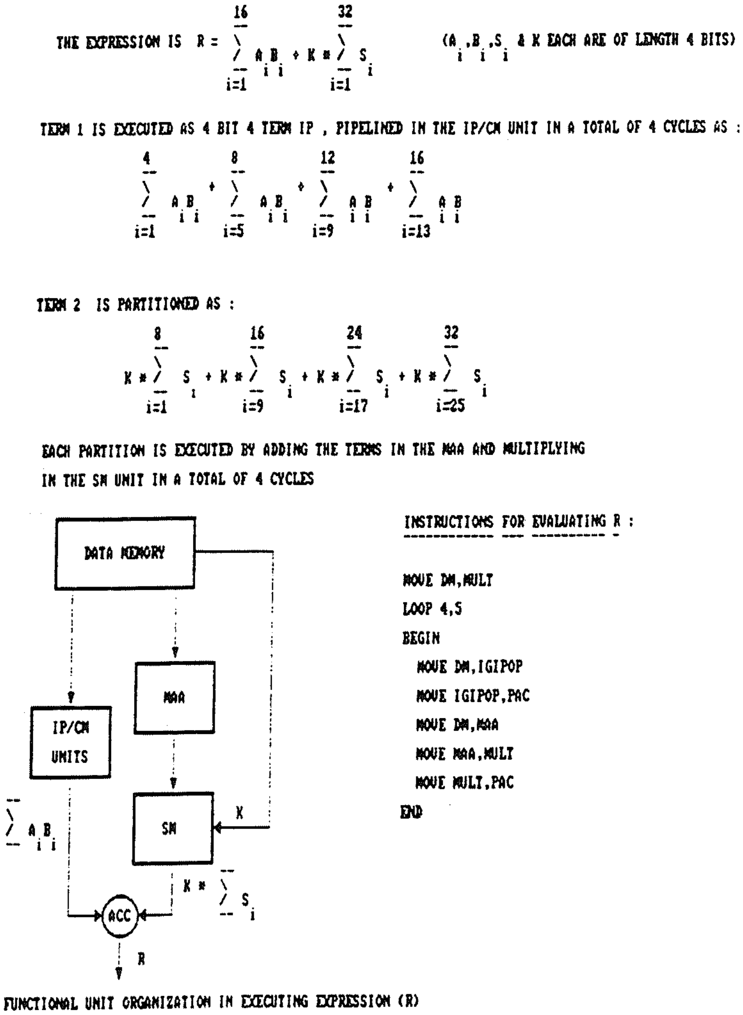


```
INSTRUCTION SET

DATA TRANSFER INSTRUCTIONS:

MOVE IPORT/DM IGIPOP
MOVE IPORT/DM CGIPOP
MOVE IPORT/DM PAC ; [WT] ; [SHIFT]
MOVE IGIPOP PAC    ; [SHIFT]
MOVE CGIPOP PAC    ; [SHIFT]
MOVE IGIPOP OGPORT ; [COMP]
MOVE CGIPOP OGPORT ; [COMP]
MOVE IGIPOP MULT
MOVE CGIPOP MULT
MOVE IPORT/DM MULT ;[SHIFT]
MOVE IPORT/DM MAA
MOVE IPORT/DM GPR
MOVE MULT PAC ; [SHIFT]
MOVE MULT OMPORT ; [COMP]
MOVE PAC OPPORT ; [COMP]
MOVE IPORT DM
MOVE MAA MULT
MOVE MAA PAC ; [SHIFT]
MOVE FUNCTIONAL UNITS  DM

KEY:

DM - DATA MEMORY
WT - WEIGHT
COMP - 2'S COMPLEMENT
INCDP - INCREMENT DATA POINTER
DECDP - DECREMENT DATA POINTER

PROGRAM CONTROL
INSTRUCTIONS:

LOOP CYCLES LENGTH
INCDP
DECDP
NOP

PROCESSOR CONTROL
INSTRUCTIONS:

RESET. GIPOP
CLEAR PAC

DECISION MAKING
INSTRUCTIONS:

COMPARE FUNCTIONAL UNITS
```

Figure 2: GIPOP Processor Instruction Set

4 Level of Pipelining and Parallelism

The different functional units of the GIPOP processor are designed to have the same pipelining rate. This is highly effective when the functional units are operated in an integrated manner. This is in contrast to the conventional processor architectures, where the standard design of different functional units cannot be easily matched to have a uniform pipelining rate. All the functional units of the processor can operate in parallel leading to very high efficiency. For example the pipelined outputs from the IP/CM and SM units can get added in the primary accumulator simultaneously. Here the efficient design of the accumulator aids in maintaining the pipelining rate.



THE EXPRESSION IS $R = \sum_{i=1}^{16} A_i B_i + K * \sum_{i=1}^{32} S_i$ (A_i, B_i, S_i & K EACH ARE OF LENGTH 4 BITS)

TERM 1 IS EXECUTED AS 4 BIT 4 TERM IP , PIPELINED IN THE IP/CM UNIT IN A TOTAL OF 4 CYCLES AS :

$$\sum_{i=1}^{4} A_i B_i + \sum_{i=5}^{8} A_i B_i + \sum_{i=9}^{12} A_i B_i + \sum_{i=13}^{16} A_i B_i$$

TERM 2 IS PARTITIONED AS :

$$K * \sum_{i=1}^{8} S_i + K * \sum_{i=9}^{16} S_i + K * \sum_{i=17}^{24} S_i + K * \sum_{i=25}^{32} S_i$$

EACH PARTITION IS EXECUTED BY ADDING THE TERMS IN THE MAA AND MULTIPLYING IN THE SM UNIT IN A TOTAL OF 4 CYCLES

INSTRUCTIONS FOR EVALUATING R :

```
MOVE DM,MULT
LOOP 4,5
BEGIN
  MOVE DM,IGIPOP
  MOVE IGIPOP,PAC
  MOVE DM,MAA
  MOVE MAA,MULT
  MOVE MULT,PAC
END
```

FUNCTIONAL UNIT ORGANIZATION IN EXECUTING EXPRESSION (R)

Figure 3: Sample execution of a complex expression in the GIPOP processor

5 GIPOP Processor Instruction Set

The GIPOP processor instruction set is given in Figure 2. This has certain unique features:

Absence of Arithmetic Instructions: This is achieved by having identical data flow for the execution of different operations, thereby requiring just moving of data to the appropriate unit. Thereafter the data flow is taken care by a controller common to all the operations. This controller is explained later.

Absence of Branch Instructions: In an array processor where processors handle partial wordlengths, the result of any control operations in the processors has to be synchronized by the array controller. The resulting branching operation can be converted as a problem of routing to different target processors at the array level. This means that the different control flow sequences in the algorithm forms different communication structures in the processor array.

Data Transfer Instructions:
The general format of the data transfer instructions is

MOVE source,destination [shift] [comp],

where the contents of the square brackets are optional. The [shift] parameter specifies the number of 4-bit shifts required and the [comp] parameter specifies whether the normal or two's complement output is required. The shift parameter is necessary for specifying the relative weight of the different operands being accumulated.

Program Control Instructions:
The format of the loop instruction is

LOOP cycles,length

The cycles refers to the number of times the loop to be performed and the length specifies the number of instructions to be included in the loop.

INC, DP and DEC DP instructions are required for changing the data pointer value by the number of bytes for accessing common data and avoiding duplicate storage within the processor.

Process Control Instructions:
This includes instructions like

RESET GIPOP and CLEAR PAC.

The instruction RESET GIPOP resets the input registers og the IP/CM unit when a new operation is started or when the pipeline has to be froze. The instruction CLEAR PAC clears the accumulator register when a new operand set is to be pipelined.

GIPOP Processor Instruction Set Coding:
The GIPOP processor uses a fixed length instruction format. Though this may seem to be inefficient with regard to storage it is not really so because the problem of mapping takes care that all the functional units are utilized to the best. This means that but for a few process control instructions, almost all possible parallel operations are initiated always. Also decoding of instructions will become simpler with fixed length format, as the decoding of different operations and generating corresponding control sequences can be done in parallel. The instruction length of the GIPOP processor is 24 bits.

6 Control Design

The GIPOP Processor array has an hierarchical level of control units. Such an hierarchical control makes the functioning of the system at any of the three levels (the Macro-cell level, Processor level and Array level) largely independent of system processes at other level. This eliminates the problems that occur due to globalization of control in an array processing system. The three levels of control are explained below:

Cell Control Unit (CCU): The CCU, lowest in the control hierarchy provides timing signals to the GIPOP macro-cell array. All the functional units of the processor are built on the basic PAcube macro-cell array and hence have a uniform data flow among them. Because of this the control sequences required for the execution of different functions, remain the same. Hence, the CCU's task is only to pipeline data through the input and output registers of the PAcube macro-cell. The CCU communicates with next control unit in the hierarchy, the PCU, to initiate and terminate respective operations.

Processor Control Unit (PCU): The job of the PCU is to execute the program stored in the instruction memory of the processor. The PCU is a two-stage pipelined unit (fetch and execution stage) with capacity to execute one instruction per clock cycle. The decoding of the instruction is highly simplified by having a fixed length format. Groups of fields fixed for different functional units are decoded simultaneously. The PCU most of the time has to do a simple data transfer operation to the different functional units. Thereafter the execution of the function (dataflow) is taken care by the CCU.

Just as the CCU receives initiation and termination signals from the PCU, so too this control unit receives initiation and termination signals from the next level of control namely the Array Control Unit. The array controller is usually a powerful general purpose computer. For executing different algorithms separate compilers have to be written. Using these compilers the array controller programs the processor array for executing these algorithms.

7 GIPOP Processor Array

Mapping algorithms onto the GIPOP processor involves the following steps.

1. Algorithm partitioning and expressing each functional partition in terms of GIPOP equations.

2. Wordlength and Problem size partitioning in terms of the processor grain size.

3. Assignment of partitions to the processors in the array, programming the processor.

4. Programming the array control unit.

We present below a few mapping examples.

Inner Product:
Consider an example of mapping 16-bit 16-term Inner Product on the GIPOP processor array. Decomposing the given problem in terms of the processor grain size (taken here as 8-bit 8 term) gives a total of 8 partitions. With 100% parallelism, each partition is assigned to a processor. The partial result outputs from the different processors get accumulated to give the final result. The cycle wise data flow is shown in Figure 4.
Fast Fourier Transform (FFT):
The FFT problem [4] is defined as follows.
Let $s(k), k = 0, 1, ..., M - 1$ be M samples of a time function. The discrete fourier transform of $s(k)$ is defined to be the discrete function $x(j), j = 0, 1, .., M - 1$ where

$$x(j) = \int_{k=0}^{M-1} s(k).\ Wjkj = 0, 1, ..., M - 1$$

$IP = \sum_{i=1}^{16} A_I * B_I$, where each A_I AND B_I are 16 bits length.

By wordlength partitioning,

$IP = \sum_{i=1}^{16} (A_{I,1} + A_{I,0}) * (B_{I,1} + B_{I,0})$, where As and Bs are 8 bits.

$= \sum_{i=1}^{16} A_{I,0} * B_{I,0} + \sum_{i=1}^{16} A_{I,1} * B_{I,0} + \sum_{i=1}^{16} A_{I,0} * B_{I,1} + \sum_{i=1}^{16} A_{I,1} * B_{I,1}$

$= IP_1 + IP_2 + IP_3 + IP_4$

With problem size partitioning of each IPs,

$IP = IP_{1,1} + IP_{1,2} + IP_{2,1} + IP_{2,2} + IP_{3,1} + IP_{3,2} + IP_{4,1} + IP_{4,2}$

where each $IP_{I,J}$ are weighted 8-bit, 8-term Inner Product function.

```
DATA FLOW  DETAILS :  SM(16,1,1)      BL(1,1 ¦ 1,2 , C2)
                      SM(16,1,2)      BL(2,1 ¦ 1,2 , C2)
                      SM(24,1,3)      BL(1,2 ¦ 1,3 , C4)
                      SM(24,1,4)      BL(2,2 ¦ 1,3 , C4)
                      SM(24,2,1)      BL(1,3 ¦ 1,5 , C6)
                      SM(24,2,2)      BL(2,3 ¦ 1,5 , C6)
                      SM(32,2,3)      BL(1,4 ¦ 1,5 , C8)
                      SM(32,2,4)      BL(2,4 ¦ 1,5 , C8)
```

SYNTAX : SM(16,1,1) Loading from System Memory, operand of weight 16 to processor in row 1 and column 1.

BL(1,1 ¦ 1,2 , C2) - Transfer of data from processor (1,1) to processor (1,2) in cycle 2.

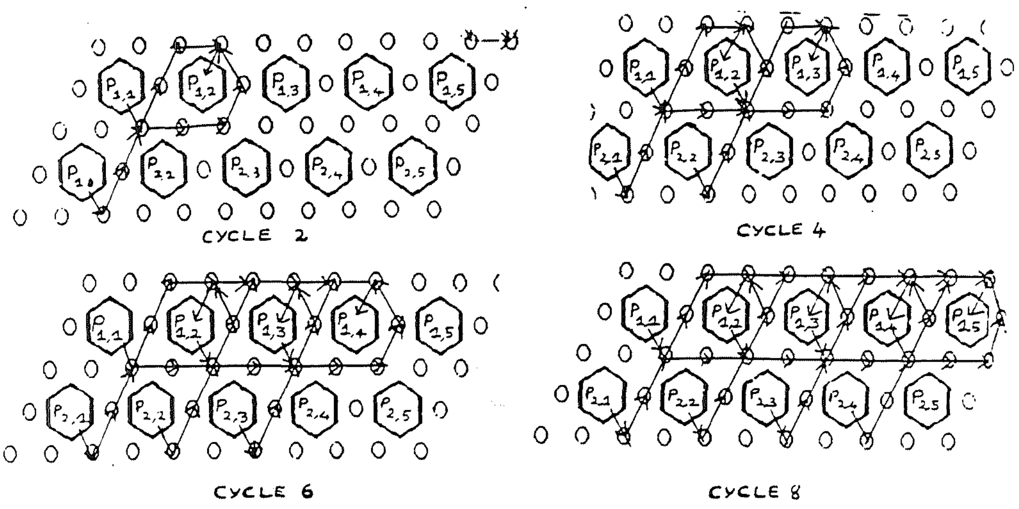


Figure 4: 16 bit 16 term Inner Product Mapping on GIPOP processor array

and

$$W = e2\pi i/M \ \ and \ \ i = \sqrt{-1}$$

The equation can be expressed in matrix format. Based on the mapping strategy discussed above, the FFT problem can be expressed in terms of the GIPOP functions. The exact decomposition process is not presented here. The cycle wise data flow for radix 2 4 point 8 bit FFT is shown in Figure. 5. Here the processor grain size is 8 bit 2 term IP function.

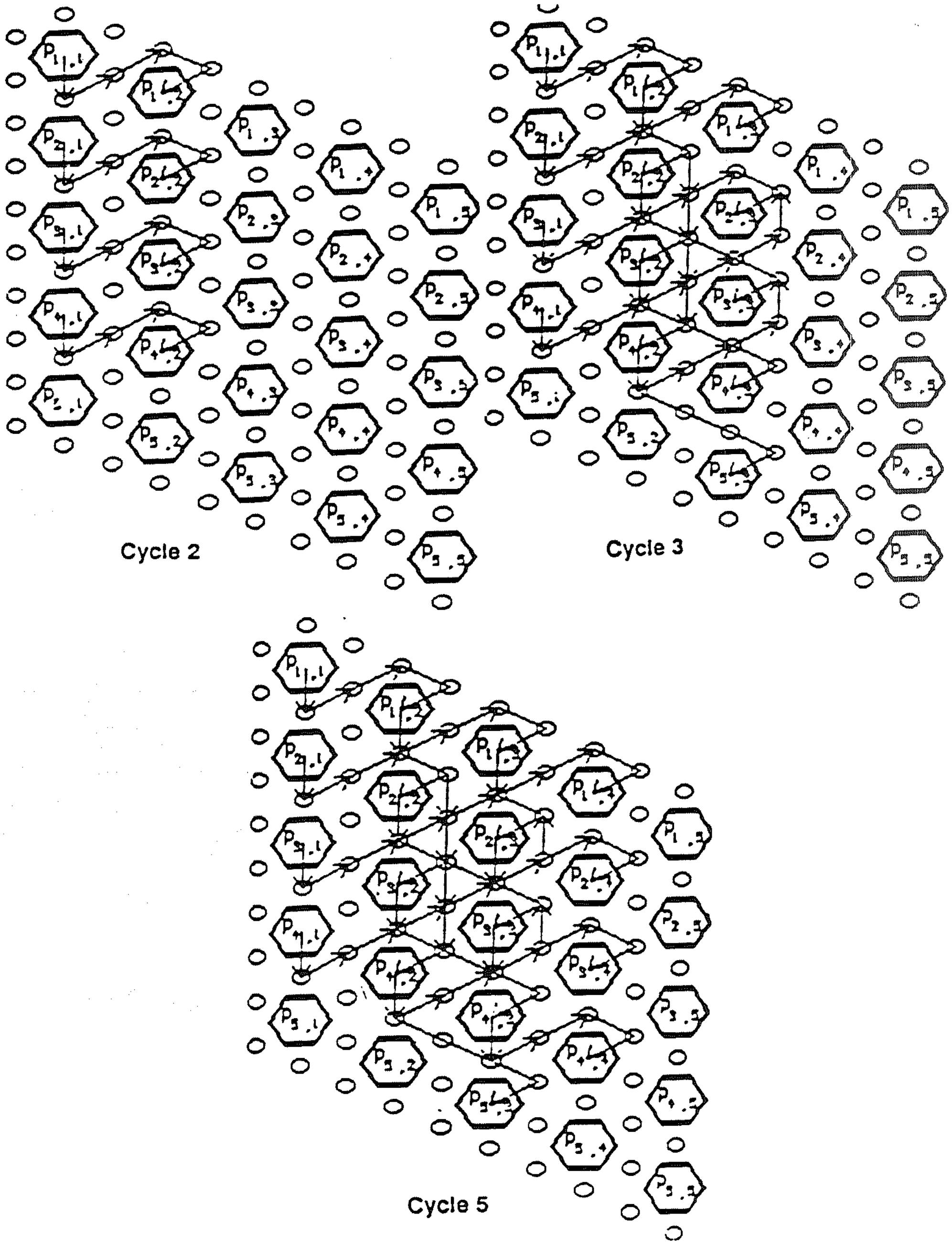


Figure 5: Mapping of radix 2 4 point 8 bit FFT mapping on GIPOP processor array

8 Conclusion

In this paper we have presented a novel VLSI processor architecture called GIPOP processor specially meant for massively parallel supercomputing arrays. The processor with its unique architectural and organizational features proves highly efficient in mapping complex algorithms of large problem size. Designed on the PAcube macro-cell array, the GIPOP processor has a simple and regular structure and highly cost effective. A simulator has been developed to verify the execution of complex operations of the GIPOP processor. Work being done includes mapping of non-numeric algorithms with complex control flow operations and floating point Inner Product and Chain Multiplication operations. Also an array level simulator is being developed to verify the GIPOP mapping strategy.

References

[1] N.Venkateswaran et.al., "PAcube VLSI Arrays for SuperComputers", Proceedings of the First International Conference on SuperComputing, conducted by IEEE computer Society and Pentagon, Florida, December 1985.

[2] N. Venkateswaran et al., "Special Purpose Silicon Compiler for Designing Supercomputing VLSI systems", 3rd NASA Symposium on VLSI Design, SERC, University of Idaho, Moscow, Idaho, USA, Oct. 1991.

[3] N.Venkateswaran et al., " A Cost Effective Methodology for Designing Massively Parallel VLSI Functional Units", submitted to FIFTH NASA Symposium on VLSI Design, Albuquerque, University of New Mexico.

[4] Lawrence R.Rabiner and Bernard Gold, "Theory and application of Digital Signal Processing", Prentice Hall.

omit

Session 9
Synthesis I

Phil Windley

Synthesis Algorithm of VLSI Multipliers for ASIC

O. H. Chua and A. G. Eldin
Electrical Engineering Department
Toledo OH 43606

Abstract - **Multipliers are critical sub-blocks in ASICs design especially for digital signal processing and communications applications. A flexible multiplier synthesis tool is developed which is capable of generating multiplier blocks for word size in the range of 4 to 256 bits. A comparison of existing multiplier algorithms is made in terms of speed, silicon area and suitability for automated synthesis and verification of its VLSI implementation. The algorithm divides the range of supported word sizes into sub-ranges, and provides each sub-range with a specific multiplier architecture for optimal speed and area. The algorithm of the synthesis tool and the multiplier architectures are presented. Circuit implementation and the automated synthesis methodology are discussed.**

1 Introduction

Multipliers are very frequently used functional blocks in ASIC's. Many high performance multiplier algorithms have been reported [1]-[5]. Each of these multiplier architectures generally achieve high performance for a relatively limited word size range. In many ASICs applications, the word size (in bits) is a design parameter that can be used to trade accuracy for smaller size of hardware and higher operating speed. Another important factor that determines which multiplier architecture to use is the designs speed and silicon area requirements. Therefore, an efficient and highly flexible multiplier generator is required.

To automate the generation of multipliers for ASIC's, there is a need to develop an is an algorithm which can select the most efficient multiplier architecture for a given word length and design priority. A synthesis tool is developed to automate the generation of multiplier blocks for word lengths in the range of 4 to 256 bits. The synthesis algorithm divides the supported range of the word lengths into smaller sub-ranges (4-12 bits, 13-64 bits and 65-256 bits). In this paper, several multiplier architectures are investigated and compared. The results of comparison in terms of speed and silicon area for those multipliers are reported in detail. The synthesis algorithm is discussed and the reason for selecting a specific multiplier architecture for each range is explained. The architecture and circuit design of each selected multiplier are discussed. Finally, the implementation and verification of the synthesis algorithm are presented.

2 Comparison of Multiplication Architectures

Several multiplier architectures have been investigated and comparisons are made based on their speed, silicon area and suitability for VLSI implementation. The architectures are listed in Table 1 and a brief description for each is given in this section.

Multiplier Types	
Sequential Add-Shift Multiplier [6],[11],[12]	Array Multiplier with Ripple Adder [10]-[12]
Modified Booth Algorithm with Ripple Adder [11],[12]	Array Multiplier with CLA [10]-[12]
Modified Booth Algorithm with CLA [11],[12]	Wallace Tree Multiplier with 4-bit AMM module [7]-[9], [11], [12]
ROM Table Look-up with Multiplier [11]	Wallace Tree Multiplier with 8-bit AMM module [7]-[9], [11], [12]
Modified ROM Table Look-up with Ripple Adder [5]	Wallace Tree Multiplier with 16-bit AMM module [7]-[9], [11], [12]
Modified ROM Table Look-up with CLA [5]	Wallace Tree Multiplier with 24-bit AMM module [7]-[9], [11], [12]

Table 1: List of multiplier architectures

2.1 Sequential Add-Shift Multiplication

Multiplication is performed by using successively the least significant bit of the ng multiplier for each add-shift cycle until all multiplier bits are exhausted [6], [11] and [12]. The functional blocks of the Add-Shift Multiplier is shown in Figure 1. During each cycle, a partial product is produced by a n-bit adder and is stored and shifted in an accumulator. Three n-bit parallel-load registers are required. The registers are denoted Accumulator (AC) which contains the high-order n-bits of the partial product, Multiplier Register (B), and Multiplicand Register (A). The concatenation of AC with B, provides for a double precision 2n-bit space in which to store the final double precision. A sequence counter is needed to keep track of the number of cycles and to recognize the completion of multiplication. An n-bit parallel adder, which employs carry look ahead adder (CLA), is required to generate new partial product from the multiplicand and last partial product.

For high speed, the sequential add-shift multiplier is not a good choice, but it requires a very small silicon area. Formulae for evaluating the area and delay of the multiplier are derived in terms of number of transistors and number of gate delays respectively.

$$Area_{Add-Shift} = n \times N_A + n \times N_B + n \times N_{AC} + n \times N_{AND} + k \times N_{CLA} + N_{Counter}$$

where N (for components A, B, AC and AND) is the number of transistors for each bit of the word, N_{CLA} is the number of transistors for a CLA adder module, k is the number of CLA adder modules, and $N_{Counter}$ is the number of transistors for a counter.

A formula for the worst case delay is derived. It accounts for the gate delay dependence on the number of its inputs. It calculates the maximum possible delay.

Δ is defined as a two-input logic gate delay. The critical path gate delay per cycle of Add-Shift Multiplier is defined as below:

$$\begin{aligned}\Delta_{Add-Shilt} &= \Delta_{AND} + \Delta_{CLA} + \Delta_{Register(storing)} + \Delta_{Register(shifting)} \\ &= 2\Delta + [8.825 + 2(2.825) \cdot (v-1)]\Delta + 8.62\Delta + 8.62\Delta\end{aligned}$$

where v is the level of CLA.

2.2 Modified Booth Algorithm Multiplication

Since the execution time of a multiplication is determined mainly by the number of additions to be performed, the speed of the multiplication can be increased by reducing the number of additions. Booth Algorithm Multiplier is implemented by recoding k consecutive ones in the multiplier with only one addition at the beginning and one subtraction at the ending of the strings [6], [11] and [12].

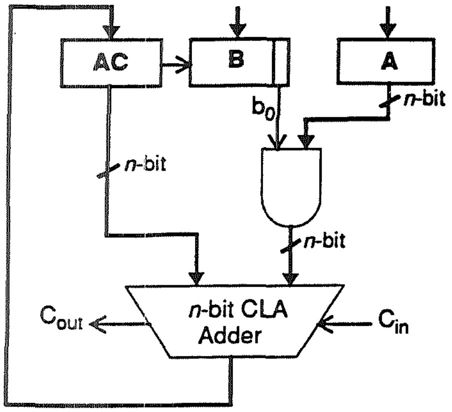


Figure 1: Block diagram of Add-Shift Multiplier with single right shift (AC-B) per circle

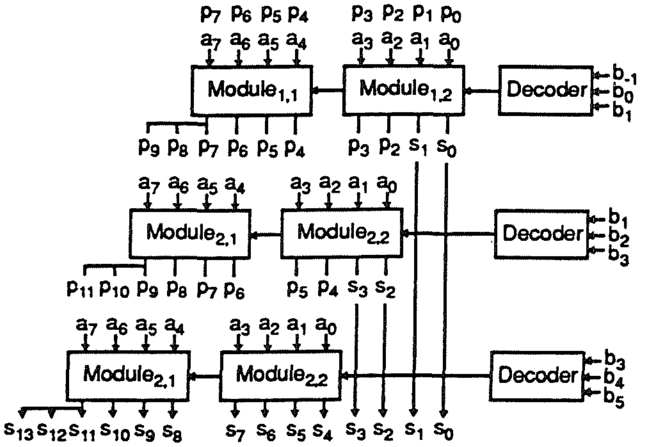


Figure 2: A 7 × 6 Modified Booth Algorithm Multiplier with planar array structure

The modified version of Booth Algorithm divides the multiplier into multiples of 3-bit segments with adjacent groups sharing a common overlapped bit [11], [12]. An increased multiplication speed can be achieved by evaluating two bits of the multiplier at a time. Table. 2 shows different versions of recoding and its operation. This architecture generates at most n/2 additions and partial products for a n-bit multiplier.

The block diagram of Modified Booth Algorithm using planar array is shown in Figure 2. The circuit consists only of 4-bit modules (to produce partial product) and decoders (to select any recoding version). Figure 3 shows the configuration of a 4-bit module which consists of four 5x1 multiplexers and a 4-bit ripple adder. The speed of this multiplication is higher than the Sequential Add-Shift multiplier but requires larger silicon area.

The area and delay of the Modified Booth Algorithm multiplier with ripple adder using 4-bit modules are given by:

$$Area_{MBA\_RA} = (3n+1)\cdot N_{register} + \frac{n}{2}[N_{decoder} + (n+1)\cdot N_{MUX} + (n+1)\cdot N_{FA}]$$

$$\begin{aligned}\Delta_{MBA\_RA} &= \Delta_{decoder} + \Delta_{MUX5\times1} + \tfrac{n}{2}[(n+1)\cdot \Delta_{FA}] \\ &= 3\Delta + [2+0.206(5-2)]\Delta + \tfrac{n}{2}[(n+1)\cdot 362\Delta]\end{aligned}$$

The ripple adder in the 4-bit modules can be replaced with a CLA adder to achieve higher speed. In this case, the area and delay are given by:

$$Area_{MBA\_CLA} = (3n+1)\cdot N_{register} + \tfrac{n}{2}[N_{decoder} + (n+1)\cdot N_{MUX} + k\cdot N_{CLA}$$

Multiplier Bits	
b_{i+1}, b_i, b_{i-1}	Operation
0 0 0	0× Multiplicand
0 0 1	+1× Multiplicand
0 1 0	+1× Multiplicand
0 1 1	+2× Multiplicand
1 0 0	−2× Multiplicand
1 0 1	−1× Multiplicand
1 1 0	−1× Multiplicand
1 1 1	0× Multiplicand

Table 2: Versions of bit-pair recoding for Modified Booth Algorithm Multiplier

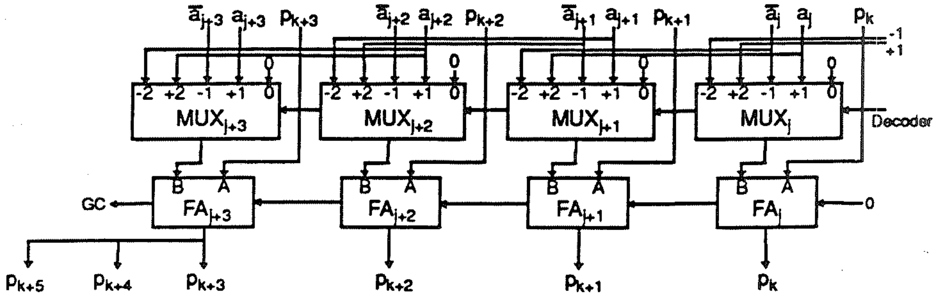


Figure 3: Configuration of a 4-bit module of Modified Booth Algorithm Multiplier

$$\begin{aligned}\Delta_{MBA\_CLA} &= \Delta_{decoder} + \Delta_{MUX} + \frac{n}{2}(\Delta_{CLA}) \\ &= 3\Delta + [2 + 0.206(5-2)]\Delta + \frac{n}{2}[8.825\Delta + 5.65(v-1)\Delta]\end{aligned}$$

2.3 Modified ROM Look-up Table Multiplier

Using a table lookup, the results of the multiplication are stored in a ROM for all possible combinations of the input vectors [11]. The size of the ROM grows exponentially with the word length of the input vectors.

Modified ROM Look-up Table uses a transformation to reduce the size of ROM from 2^{2n} to 2^n words [5]. However, additional overhead circuits are required. The area and delay of the Modified ROM Look-up Table Multiplier with ripple adder are given by:

$$Area_{Mod\_ROM\_RA} = n \cdot N_{2s} + 2n \cdot N_{FA1} + 2 \cdot N_{ROM} + 2nN_{2s} + 2n \cdot N_{CSA} + 2n \cdot N_{FA2}$$

where N_{FA1} and N_{FA2} are the number of transistors per bit of n-bit and 2n-bit ripple adders respectively.

$$\begin{aligned}\Delta_{Mod\_ROM\_RA} &= 2\Delta_{2s} + \Delta_{RA1} + \Delta_{ROM} + \Delta_{CSA} + \Delta_{RA2} \\ &= 2\Delta + 3.62n\Delta + \Delta_{ROM} + 3.62\Delta + 2n(3.62\Delta)\end{aligned}$$

Where Δ_{ROM} is the delay of the ROM, Δ_{RA1} and Δ_{RA2} are delays of n-bit and 2n-bit ripple adders respectively.

A CLA adder can be used instead of ripple adder (RA) to speed up the multiply time.

2.4 Array Multiplier Architecture

Array Multipliers are widely used because of its parallel multiplication technique, regular interconnection structure of array cell elements [10]-[12]. Since all summands of partial products are produced at the same time, the addition operation can be performed in parallel. The schematic circuit diagram of a 4 × 4 array multiplier with ripple adder (RA) is shown in Figure 4. The circuit for a $n \times n$ array multiplier needs n^2 AND gates and $n(n-1)$ full adders.

The area and delay of the Array Multiplier with RA are given by:

$$\begin{aligned}Area_{AM\_RA} &= n \cdot (n+1) \cdot N_{FA} + n^2 \cdot N_{AND} \\ \Delta_{AM\_RA} &= \Delta_{AND} + 2(n-1) \cdot \Delta_{FA} \\ &= 2\Delta + 2(n-1)(3.62\Delta)\end{aligned}$$

The multiplication speed can be increased by replacing the ripple adder in the final row of the array multiplier with CLA adder. In this case, the area and delay are given by:

$$Area_{AM\_RA} = (n+1)^2 \cdot N_{FA} + n^2 \cdot N_{AND} + k \cdot N_{CLA}$$

$$\begin{aligned}\Delta_{AM\_RA} &= \Delta_{And} + (n-1) \cdot \Delta_{FA} + \Delta_{CLA} \\ &= 2\Delta + (n-1) \cdot (3.62\Delta) + [8.825 + 2(2.825) \cdot (v-1)]\Delta\end{aligned}$$

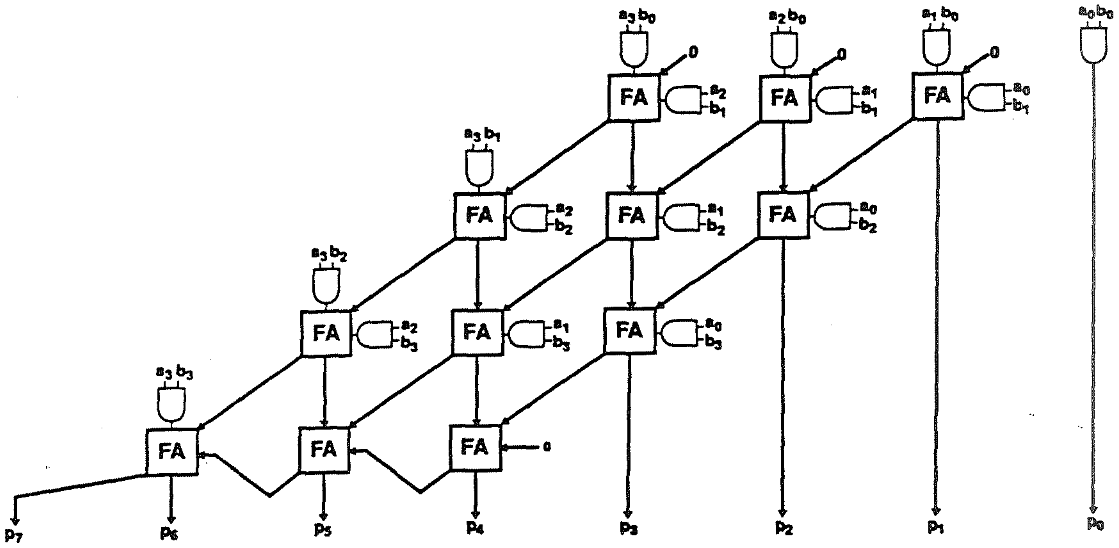


Figure 4: A 4 × 4 Array Multiplier with Ripple Adder (RA).

2.5 Wallace Tree Multiplier with Array Multiplier Module (AMM)

This multiplier architecture consists of three different functional blocks which include array multiplier modules, Wallace Tree adders and CLA adders [10]-[12]. The operation is modularized in sections of n-bit operands using AMM modules. Figure 5 shows a schematic block diagram of 8 × 8 bit Wallace Tree multiplier with 4-bit AMM modules. Four 4-bit AMM modules are used to generate 8-bit partial products simultaneously. The alignment of partial products in the proper column is accomplished by the appropriate 3-bit Wallace Trees using carry save adders (CSA). The sum and carry outputs from the Wallace Trees are input to a CLA adder to generate the final products. This multiplier has outstanding performance for large-scale word length of multiplication. However, more silicon area is required and the structure is less regular than the Array Multiplier. In this paper, this multiplier is used for discrete word lengths in steps of N where N is the size of the AMM module.

Let n be the word length of input vectors,

$$\rho(No.\_of\_AMM\_Modules) = \frac{n \times n}{N \times N}$$

Let h be the height of a Wallace tree,

$$\sigma(Level\_of\_Wallace\_Tree) = \lceil \log_{3/2}(\frac{h}{2}) \rceil$$

$$\chi(No\_of\_CSA) = h - 2$$

The area and delay are given by:

$$Area_{WTM} = \rho \cdot N_{AMM} + \chi \cdot N_{CSA} + k \cdot N_{CLA}$$

$$\Delta_{WTM} = \Delta_{AMM} + \Delta_{Wallace} + \Delta_{CLA} \ .$$

$$= (7.24N - 5.24)\Delta + \sigma(3.62\Delta) + [8.825 + 2(2.825) \cdot (v-1)]\Delta$$

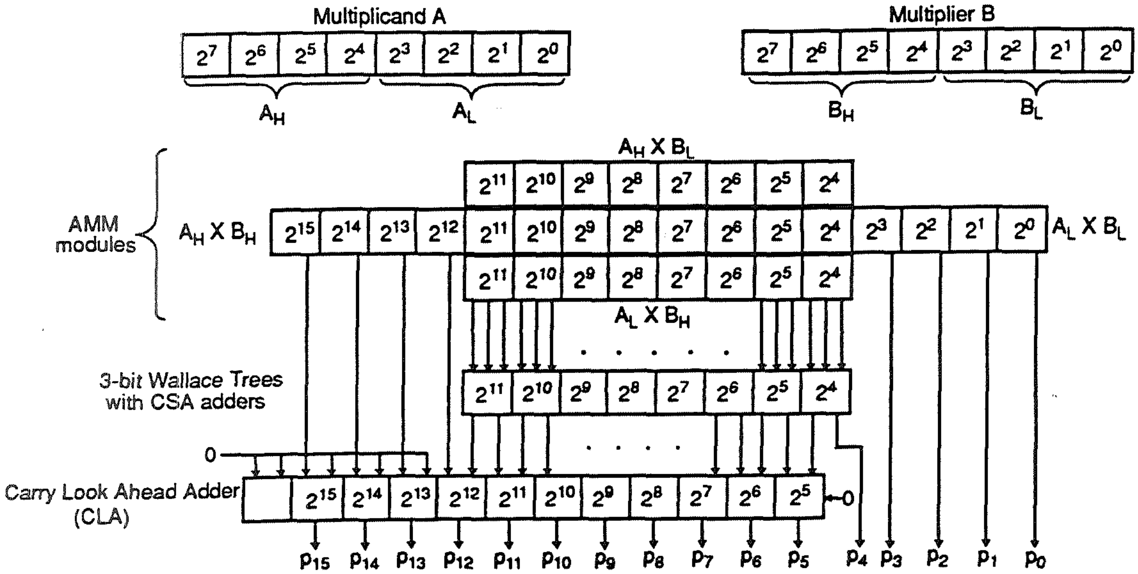


Figure 5: Block Diagram of a 8 × 8 Wallace Trees Multiplier with 4-bit Array Multiplier modules (AMM)

The 4-bit AMM modules can be replaced with 2k-bit ROM. To reduce the complexity of the circuit, especially for large word length, 8-bit AMM modules using CLA are used to produce 16-bit partial products so that the height of Wallace tree decreases and structure regularity increases.

2.6 Comparison Results

The comparison is based on the formulae that are described above for each multiplier architectures. Table 3 provides the relative speed, area and suitability for VLSI implementation of different multiplier architectures for the sub-range of 4 to 8 bits of word sizes. The speed is measured in units of a 2-input gate delay while the area is in terms of the total number of equivalent transistors. The suitability for VLSI implementation is measured basically by the structure regularity. Figure 6 and Figure 7 provide plots of the relative delay and area respectively for word lengths in the range of 4 to 8 bits. Similar plots are made for all other sub-ranges. These curves are used to select the proper multiplier architecture for each sub-range of word length.

As an example for this range, if speed is the designs first priority, the Array Multiplier with CLA adder is selected. This is because it has the highest speed and small relative area than other architectures. It is suitable for VLSI implementation due to its regular array structure which uses identical full adder elements and CLA adder as well as regular interconnections.

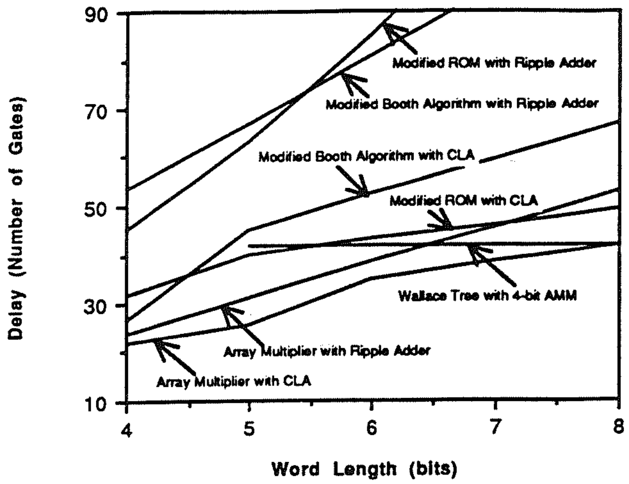


Figure 6: Comparison of delay of different multiplier algorithms for various word length

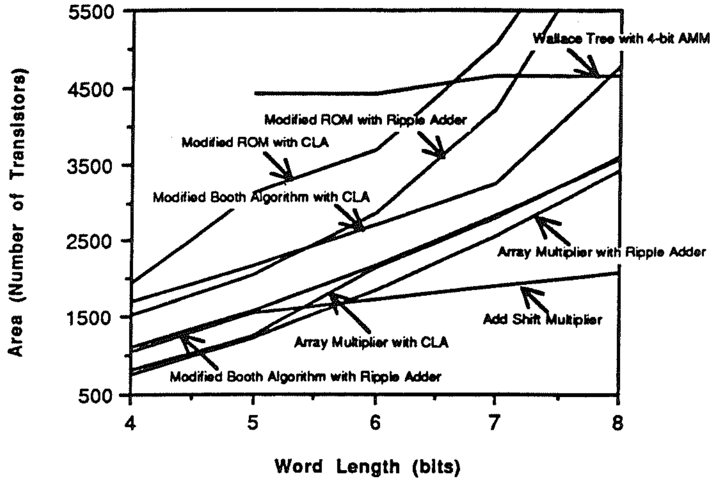


Figure 7: Comparison of different multiplier algorithms for various word length

Multiplier Types	Speed		Area		Suitability for VLSI Implementation†	
	4-bit	8-bit	4-bit	8-bit	4-bit	8-bit
Array Multiplier with Ripple Adder	2.28	2.58	1.00	1.64	1	1
Array Multiplier with CLA	2.49	3.25	1.10	1.73	2	2
Wallace Tree Multiplier with 4-bit AMM	–	3.24	–	2.24	–	3
Sequential Add-Shift Multiplier	0.48	0.50	1.38	1.00	3	3
Modified Booth Algorithm with Ripple Adder	1.30	1.00	1.49	1.71	2	2
Modified Booth Algorithm with CLA	2.30	2.14	2.28	2.29	3	3
Modified ROM Table Lookup with Ripple Adder	1.00	1.24	2.04	3.30	2	3
Modified ROM Table Lookup with CLA	1.67	2.72	2.60	3.60	3	4

† Best suitability is indicated by 1 and increasing numbers for higher complexity

Table 3: Comparison of relative speed, area and suitability for VLSI implementation for different multiplier architectures

3 Synthesis Algorithm

Based on the speed and area comparison in the whole range of 4 - 256 bits, the synthesis algorithm divides the supported range of the word lengths into sub-ranges (4-12 bits, 13-64 bits and 65-256 bits). Based on the designs priorities and requirements, an optimal multiplier architecture is selected for each sub-range. These priorities include speed and silicon area. A third important factor that affects the selection is the suitability Z automated synthesis and verification of its VLSI implementation.

Table 4, shows four different design options for the supported range of the word length.

		Word Length Ranges		
Priority	Design Options	4-12 bits	13-64 bits	65-256 bits
1	Speed	Array Multiplier	Wallace Trees	Wallace Trees
2	Area	with	with	with 8-bit AMM
3	Suitability for VLSI Implementation	CLA†	4-bit AMM‡	using CLA
1	Area	Array Multiplier	Array Multiplier	Wallace Trees
2	Speed	with	with	with
3	Suitability for VLSI Implementation	Ripple Adder	CLA	8-bit AMM
1	Speed	Array Multiplier	Wallace Trees	
x	Area	with	with	
x	Suitability for VLSI Implementation	CLA	4-bit AMM	
1	Area			
2	Speed	Sequential Add-Shift Multiplier		
3	Suitability for VLSI Implementation			

†CLA - Carry Look Ahead Adder
‡AMM - Array Multiplier Module with ripple adder

Table 4: Algorithm of selecting optimal multiplier architecture for the four design options

The entries in the table give the selected multiplier architectures. For example, in the first and option, speed has the highest priority followed by the silicon area in the second place while the suitability for VLSI implementation has the least priority. For this option, the array multiplier and Wallace tree multiplier architectures cover the whole range of word length.

The flow chart of synthesis algorithm is shown in Figure 8.

The selection of the different multiplier architectures in Table 4 is based on comparing their silicon areas and delay times as given by the formulae presented in the previous section. In order to verify the accuracy of these formulae, the actual physical layout and accurate simulation have been done. For the Array Multiplier with RA, Array Multiplier with UCLA and Wallace Trees Multiplier with 4-bit AMM, the physical layouts were generated using a parameterized cell library and an automatic placement and routing tools (Cells and AutoCells). The simulation was done using a mixed mode simulator (LSIM). Figure 9 and Figure 10 show the measured and calculated curves for critical path delay and silicon area in the range 4 to 8 bits. The layout areas and simulated delays are in good agreement with results obtained using the formulae. The same methodology is used for verifying the accuracy of the formulae for other sub-ranges.

The synthesis algorithm takes the users specifications of word length, design speed and

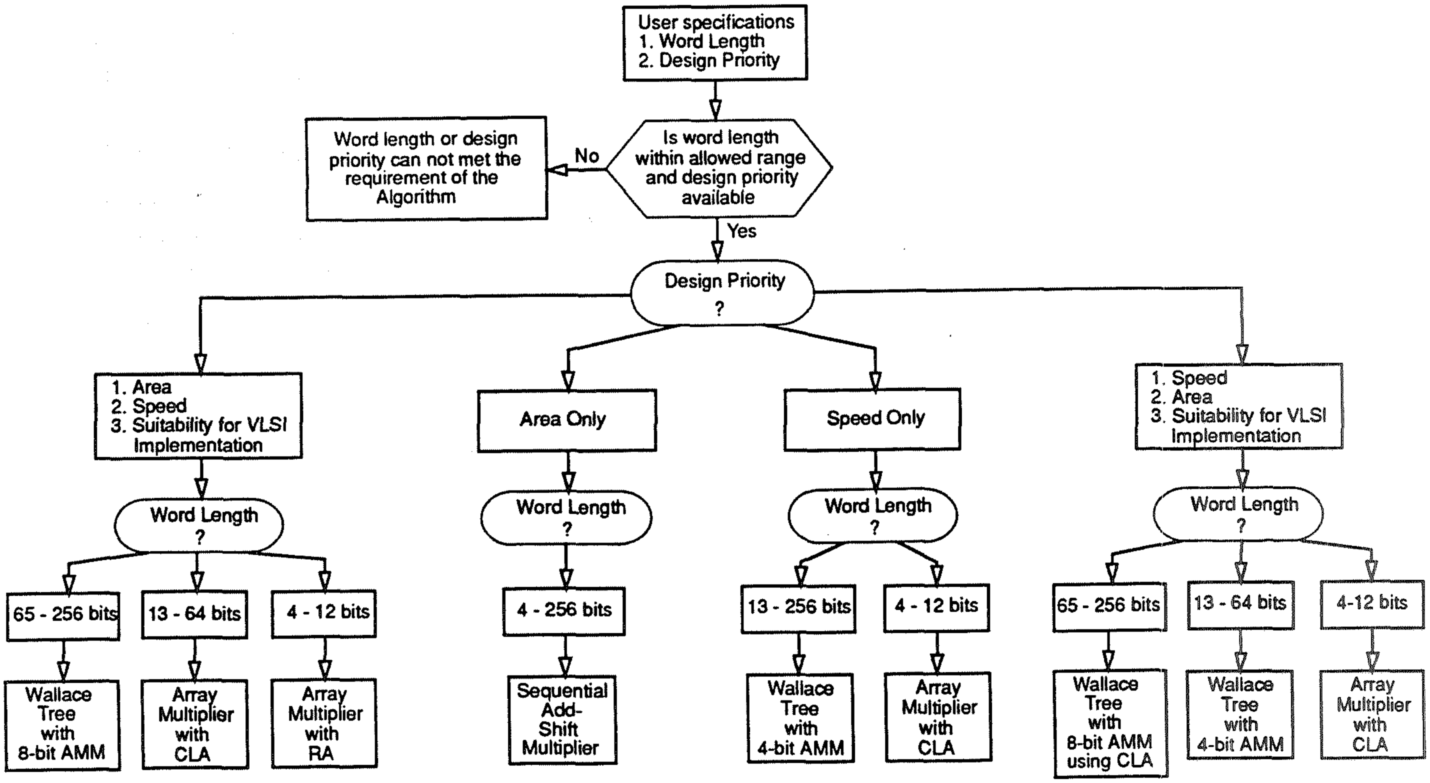

User specifications
1. Word Length
2. Design Priority
Is word length within allowed range and design priority available
No
Word length or design priority can not met the requirement of the Algorithm
Yes
Design Priority ?
1. Area
2. Speed
3. Suitability for VLSI Implementation
Area Only
Speed Only
1. Speed
2. Area
3. Suitability for VLSI Implementation
Word Length ?
Word Length ?
Word Length ?
Word Length ?
65 - 256 bits
13 - 64 bits
4 - 12 bits
4 - 256 bits
13 - 256 bits
4 - 12 bits
65 - 256 bits
13 - 64 bits
4-12 bits
Wallace Tree with 8-bit AMM
Array Multiplier with CLA
Array Multiplier with RA
Sequential Add-Shift Multiplier
Wallace Tree with 4-bit AMM
Array Multiplier with CLA
Wallace Tree with 8-bit AMM using CLA
Wallace Tree with 4-bit AMM
Array Multiplier with CLA

Figure 8: Synthesis Algorithm for determining an optimal multiplier of specific design priority for each subring

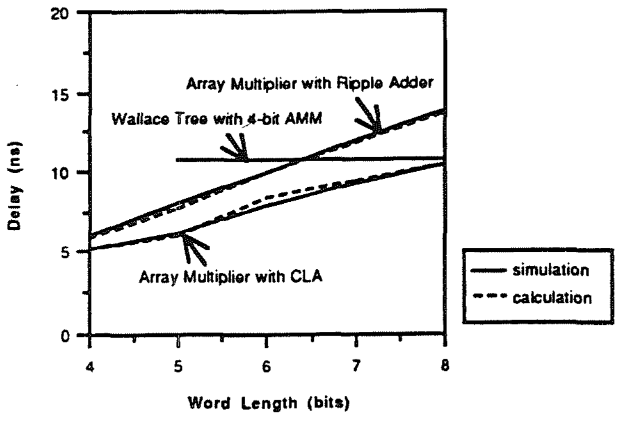


Figure 9: Comparison of delay of three different multplier architectures for word length (4-8) bits

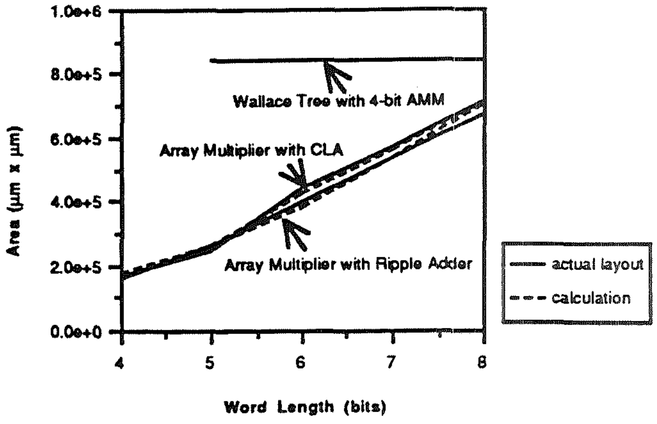


Figure 10: Comparison of silicon area fo three different multiplier architectures for word length (4-8 bits)

silicon area priorities as inputs, and automatically provides the physical layout of the optimal multiplier architecture.

4 Multiplier Architecture and Circuit Design

The full adder is the basic sub-circuit used in the different multiplier architectures of Table 4., Figure 11 shows the logic diagram of the full adder and its circuit implementation (using transmission gate logic. This provides very small area and high performance. The latency of the adder is 1.38 ns using 0.8 μm CMOS technology.

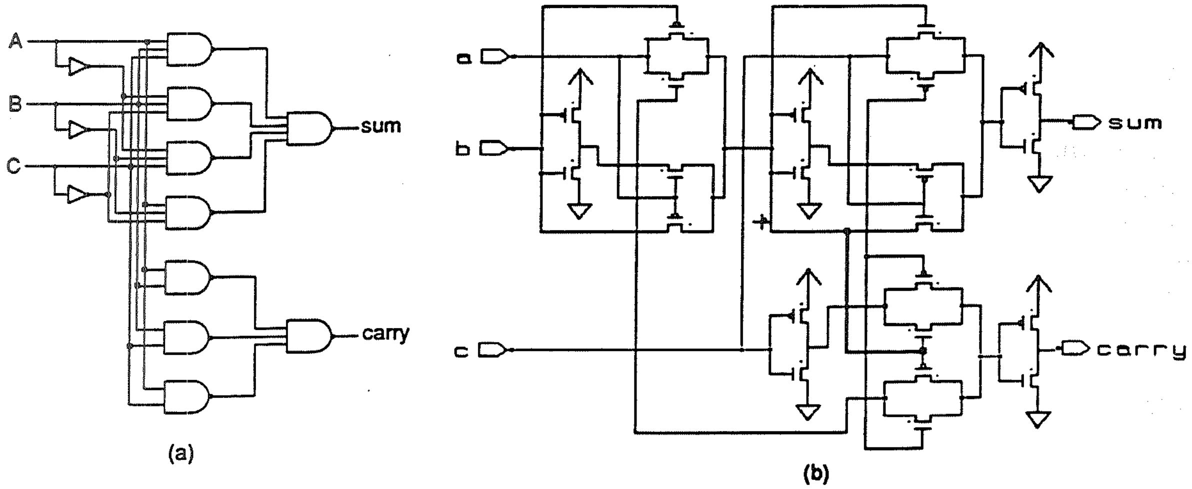


Figure 11: Full Adder (a) Logic Diagram (b) Transmission Gate logic implementation

5 Automated Physical Synthesis

An automatic placement and routing tool (AutoCells) is used to verify the validity of the silicon area formulae and in turn the multiplier architecture selection algorithm. The synthesis algorithm uses a library of multiplier generators. Each is dedicated to produce a very compact layout of one of the multiplier architectures of Table 4.

These generators have been developed using a general-purpose interpreter language (GENIE) and a procedural interface language (Lx). Figure 12 shows an example layout of 9 × 9 bit Array Multiplier with RA. The leaf cell has dimensions of 52 × $29\mu m^2$ using $0.8\mu m$ CMOS process. The complete multiplier layout is shown in Figure 13 with dimensions of 604 × $245\mu m^2$.604 × $245\mu m^2$.

6 Conclusion

A flexible multiplier synthesis algorithm has been developed which is capable of generating multiplier blocks for word length in the range of 4 to 256 bits. Silicon area and delay time formulae have been developed for several promising multiplier architectures and used to select

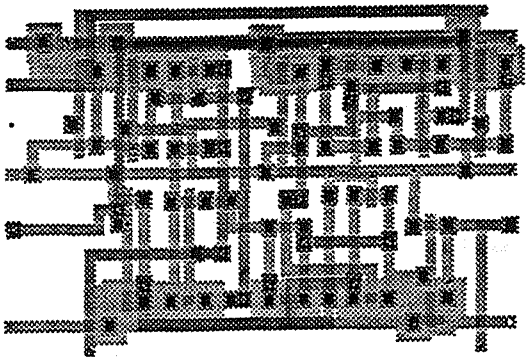


Figure 12: Layout of a leaf cell of Array Multiplier with RA

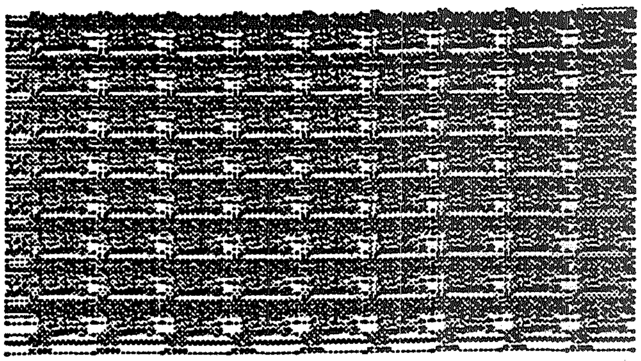


Figure 13: Layout for a 9 × 9 Array Multiplier with Ripple Adder

an optimal multiplier architecture for each sub-range. Several design priorities are considered in the choice of the multiplier architecture. The algorithm of the structural synthesis is presented and the method of automated physical layout generation is discussed. Circuit implementation of the basic full adder cell is described. A sample 9 × 9 bit multiplier layout is presented. The algorithm presented and the automated generation of layout discussed provide high degree of design flexibility, high performance and very dense layout of ASICs multiplier blocks. It also enables very quick and accurate system optimization.

References

[1] S. Wallace, "A Suggestion for a Fast Multiplier", IEEE Trans. on Electronic Computers, vol. EC-13, pp. 14-17, Feb. 1964.

[2] D. Booth, "A Signed Binary Multiplication Technique", Qt. J. Mech. Appl. Math., vol. 4, Part2, 1951.

[3] J. Mori et al., "A 10-ns 54 × 54 Parallel Structured Full Array Multiplier with $0.5 - \mu m$ CMOS Technology",IEEE J. Solid-State Circuits, vol. 26, pp. 600-606, Apr. 1991.

[4] M. Nagamatsu et al., "A 15-ns 32 × 32-b CMOS Multiplier with an Improved Parallel Structure", IEEE J. Solid-State Circuits, vol. 25, pp. 494-497, Apr. 1990.

[5] H. Ling, "An Approach to Implementing Multiplier with Small Tables", IEEE Trans. on Computers, vol. 39, no. 5, pp. 717-718, May 1990.

[6] G. K. Ma and F. J. Taylor, "Multiplier Policies for Digital Signal Processing", IEEE ASSP Magazine, pp. 6-20, Jan. 1990.

[7] T. Sato et al., "A Regularly Structured 54-bit Modified-Wallace-Tree Multiplier", in Proc. IFIP International Conference on VLSI (VLSI91) (Edinburgh), Aug. 1991, pp. 1.1.1-1.1.9.

[8] Paul J. Song et al., "Circuit and Architecture Trade-offs for High-Speed Multiplication", IEEE J. Solid-State Circuits, vol. 26, no. 9, pp. 1184-1198, Sept. 1991.

Design and Implementation of GaAs HBT Circuits with ACME[1]

Brad L. Hutchings
Dept. of Electrical and Computer Eng.
Brigham Young University
Provo, UT 84602

Tony M. Carter
Dept. of Computer Science
University of Utah
Salt Lake City, UT 84112

Abstract - **GaAs HBT circuits offer high performance (5-20 GHz) and radiation hardness (500 Mrad) that is attractive for space applications. ACME is a CAD tool specifically developed for HBT circuits that implements a novel *physical* schematic-capture design technique where designers simultaneously view the structure and physical organization of a circuit. ACME's design interface is similar to schematic capture; however, unlike conventional schematic capture, designers can directly control the physical placement of both function and interconnect *at the schematic level* . In addition, ACME provides design-time parasitic extraction, complex wire models, and extensions to Multi-Chip Modules (MCMs). A GaAs HBT gate-array and semi-custom circuits have been developed with ACME; several circuits have been fabricated and found to be fully functional.**

1 Introduction

This paper presents ACME, a VLSI design tool that is based upon a unification of the physical and structural design domains. ACME uses a single unified design representation that allows designers to simultaneously manipulate and inspect the physical and structural features of a design *from within a single design view*[2]. "Schematic capture with physical placement" is perhaps the most succinct and intuitive way to describe ACME's unified capabilities. ACME is well-suited to high-speed design because designers can precisely control the physical layout of circuit elements that have critical performance or area concerns and yet still employ a structural design style to quickly and efficiently design integrated circuits. In addition, designers have ready access to information such as circuit area and speed, interconnect delay, interactions between interconnect (crosstalk) and power dissipation *early in the design process when this information can have the largest impact.*

2 Technical Approach

Conventional design environments organize the physical and structural parts of VLSI design hierarchically as domains; circuit design is typically a top-down process where the *designer* provides the structural description (schematic) and the *system* generates the physical description (layout). An unfortunate limitation of this hierarchical organization is that *designer interaction is strictly limited to a single domain at a time.* The designer is stuck with an

[1] Sponsored by DARPA through the FBI under contract number J-FBI-89-057

"all-or-nothing" approach where only purely structural (schematic) or purely physical (layout) descriptions can be manipulated —no methodology exists for fully integrating the two design techniques in a single design environment.

In contrast to the structural-physical dichotomy found in most design environments, the ACME design system uses an alternate design organization that allows designers to freely intermix structural and physical design entities. ACME unifies the structural and physical design spaces and fully integrates structural and physical design tools. A crucial advantage of the ACME approach is that *the concepts "structural" and "physical" no longer refer to a domain or design view, but rather to individual design entities, e.g., cells and wires.* Structural cells and wires, for example, have screen representations that are completely unrelated to their physical realizations. A cell or wire may be any shape or size and may be placed at any location; it still represents a strictly structural organization. However, if a cell is physical, then its screen representation, i.e., its relative shape, size and port locations, is directly related to its physical organization. This means that, for physical components, the relative topology, size and shape of the schematic as seen on the computer monitor directly represents the final topology of the physical circuit.

3 Organization of the Unified Design Environment

The unified design environment of ACME was developed by augmenting the physical design capabilities of Path-Programmable Logic (PPL) [3, 4] and incorporating structural design capabilities. A conceptual view of a structural-physical design example is shown in Figure 1 (this is a conceptual example because it shows the cell and wire grids that are invisible when performing design with ACME).

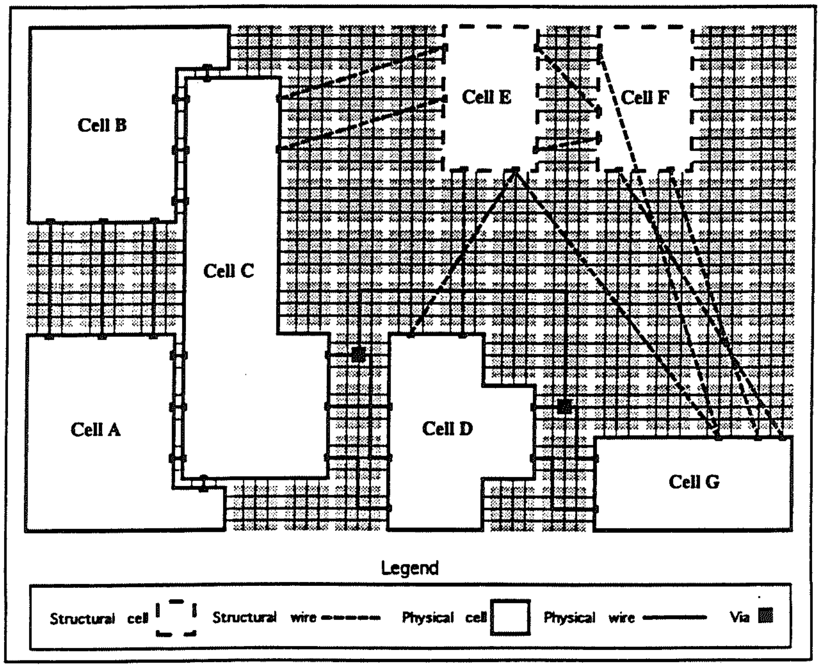


Figure 1: A Conceptual View of ACME

For physical design purposes, ACME uses a cell-grid and wire-grid similar to PPL. The cell-grid is shown in the figure as gray squares. These squares represent the unit locations, that is, the x-y locations on the grid where cells may be placed. The wire grid is superimposed over the cell-grid and consists of the straight dotted lines oriented in the x and y directions. Both structural and physical entities are represented in the figure: physical wires (drawn with a non-dashed line), physical cells (drawn with a non-dashed outline), structural wires (drawn with a dashed line), and structural cells (drawn with a dashed outline). As shown in the figure, structural and physical cells are aligned with the cell-grid. Physical wires are aligned with the wire-grid. Structural wires are not aligned to any grid; their only restriction is that they start and end on *port locations.* Port locations represent the permissible wiring locations along the periphery of the cell; they are represented in the figure as the small black rectangles situated where the wire touches the cell.

4 Design Interaction in the Unified Environment

Wherever possible, the unified design environment treats physical and structural design entities in uniform fashion. Physical and structural wires, for example, are entered into a design with an interactive *wiring tool* similar to that used for connecting nets in a schematic-capture editor. This in in contrast to the more awkward cell-based routing approach used in PPL. General interconnect is implemented by placing physical or structural wires between the cell ports of interest. Structural wires have the fewest number of constraints imposed upon them; they may overlap any cell or wire at any point and are used to represent functional interconnect. Multiple structural wires can "fan-out" from a single cell port; this allows multiple structural wires to be interconnected without the need of a "via" mechanism. Physical wire placement is more constrained. Physical wires are allowed to overlap only if they exist on different layers. Overlapping physical wires on different layers can be connected together with vias (represented in the figure as small rectangles at wire intersections). If no via is present, overlapping wires are not connected. Physical wires may overlap structural cells at any point; however, they are allowed to overlap *physical* cells only if the physical wire track in the interior of a cell is not already occupied by a physical wire or some other active circuitry such as a transistor, etc. All physical wire constraints are imposed *interactively* as wires are added to the design. As physical wires are added to the design, the wires are automatically "snapped" to the physical grid. This "snapping" of physical wires insures that wires can only be inserted at integer x-y locations. Power and ground connections are not shown in Figure 1; similar to PPL, they are automatically handled by a separate grid that is ordinarily hidden from the designer.

Cell interaction is similar to wires; designers select cells from available libraries and insert them into designs with an interactive *cell tool.* Similar to wires, cells are snapped to the cell-grid as they are added to the design. Designers can create new cells by grouping cells and wires through a hierarchical process referred to as "wrapping". This is performed in two steps. First, the designer creates the *implementation* of the new cell. An implementation can consist of either other cells and wires, or a mask-level physical layout described in Caltech Intermediate Format (CIF). Second, the designer creates the cell's *interface* by drawing a bounding box, placing and naming the desired ports, and creating the cell's *wrapper.*

4.1 Cell Interaction and the Bounding Box

For all cells, the bounding box provides designers with a "handle" to the cell. Designers *select* cells (indicate that the cells are to be affected by commands) by clicking the mouse button when the cursor is within the cell's bounding box. For physical cells, the bounding box (as shown on the computer monitor) also indicates the actual physical size and shape of the cell, relative to the other cells in the design. For structural cells, the bounding box is used only for selection purposes. The bounding box also serves to indicate the physical/structural attribute of the cell. As shown in Figure 1, physical cells have a solid bounding box and structural cells have a dashed bounding box.

4.2 Cell Views, Wrappers and Hierarchy

The wrapper provides the graphical view that the designer normally "sees" when viewing a cell. The wrapper is generated so that it fits within the bounding box. The designer may use a default wrapper, automatically created by ACME, or the designer may create a custom wrapper with a drawing editor. For example, the wrappers for the cells in Figure 1 are the text symbols for the cell names, e.g., "Cell A", etc. Wrappers can consist of any graphics that the user wishes to draw. A designer can easily create multiple views of a cell by creating gate wrappers, block-diagram wrappers, etc.

The implementation of a cell may be viewed at any time by *clear-wrapping* the cell, a process that makes the cell's wrapper effectively transparent. Thus when a cell is clear-wrapped, the designer sees the cell's bounding box and the graphical view(s) of the cell's constituent components. The cell's wrapper is *not* visible and the designer views the cell's *implementation* through its "clear" wrapper. Clear-wrapping does not affect the actual hierarchical organization of a cell, only its view.

Hierarchy can be destroyed or modified by *unwrapping* the cell. In-place hierarchical editing of cell constituents is also fully supported. Since ACME maintains all levels of hierarchy, designers can *push-into* a cell and edit its implementation within the context of the surrounding design. Since the effects of in-place hierarchical edits are encapsulated within the cell being edited, designers can edit a cell's implementation within the context of its surrounding circuitry without the possibility of inadvertent modification of the surrounding circuitry.

4.3 Determination of Physical and Structural Design Attributes

The structural or physical attribute of a design entity can be assigned manually by the user or automatically by ACME. Automatic assignment of the physical/structural attribute makes structural cells and wires extremely useful as an intermediate result while experimenting with different physical implementations. For example, during the floorplanning stages, the designer may choose to manipulate physical circuit modules that are interconnected with physical wires. As these modules are rearranged, the physical wires interconnecting them may no longer be viable because of some physical constraint, e.g., not enough space. Rather than delete them, ACME automatically converts these physical wires into structural wires, maintaining interconnect information for simulation and netlisting purposes.

Automatic assignment of the structural attribute makes structural cells useful in a manner similar to wires. For example, if a designer edits the implementation of a physical cell such that it becomes larger than its original bounding box, ACME automatically converts the physical cell into a structural cell. Once converted to structural, changes in the physical size and shape of a cell's implementation no longer immediately affect the overall physical organization and topology of the design. Thus the designer has fewer constraints to deal with. Cell overlap, for example, is not a concern with structural entities. The designer can freely experiment with the physical details of the cell's implementation and any spatial changes (shape, size, etc.) remain *encapsulated* within that structural implementation. Once a physical cell is converted to structural the designer need not worry prematurely about how the implementation will affect the global chip topology *until the designer has the information (usually simulation data and area estimates) necessary to commit to a specific physical implementation.* This facilitates exploration of the design space because all of the physical effects of design exploration, e.g., changes in the shape or size, need not affect the design's topology. The designer can easily switch between implementations, simulate them and make area estimates, etc., without making any changes to the general chip topology.

ACME automatically classifies cells to be either structural or physical according to the following criteria:

physical cell A cell is physical only if its implementation contains only physical cells and wires and its bounding-box fully covers the physical space occupied by its implementation.

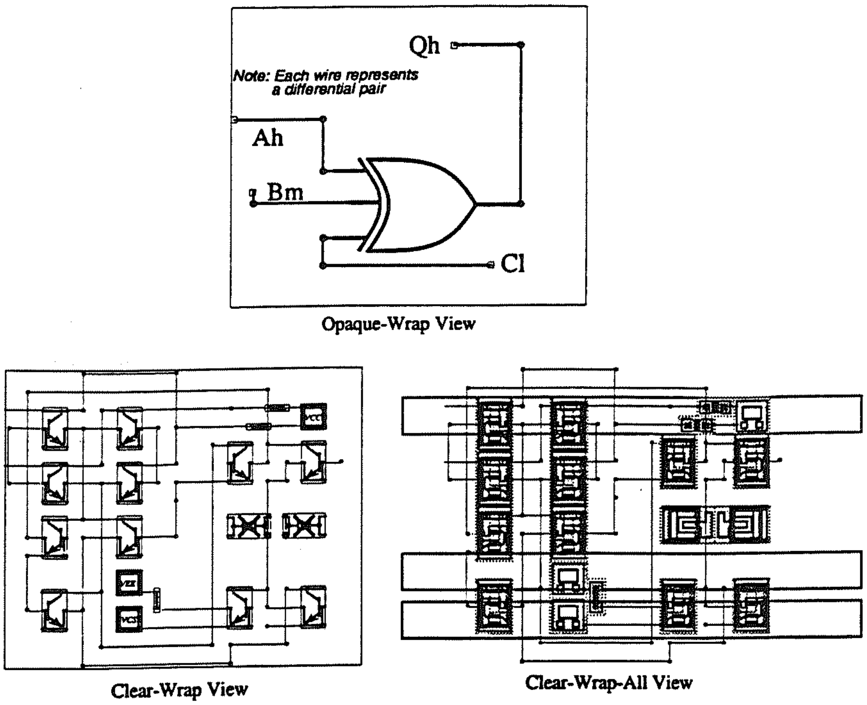
structural cell A cell is considered structural if its implementation contains any cells that are structural or if its bounding-box does not fully cover the space occupied by its implementation.

5 Demonstration of the ACME Design Tool

ACME has been implemented in approximately 100K lines of GNU C++ and runs on SUN workstations. It is currently being used to design CMOS and GaAS MESFET circuits and an HBT gate array. Figure 2 presents three design views of a three input, HBT exclusive-or (exor) cell that was designed with ACME.

5.1 Viewing Designs in ACME

The "Opaque-Wrap" view, shown at the top of the figure consists of the cell's bounding box and a custom wrapper drawn by the designer with Idraw. The bounding box is displayed with a solid outline; this indicates that the cell is *physical.* The custom wrapper provides a graphical view that indicates the gate-level functionality of the cell. The "Clear-Wrap" view shown at the lower left-hand side of the figure is another view of the same exor-cell shown in the Opaque-Wrap view after executing the clear-wrap command. The bounding box is still visible; however, when a cell is clear-wrapped, the wrapper becomes effectively transparent and the cell's "guts", or implementation, becomes visible. As can be seen in the Clear-Wrap view, the exor-cell is a hierarchical cell composed of other cells and wires. This exor-cell was



Opaque-Wrap View

Clear-Wrap View

Clear-Wrap-All View

Figure 2: HBT Gate-Array Cell: Exor Gate

created with the wrap command described in Section 3. It consists of 19 cells: 11 transistors, 2 diodes (the cells with the "x" in them, they are currently disabled), 3 resistors (the ohmic value can be customized by the designer) and 3 voltage sources (VCC, VEE, VCS). All the cells and wires used to implement the exor-cell are physical as shown by the solid lines for the wires and the solid outlines for the cell's bounding boxes. Note the use of vias where wires change directions (change layers).

Finally, the "Clear-Wrap-All" view in Figure 2 is another view of the same exor-cell shown in the Opaque-Wrap view, obtained by executing the clear-wrap-all command. This command implements a *recursive* clear-wrap operation that clear-wraps all the constituent cells throughout all levels of the hierarchy. Because the clear-wrap-all command clear-wraps all cells, the actual physical layout used to implement the primitive cells (transistors, diodes, resistors, etc.) becomes visible. Note that the topology of the Clear-Wrap-All view closely corresponds to the topology of the schematic view. The implicit power and ground connections (the long narrow rectangles that horizontally span the design) that are part of the CIF descriptions also become visible when all cells are clear-wrapped. The view of the user-entered wires remains in stick-form, the actual physical width of these wires can only be seen when CIF is generated for the cell.

5.2 Developing HBT Circuits With ACME

The cell library prepared for the HBT gate array consists of low-level circuit primitives, e.g., transistors, diodes, resistors, etc., that were developed with a layout editor. This is in contrast to the more complex cells, e.g., gates, flip-flops, etc., found in typical cell libraries. These low-level primitives provide the high-speed designer with more flexibility in developing circuit organizations. However, the designer of HBT circuits is not restricted to design with only these low-level primitives. The configuration of the low-level cells used in the exor-cell (as shown in the Clear-Wrap view of Figure 2) also serves as the base configuration for developing a complete logic-cell library for the HBT gate array. Designers can customize this base cell (the exor-cell, sans wires) by connecting the internal components with physical (or structural) wires to create the desired organizations. Several additional cells have already been designed (using the base configuration, with different wiring), including: muxes, latches, and-gates, buffers, etc. These hierarchical cells, along with the lower-level primitive cells such as transistors, diodes, etc., are all available to the circuit designer.

ACME is an ideal choice for HBT gate-array prototyping work. Different base-cell configurations can be rapidly prototyped and their physical size, performance characteristics, etc., are readily available for inspection and simulation purposes. Extensive wire modeling information, based on physical design parameters, is available to the designer just by clicking the mouse on the wire of interest. The ready availability of physical wire information is largely due to the constrained physical placement implemented in ACME. Since the potential placement of all physical wires is known a priori, much of the data necessary for extensive wire modeling can be *pre-extracted* so that it can be quickly accessed within the ACME design environment on an interactive basis. For example, ACME can interactively determine series resistance and capacitance for cross-overs and parallel wires up to a user-defined distance for the wire of interest. Additional support is available for determining self-inductance with reference to a nearby ground plane. Using the determined and lumped values of R, L and C, ACME can emit decoupled models for wires that include any or all of these three values. Planned extensions to ACME's physical modeling capabilities include crosstalk analysis (inductive and capacitive), use of transmission line models for long interconnects and improved inductance modeling.

5.3 CIF Layout Generation

Figure 3 depicts the actual composite layout (CIF format) that ACME generates directly from the exor-cell's implementation as shown in the Clear-Wrap view of Figure 2. Again, note how closely the physical layout corresponds to the "Clear-Wrap" view. The physical location of all transistor, diode and power cells and interconnect directly corresponds to that seen in the schematic view of Figure 2. Physical wires are correctly rendered in the composite layout and are shown in gray with via interconnects between layers. The diodes are left unconnected as they are not used for this cell (they are part of the base-cell configuration). As before, the implicit power and ground wires span the design horizontally.

5.4 Mixed Physical-Structural Design

Figure 4 depicts a mixed structural-physical schematic in ACME. This design was generated by performing an in-place, hierarchical edit of the exor-cell depicted in the schematic view in Figure 2. Two transistors were moved for demonstration purposes; these are marked as "Cell A" and "Cell B." Cell A was moved upward from its original position. Note that the original physical wires interconnecting the cell have moved with the cell and have been automatically

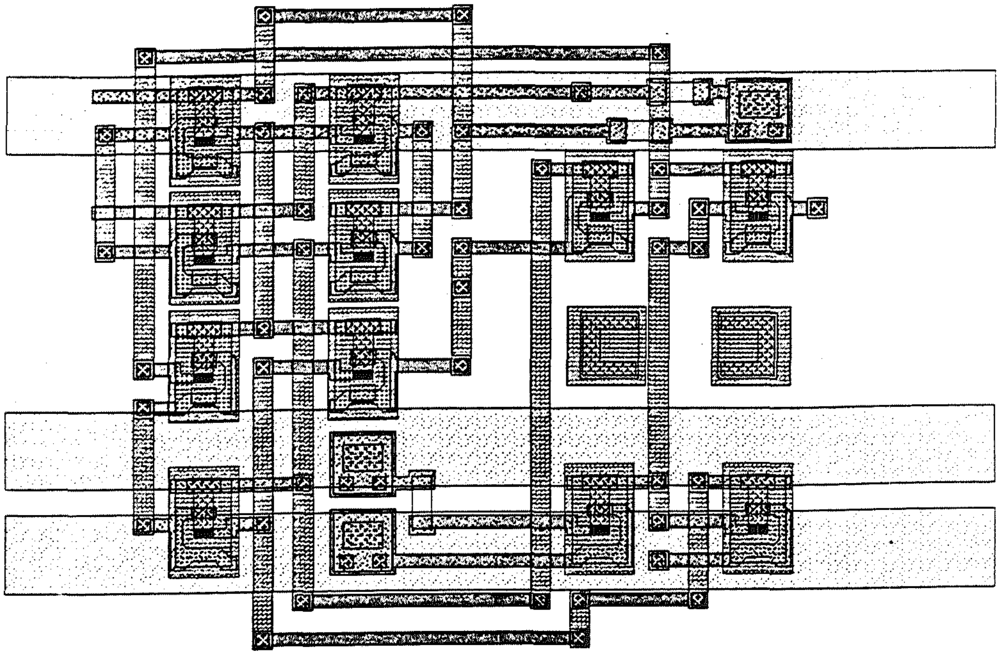


Figure 3: Composite Layout of Exor Gate

converted to structural wires (dashed line). Cell B was moved horizontally to the left from its original position; the two horizontal connecting wires were automatically stretched out and remain physical, the vertical wire was converted to structural. This demonstrates ACME's capability to automatically maintain physical connections as cells are moved. In cases where the physical wires can be maintained by simple stretches in the x- or y-direction, ACME maintains the physical connection. ACME converts physical wires to structural only if it is not possible to maintain the wire as described above. In addition, the boundary of the exor-cell is also drawn with a dashed line, indicating that the exor-cell itself has been automatically converted to structural. This is in contrast to the Clear-Wrap view in Figure 2 where the boundary of the exor-cell is drawn with a solid line. The cell becomes structural because it now contains structural design entities, i.e., wires. Although this cell configuration is not physically realizable, (transistor placement is critical for HBTs), the cell can be simulated because the interconnection information is maintained by the structural wires. This figure demonstrates ACME's ability to combine structural and physical design entities, and its ability to automatically convert physical cells and wires into structural cells and wires, as necessary.

6 Conclusions

This paper contributes a new unified design approach that allows designers to simultaneously develop and manipulate the structural and physical features of VLSI circuits. The resulting unified design environment provides a physical design approach that, based upon the cell-matrix design methodology, competes very well with full-custom with regard to area and far surpasses it with regard to manpower requirements[1]. This research also augments the cell-matrix design methodology to include structural design techniques. It demonstrates that structural design techniques are effective for *both manual physical design and structural*

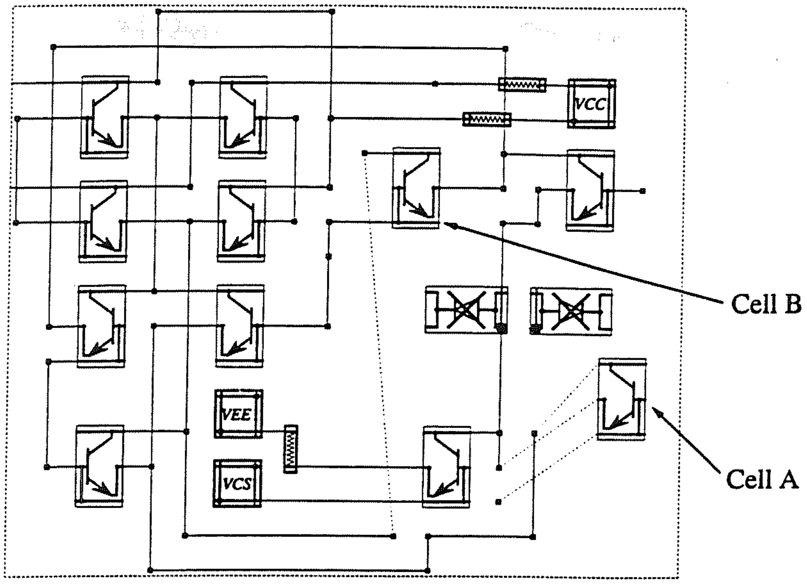


Figure 4: Exor Cell: Structural-Physical View

design. The unified design environment allows the designer to immediately view the effects of automated design techniques (auto-routing, etc.) on the design, and then in turn, modify the automatically-generated circuitry, if desired. In addition, all manual modifications to the automatically-generated physical description are immediately reflected in both the physical and structural description *with the same effect as extensive, automated back-annotation in a conventional nonunified design environment.* The approach outlined in this paper accomplishes all of this within a single design environment and requires only a single style of design interaction.

6.1 ACME and High-Speed Design

ACME is very well-suited to the design of high-speed circuits. It is arguably the only viable *high-level* design tool for the design of GaAs HBT circuits. The only other possibilities are to manually create the physical layout or to use conventional schematic capture. Physical layout does allow the designer to control the placement of all cells and wires; however, it also forces designers to deal with design rules, cell overlap, etc. Requiring the designer to deal with *all* physical design details makes physical layout an inappropriate choice for the design of MSI HBT circuits. Schematic-capture approaches (standard cell, gate arrays, etc.) allow designers to develop circuitry from a higher level (structural). Unfortunately, the requirements of HBT circuits (and high-speed circuits in general) are beyond the capabilities of the place-and-route software used in these approaches. For example, HBT circuitry transmits signals on pairs of differentially-driven wires and it is imperative that these pairs of wires are carefully placed, typically maintaining a minimal interwire spacing, to preserve the signal integrity. ACME is well-suited to high-speed design because it allows designers, within a cell-based physical design context, to control the most important physical design details, e.g., cell and wire placement, without requiring them to deal with less germane and distracting problems such as design rules, cell overlap, etc. ACME provides a high-level approach for the design of HBT circuits that provides important physical feedback early in the design process so that the designer can select components and control the placement and organization to maximize performance.

7 Future Work

ACME provides a good foundation for a variety of research efforts related to CAD-VLSI design. This foundation is already being used for the study of a variety of physical design topics: physical modeling, module generators, integrated simulation environments and routing kernels. In addition, ACME could support research in the area of floor planning, partitioning, global routing and other automated structural design techniques. The unified structural design environment implemented by ACME is also well suited for supporting research in physically-constrained synthesis.

References

[1] T. M. Carter, K. F. Smith, S. R. Jacobs, and R. M. Neff. Cell matrix methodologies for integrated circuit design. *Integration, the VLSI Journal*, 9(1):81–97, 1990.

[2] B. L. Hutchings. *A Cell-Based Approach to Hierarchical Interdomain VLSI Design.* PhD thesis, University of Utah, 1992.

[3] B. E. Nelson K. F. Smith. The PPL integrated circuit design methodology. *Computer Aided Design*, November 1986.

[4] B. E. Nelson, K. F. Smith, and T. M. Carter. Cost effective VLSI design system. In *IEEE International Symposium on Circuits and Systems*, May 1988.

Experiments on Technology Mapping Using Different Cell Libraries

Enrico Macii Massimo Poncino

University of Colorado
Dept. of Electrical and Computer Engineering
Boulder, CO 80309

Abstract - **Technology mapping is one of the major steps in the synthesis process of integrated circuits. Efficient algorithms have been proposed in the recent past to optimize the number of cells required to implement a given design, but not enough effort has been spent in studying the relationship between the quality of the circuit produced by the logic synthesis tool and the cell library used to perform the mapping. In this paper we present experiments on technology mapping that show how the design of the cell library can influence the area and the delay of the synthesized circuit.**

1 Introduction

A cell library is a library of simple (AND, OR, ...) and complex (MUX, AO, AOI, ...) devices implementing logic functions in a given technology. Cell libraries serve as a common back end for contemporary logic synthesis tools. The quality of a synthesized design depends on three interdependent components: the synthesis tool, the place and route tool, and the cell library used to perform the technology mapping. Each of these elements is constrained by the other two. In particular, in current usage, a synthesis tool must synthesize a design into cells provided by a particular library, and the place and route tool must place and route precisely the resulting net-list of cells produced by the synthesis tool. Thus, the quality of the cell library can considerably impact the final quality of the design.

In the recent past, a lot of work has been done in developing efficient algorithms for technology mapping [1, 2, 3]. However, the performance of technology mapping algorithms heavily depends on the characteristics of the input gate library. Consequently, design of cell libraries is crucial to obtain circuits which are fast, compact, and easily testable out of the overall logic synthesis process. In [4], Keutzer *et al.* discuss the relationship between the size, *i.e.*, the number of cells, of the library used by the technology mapper and the area of the resulting circuit; however, the synthesis tool used to produce the net-list is so dated that results may no longer be relevant. In [5], some directions for the design of cell libraries are given; the target, in this case, is the improvement of the speed of the synthesized network.

In this paper we present experiments on technology mapping performed using different cell libraries; all these libraries are built by incrementally adding various cells to the minimal `nand2-inv` library (*i.e.*, a library consisting only of two-input NAND gates and inverters). The goal we would like to accomplish is to show how the choice of the elements in the library strongly influences both the area and the speed of the circuit produced by the synthesis tool. For the experiments we used the technology mapper included in SIS [6, 7], and we built our libraries starting from the ones distributed with the MCNC'91 Logic Synthesis Benchmarks [8].

The rest of this paper is organized as follows. In Section 2 we describe the cell libraries we have used to perform our experiments, and the format utilized for their specification. Section 3 summarizes the characteristics of the benchmarks we have employed to compare the quality of the circuits produced by the synthesis tool. Section 4 and Section 5 present the operations (logic optimization and technology mapping) we have applied to each circuit in order to obtain the final version of the design. In Section 6 we report and discuss experimental results concerning both chip area and static delay of the synthesized circuits. Finally, Section 7 is devoted to conclusions.

2 Library Description

In our experiments we have used eight libraries of different size, (*i.e.*, number and type of cells in the library) whose characteristics are summarized in Table 1.

Name	*Cells*	*# of Cells*
lib1	nand2, inv	2
lib2	nand2, nor2, inv	3
lib3	nand2, nor2, xor2, xnor2, inv	7
lib4	nand2-nand4, nor2-nor4, inv	7
lib5	nand2, nor2, xor2, xnor2, mux2-1, inv	9
lib6	nand2-nand4, nor2-nor4, xor2-xor4, xnor2-xnor4, inv	20
lib7	nand2, nor2, xor2, xnor2, mux2-1, ao21, oa21, ao22, oa22 aoi21, oai21, aoi22, oai22, inv	17
lib8	nand2-nand4, nor2-nor4, xor2-xor4, xnor2-xnor4, mux2-1,mux4-1, ao21-ao4444, oa21-oa4444, aoi21-aoi4444, oai21-oai4444, inv	266

Table 1: Library Characteristics.

Following the library format employed in SIS (*i.e.*, the genlib format), for each cell we have specified the name, the area (in μm^2, defining the relative area cost of the cell), the logic function (as an equation written in the conventional algebraic notation using the operators + for OR, * for AND, ! for NOT, and parentheses for grouping), and the characteristics of each pin, that is, the pin name, the phase (INV for a logic function which is negative-unate, NONINV for a logic function which is positive-unate, and UNKNOWN for a logic function which is binate), the input load, the output load, the rise-block delay, the rise-fanout delay, the fall-block delay, and the fall-fanout delay. For the sake of simplicity in performing the experiments, we have assumed each cell to be symmetric, and to have a unit (in *nsec*) rise-block and fall-block delay.

As an example, the simple lib1 library, which contains only two-input NAND gates and inverters, is shown in Figure 1. Notice that the * following the PIN keyword indicates that all the pins of the cell have the same timing behavior. Notice also that the *Ground* pin (`zero`) and the V_{cc} pin (`one`) need always to be specified in every cell library.

```
#
# library: lib1.genlib
#
# cells: nand2
#        inv
#
# total number of cells (excluding Ground and Vcc): 2
#

GATE nand2      1392.0  O=!(a*b);       PIN * INV 1 999 1.0 0.2 1.0 0.2
GATE inv        928.0   O=!a;           PIN * INV 1 999 1.0 0.2 1.0 0.2
GATE zero       0       O=CONST0;
GATE one        0       O=CONST1;
```

Figure 1: The `lib1` Library.

In all the libraries, we have implemented the XOR, XNOR, and MUX cells in duplicate form, because this is what usually happens in commercial libraries. This means that, for example, a two-input XOR is implemented as both `O = a*!b+!a*b` and `O = !(a*b+!a*!b)`, being `a` and `b` the input pins of the cell.

3 The Benchmark Set

To perform our technology mapping experiments we have selected 7 circuits from each of the four categories of the MCNC'91 benchmark set [8]. The fundamental characteristics of these examples are presented in Table 2 (Combinational Two-Level Circuits), Table 3 (Combinational Multi-Level Circuits), Table 4 (Sequential Multi-Level Circuits), and Table 5 (Finite State Machines).

Each benchmark category has its own data representation, or format. Combinational Two-Level Circuits use SLIF (Structure Logic Interchange Format), Combinational Multi-Level Circuits use BLIF (Berkeley Logic Interchange Format), Sequential Multi-Level Circuits use Extended BLIF, and Finite State Machines use KISS2.

4 Logic Optimization

We have performed different levels of logic optimization on the four sets of benchmark circuits we have considered in our experiments.

Circuit Name	*Inputs*	*Outputs*	*Product Terms*
duke2	22	29	87
rd84	8	4	256
misex2	25	18	29
b12	15	9	431
cps	24	109	654
rd73	7	3	141
9sym	9	1	87

Table 2: Combinational Two-Level Circuit Characteristics.

Circuit Name	*Circuit Function*	*Inputs*	*Outputs*	*Approx. Gates*
C432	Priority Decoder	36	7	160
C1908	Error Correcting	33	25	880
C2670	ALU and Control	233	140	1193
C6288	16-bit Multiplier	32	32	2406
t481	Logic	16	1	2072
dalu	Dedicated ALU	75	16	1697
k2	Logic	45	45	1201

Table 3: Combinational Multi-Level Circuit Characteristics.

Circuit Name	*Circuit Function*	*Inputs*	*Outputs*	*Flip-Flops*	*Approx. Gates*
s298	PLD	3	6	14	119
s400	Traffic Light Controller	3	6	21	162
s444	Traffic Light Controller	3	6	21	181
s510	Controller	19	7	6	211
s713	PLD	35	23	19	393
s820	PLD	18	19	5	289
s1488	Controller	8	19	6	653

Table 4: Sequential Circuit Characteristics.

FSM Name	*Inputs*	*Outputs*	*Products*	*States*
cse	7	7	91	16
dk16	2	3	108	27
ex1	9	19	138	20
keyb	7	2	170	19
donfile	2	1	96	24
styr	9	10	166	30
tbk	6	3	1569	32

Table 5: FSM Characteristics.

On the Combinational Two-Level Circuits, Combinational Multi-Level Circuits, and Sequential Circuits we have executed the following sequence of SIS commands:

```
sweep
eliminate 5
simplify -m nocomp -d
resub -a
decomp -g *
```

`sweep` eliminates from the circuit all the single-input nodes and constant nodes (nodes which assume the constant logic value 0 or 1).

`eliminate 5` removes all the nodes in the circuit whose value is less than or equal to 5. The value of a node represents the number of literals saved in the literal count for the network by leaving the node in the network. If the value is less than (or equal to) 5, the node is eliminated by collapsing the node into each of its fanouts. Primary inputs and primary outputs are never eliminated.

`simplify -m nocomp -d` simplifies each node in the network using a full minimization procedure similar to ESPRESSO [9] without computing the complete off-set and without using any information about the don't-care set.

`resub -a` resubstitutes each node of the circuit into all the others, trying to use both the positive and negative phase of the node. The resubstitution operation uses algebraic division when substitutes one node into another, and it keeps looping until no more change in the circuit is observed. Notice that the division is performed on both the divisor and its complement.

`decomp -g *` decomposes all the nodes in the circuit. The decomposition operation factors the nodes and makes the divisor a new node within the network, re-expressing other nodes in terms of this newly introduced node. It is one of the transforms used to break down large functions into smaller pieces, usually at the cost of introducing a few more literals. The `-g` option allows the system to use the *good decomposition* algorithm, which successively extracts out the best kernel until the function is factor free, and applies the same algorithm to all the kernels just extracted. This operation gives the best algebraic decomposition for the nodes but, since it generates all the kernels at each step, it is pretty expensive in terms of CPU time.

On the FSM's, for which a better optimization is possible, we have run the SIS standard script file `script.algebraic`. In such a file, after running `sweep`, `eliminate 5`, and `simplify -m nocomp -d`, an interleaved series of `gkx` (extraction of multiple-cube common divisors from the network) with decreasing threshold, `resub -a`, and `sweep` is applied to the circuit before the final decomposition step (`decomp -g *`).

5 Technology Mapping

When the target was area optimization, the technology mapping step has been performed using the SIS command:

```
map -m 0 -AF
```

The `-m 0` option produces a minimum area circuit which takes into consideration the load limits specified in the library, by accurately controlling the cost function used for a simple version of the tree covering algorithm. The `-F` option performs fanout optimization. The `-A` option recovers area after fanout optimization at little or no delay cost by resizing buffers and inverters.

On the other hand, when the target was delay minimization, the technology mapping has been done using the SIS command:

```
map -n 1 -AFG
```

The `-n 1` option produces a minimum static delay circuit that respects the load limits imposed by the library used to perform the mapping, by allowing the access to a better tree covering algorithm. The `-F` and `-A` options produce the same effects as in the case of technology mapping for area minimization. Finally, the `-G` option recovers area after fanout optimization at no cost in delay by resizing all gates in the network.

6 Experimental Results

Table 6 and Table 7 show the results we have obtained on the circuits of Table 2, Table 3, Table 4, and Table 5, concerning both area and maximum static delay. The area is expressed in μm^2, the delay in *nsec*.

As expected, libraries of increasing size produce designs which are smaller and faster. However, it should be noticed that circuit area is much more sensitive to the characteristics of the cell library than circuit speed.

Another key observation is that not all the cells of a library have the same importance; for example, by looking at the results for `lib2`, `lib3`, and `lib4` one can notice that circuit areas get much more advantage from the presence in the library of multiple-input NAND/NOR gates than two-input XOR/XNOR cells (delays are less influenced).

Finally, it should be pointed out that increasing the number of cells from 17 (`lib7`) to 266 (`lib8`) does not always pay-off in terms of circuit speed, and can cause memory management problems in the technology mapper (see, for example, the largest Combinational Multi-Level Circuits); on the other hand, area results substantially improve with this large variety of cells available.

Notice that for the sequential circuits we have considered only the combinational portion of the logic, leaving the memory elements out of the area estimation; this is because no accurate information on the area of the flip-flops can be determined using SIS.

Example	`lib1`		`lib2`		`lib3`		`lib4`	
	Area	*Delay*	*Area*	*Delay*	*Area*	*Delay*	*Area*	*Delay*
duke2	1386896	24.6	1107568	22.0	1107568	22.0	896912	20.0
rd84	534528	27.0	462144	27.2	462144	27.2	402752	27.2
misex2	335472	16.8	238032	10.6	238032	10.6	187920	9.6
b12	165648	14.2	145232	12.8	145232	12.8	122960	10.8
cps	4825136	37.2	3616880	31.2	3616880	31.2	2730640	27.4
rd73	363312	25.0	310416	18.4	310416	18.4	254736	14.4
9sym	489520	24.4	429200	20.0	429200	20.0	372592	16.8
C432	551232	64.4	410176	47.0	410176	47.0	348000	36.0
C1908	1152576	43.6	965120	41.2	965120	41.2	850976	39.8
C2670	1368800	39.2	1203616	35.6	1203616	35.6	1140512	30.2
C6288	4938352	177.2	4521680	174.2	4521680	174.2	4493840	171.2
t481	5507216	46.4	3737520	29.6	3737520	29.6	2515344	20.4
dalu	3575584	59.8	3020640	52.6	3020640	52.6	2530656	46.6
k2	6158208	47.2	3995968	30.8	3995968	30.8	2524160	21.2
s298	251024	19.0	216688	12.8	216688	12.8	176784	11.6
s400	382336	23.8	318304	16.0	318304	16.0	268192	12.6
s444	398576	20.4	330832	15.8	330832	15.8	282576	14.6
s510	599952	22.0	490448	17.2	490448	17.2	388368	14.6
s713	380944	47.4	299280	35.4	299280	35.4	255664	29.2
s820	815248	24.6	650064	20.2	650064	20.2	516432	17.8
s1488	1510784	28.8	1265792	21.0	1265792	21.0	1003168	17.6
cse	398112	20.4	296032	15.2	296032	15.2	227360	11.4
dk16	589744	22.6	439408	16.6	439408	16.6	327120	12.8
ex1	532672	22.8	382336	15.8	382336	15.8	307168	13.0
keyb	561904	26.2	406928	18.0	406928	18.0	301136	13.8
donfile	390224	19.6	296496	14.6	296496	14.6	215760	10.8
styr	1041216	27.8	750752	19.4	750752	19.4	570720	14.8
tbk	1709376	30.4	1199904	21.8	1199904	21.8	875104	15.8

Table 6: Area and Delay Results.

Example	lib5		lib6		lib7		lib8	
	Area	*Delay*	*Area*	*Delay*	*Area*	*Delay*	*Area*	*Delay*
duke2	1107568	22.0	889488	20.0	880208	13.6	646816	12.6
rd84	462144	27.2	402752	27.2	344288	15.0	286284	14.2
misex2	238032	10.6	187920	9.6	221792	9.8	156368	7.6
b12	145232	12.8	122960	10.8	104864	8.2	89552	7.6
cps	3616880	31.2	2730640	27.4	3055440	19.2	2173364	18.0
rd73	310416	17.4	254736	17.4	229216	12.2	184668	10.6
9sym	431984	20.2	374448	17.4	302528	13.8	259372	12.6
C432	410176	47.0	348000	36.0	346144	37.0	256120	28.4
C1908	965120	41.2	850976	35.0	763744	24.4	654232	22.4
C2670	1203616	35.6	1129376	30.0	825456	24.2	776736	22.0
C6288	4521680	174.2	4493840	171.2	3820576	154.0	3740768	–
t481	3737520	29.6	2497712	20.4	3330592	25.4	2080973	–
dalu	3020640	52.6	2530656	46.6	2192400	36.4	1759488	–
k2	3995968	30.8	2525088	21.2	3919872	30.4	2474044	–
s298	203696	12.8	176784	11.6	167504	10.8	139664	9.6
s400	302528	15.0	268192	12.6	263088	12.0	215760	9.0
s444	318768	14.2	282576	14.6	264944	11.2	231536	9.2
s510	483952	17.0	388368	14.6	366560	13.2	273288	11.0
s713	299280	35.4	255664	29.2	254272	20.4	215760	17.0
s820	650064	20.2	516432	17.8	522928	14.6	380008	12.6
s1488	1266720	21.0	1003168	17.6	928000	14.6	702480	13.8
cse	296032	15.2	227360	11.4	254272	15.2	182348	11.4
dk16	439408	16.6	327120	12.8	393936	16.6	291852	12.8
ex1	382336	15.8	307168	13.0	367024	15.8	283968	12.8
keyb	406928	18.0	301136	13.8	373984	18.0	251946	13.4
donfile	296496	14.6	215760	10.8	258912	14.0	175856	10.8
styr	750752	19.4	570720	14.8	681616	19.6	510864	14.8
tbk	1199904	21.8	875104	15.8	1074160	21.8	741447	18.2

Table 7: Area and Delay Results (continued).

7 Conclusions

In this paper we have presented experimental results on technology mapping of both combinational and sequential circuits performed with cell libraries of different size. The purpose of the experiments is to point out how the choice of the cell library can influence the quality of the design produced by the synthesis tool. The results we have shown may provide a useful guideline to the users of logic synthesis tools when they have to choose the cell library to be adopted for the implementation of a specific design.

Note

The authors are also with the Politecnico di Torino, Dip. di Automatica e Informatica, Torino, ITALY 10129.

References

[1] R. Lisanke, F. Brglez, G. Kedem, "McMap: A Fast Technology Mapping Procedure for Multi-Level Logic Synthesis", *ICCD'88: IEEE International Conference on Computer Design*, pp. 252-256, Cambridge, MA, October 1988.

[2] C. W. Moon, B. Lin, H. Savoij, R. K. Brayton, "Technology Mapping for Sequential Logic Synthesis", *IWLS'89: International Workshop on Logic Synthesis*, Research Triangle Park, NC, May 1989.

[3] H. Touati, C. W. Moon, R. K. Brayton, A. Wang, "Performance Oriented Technology Mapping", *6th MIT Conference on VLSI*, pp. 79-97, Cambridge, MA, April 1990.

[4] K. Keutzer, K. Kolwicz, M. Lega, "Impact of Library Size on the Quality of Automated Synthesis", *ICCAD'87: IEEE International Conference on Computer Aided Design*, pp. 120-123, Santa Clara, CA, November 1987.

[5] K. Keutzer, K. Scott, "Improving Cell Libraries for Synthesis", *IWLS'93: International Workshop on Logic Synthesis*, Lake Tahoe, CA, May 1993.

[6] E. M. Sentovich, *et al.*, *SIS: A System for Sequential Circuits Synthesis*, Technical Report Memorandum No. UCB/ERL M92/41, University of California at Berkeley, May 1992.

[7] E. M. Sentovich, K. J. Singh, C. W. Moon, H. Savoij, R. K. Brayton, A. Sangiovanni-Vincentelli, "Sequential Circuits Design Using Synthesis and Optimization", *ICCD'92: IEEE International Conference on Computer Design*, pp. 328-333, Cambridge, MA, October 1992.

[8] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide, Version 3.0*, Microelectronics Center of North Carolina, Research Triangle Park, NC, January 1991.

[9] R. K. Brayton, G. D. Hachtel, C. T. McMullen, A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic Publishers, Boston, MA, 1984.

omit

MOSIS Integrated Circuit Wafer Fabricator Selection, Qualification and Monitoring

Vance C. Tyree
USC/Information Sciences Institute
Marina del Rey CA 90292-6695
Internet: tyree@mosis.edu

Abstract - **MOSIS is an ARPA sponsored prototyping and low volume production service for custom and semicustom integrated circuits. Layout design files are accepted via E-mail or magnetic tape, fabricated at commercial foundries and returned as packaged parts to users in approximately 8 weeks.**

Currently, the MOSIS service provides fast turnaround fabrication of integrated circuits in prototype or small production quantities to over 200 ARPA and NSF sponsored organizations as well as to approximately 150 DoD contractors and commercial customers.

Wafer fabrication is provided at regular intervals defined in a published schedule. Submitted ASIC designs (which may be designed in the MOSIS scaleable rules, DoD scaleable rules or the fabricator's rules) are merged into a single mask set for wafer fabrication to optimally utilize wafer area, which is an expensive resource. The cost to designers for prototype quantities of ASIC's is reduced because the wafer fabrication is cost is shared amongst multiple designers. An additional value added by the MOSIS service is rapid turn-around, careful fabricator qualification and extensive quality monitoring. This presentation will focus upon the vendor qualification procedures and the methods used in quality monitoring with samples of the results of this monitoring.

Fabricator qualification procedures involve fabricating a qualification wafer lot in which the entire wafer lot is devoted to a combination of parametric, functional (for yield assessment) and reliability monitoring test structures. In addition, there are a few ”invited” designs that have been successfully fabricated on other wafer lots at other fabricators which are used to test transportability of designs from a ”typical” designer. To qualify, a fabricator must produce a test wafer lot that meets their own parametric specifications (which have already been compared with MOSIS expectations) and that has a minimum yield of the MOSIS scaleable yield monitor. The yield monitor is designed to implement the highest circuit density permitted by the MOSIS scaleable design rules. The qualification wafer lot also contains a number of wafer level reliability testing calibration structures that are used to begin a reliability data base on the fabricator.

A qualified fabricator is monitored in every wafer lot fabricated for MOSIS using a subset of the qualification run parametric and reliability test structures. Data from these test structures are available to the fabricator and the designers. The parametric test data is used to accept or reject the wafer lot and to select the best wafers from the lot for distribution to the designers. Currently the reliability data is only used for information with no lot jeopardy.

Experience that MOSIS has gained from more than 12 years of CMOS wafer monitoring has given great insight into how foundry sourced wafers can be monitored with sufficient confidence that a QML certified fabricator could be a source of high reliability ASIC devices for space and other critical applications. Test structures developed at MOSIS and at JPL (by Martin Buehler, et al.) will provide the basis for confident assessment of both quality and reliability.

omit

Session 10
Design Techniques II

John Purviance

Reliable Interface Design for Combining Asynchronous and Synchronous Circuits

Lüli Josephson   Erik L. Brunvand   Ganesh Gopalakrishan   John F. Hurdle
Computer Science Department
University of Utah
Salt Lake City, Utah 84112
E-mail: {luli,elb,ganesh,hurdle}@cs.utah.edu

Abstract - **In order to successfully integrate asynchronous and synchronous designs, great care must be taken at the interface between the two types of systems. Synchronizing asynchronous inputs with a free running clock can cause well-known problems with metastability in the synchronization circuits. Stretchable clocks allow a clock cycle to expand dynamically in response to the metastability effects of sampling asynchronous inputs. We use an interface organization where the special circuitry for detecting metastability and for stretching the clock that is delivered to the synchronous part of the system is encapsulated in a Q-flop-based interface. This provides a very convenient method for interfacing mixed systems, as the interface and clock generation circuitry are isolated into one special module, and neither the asynchronous nor the synchronous system need be modified internally to accommodate the interface. This is especially important when standard synchronous components are used as there is no opportunity to modify these parts. We show that this interface module is suitable for most mixed design needs and conclude with an example.**

1 Introduction

As VLSI technology improves, hardware systems become larger, faster, and more complex. Along with these improvements, however, come many problems directly associated with the speed and scale of the new systems. Asynchronous and self-timed design techniques are currently attracting renewed interest as a method for coping with some of the problems associated with the scale of modern systems. These systems that do not rely on a global clock to keep system components synchronized have been shown to exhibit a number of inherent advantages: more robust behavior (in terms of process and environmental variations), a capability for higher performance operation, decreased power consumption, and inherently higher reliability in high speed applications. However, for many applications, designing completely asynchronous circuits is currently an impractical approach. Because of the large body of work related to synchronous system design, and the many tools available for this style of design, mixed asynchronous-synchronous design is an important approach.

One domain where this mixed design style may be especially appropriate is in embedded systems and controllers. These systems typically use a commodity-type synchronous controller, but respond to a variety of external inputs that are not synchronized to the system clock. Processing the asynchronous input data as it arrives may be best accomplished using a specialized asynchronous circuit with overall system response being coordinated by the synchronous controller. When these devices are used in remote locations there is a real need for low power control circuits that are extremely robust under significant temperature and power supply variation. These types of systems present an inherent opportunity for highly parallel control interaction that must operate correctly and whose behavioral parameters are well proven and understood, due to the extreme remote nature of their operation and the resultant expense of repair.

In order to successfully integrate asynchronous and synchronous designs, great care must be taken at the interface between the two types of systems. Synchronizing asynchronous inputs with a free running clock can cause well-known problems with metastability in the synchronization circuits. To achieve this interfacing successfully, we implement an interface/clock generation module based on Q-flops, special flip-flops that signal when they have exited metastability, and stretchable clocks. Stretchable clocks allow a clock cycle to expand dynamically in response to the metastability effects of sampling asynchronous inputs. In our scheme the special circuitry for detecting these events and for stretching the clock that is delivered to the synchronous part of the system is encapsulated in a Q-flop-based interface, as an extension of Rosenberger's Q-Module organization [24]. This provides a very convenient method for interfacing mixed systems as the interface and clock generation is isolated into one special module, and neither the asynchronous nor the synchronous system need be modified internally to accommodate the interface. This is especially important when standard synchronous components are used as there is no opportunity to modify these parts.

In spite of the natural inertia from the mass of existing synchronous designs, there is currently a renaissance in asynchronous circuit design. As one aspect of our asynchronous systems design work, we look for ways to exploit the rediscovered advantages of asynchronous designs while capitalizing on the preponderance of previously designed synchronous parts. One approach involves designing globally asynchronous, locally synchronous systems (hereafter called "mixed systems"). We use Q-module-based interface/clock generation elements as shown in Figure 1 and described in Section 3 to safely coordinate and synchronize interactions between the different regions in these mixed systems. Unlike other value-reliable [9] synchronization circuits such as stretchable or stoppable clock methods, Q-modules (internally clocked, delay insensitive circuits) conform naturally to the self-timed asynchronous design style that we advocate in our asynchronous systems research.

One goal for this work is to devise an interface scheme that will allow us to use previously designed synchronous parts "as is". (We consider this category of synchronous parts to include off-the-shelf parts, design library modules, and modules synthesized from a behavioral specification by a high-level synthesis system.) This allows us to focus our efforts on the (to us) more interesting asynchronous design aspects yet avail ourselves as necessary of existing synchronous circuits.

The asynchronous circuits we consider in this paper are all built using two-phase delay-insensitive (DI) control circuits (the individual control modules may be speed independent, but exhibit delay-insensitive interfaces) and bundled data paths.

This paper describes our approach to building globally asynchronous, self-timed systems that incorporate locally synchronous regions. First we discuss mixed-system design issues, then review typical synchronization approaches and list their limitations. Next we present a Q-module-based interface implementation which combines synchronization and clock gen-

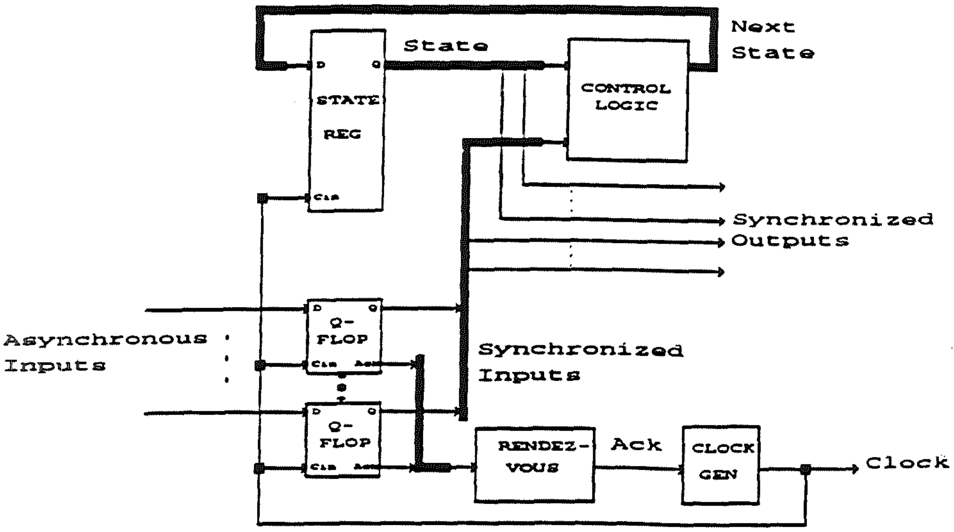


Figure 1: Q-module organization.

eration and facilitates the mixed-system design style we are advocating. We follow this by an example using the interface in a mixed asynchronous-synchronous implementation for a Finite Input Response (FIR) filter. Finally, we draw some conclusions about the method and indicate future extensions to this work.

2 Mixed Systems - Combining Asynchronous and Synchronous Regions

As system sizes and circuit technology speeds increase, it becomes more difficult to satisfy the high frequency clock requirements of robust, hazard- and race-free clock schemes and small skew global clock distribution [3]. Using autonomous locally clocked and/or asynchronous subsystems can facilitate large system design. As a corollary, global communication requirements are reduced since most communications can be localized. However, communications among synchronous systems not sharing a common clock must be synchronized.

2.1 Synchronization Issues in Mixed Systems

In synchronous systems, whenever flip-flop timing constraints such as minimum pulse width, setup, or hold times are not met, there is a finite probability that instead of resolving to a state matching the input value, the flip-flop will enter a metastable state. This metastable state may eventually resolve to the correct output state, or it may resolve to an incorrect output state. If it takes longer than one clock period for the metastability to resolve, and the synchronous system is allowed to proceed, the metastable output may be interpreted as a "1" by part of the system and "0" by the rest. We refer to this effect as "value failure".

Metastability occurrence is probabalistic and exponentially related to the synchronous clock rate and the asynchronous data rate: the mean time between failure is shorter at high asynchronous data rates and high fast synchronizing clock frequencies. Conversely, the longer one waits before using the sampled data, the lower the probability of synchronizer failure. Conventional techniques for decreasing the probability of metastable failure [17, 20] include using fast devices, allowing for extended settling times, and employing masking and redundancy. These techniques all amount to allowing a long but bounded amount of time for metastability to resolve, accepting the finite probability that failures will occur. Thus these methods trade the advantage of time-certainty for the possibility of value failure.

2.2 Conventional Interface Techniques

Because of its probabilistic nature, metastability is hard to characterize, detect, and reproduce. As circuit technologies improve and circuit speeds increase, handling metastability issues becomes more important. Although the consequences of ignoring its effects are potentially catastrophic, it is still considered a very difficult problem to treat correctly in the design of interface circuits. Evidence for this is the number of interface specification systems that completely disregard this issue. For example, a range of interface specification methods are being developed in conjunction with several high-level synthesis systems. The implementations synthesized from these descriptions either include conventional synchronizers on the asynchronous inputs [2, 4, 13], or only handle signals related to the same clock internally or externally [12, 18, 21]. For high-speed, value-critical system interfaces, these methods cannot guarantee the required combination of value-reliability plus speed.

2.3 Reliable Value Synchronization Methods

Alternatively, less conventional methods using stoppable or stretchable clocks trade a degree of time-uncertainty for value-reliability. *Stoppable* clocks [9, 22, 25, 27, 28, 31] can be used for interfacing asynchronous-synchronous circuits communicating via specific protocols and when the synchronous system does not require a continuously running clock. The locally generated clock is stopped synchronously, and the system passively awaits the next asynchronous input. The clock is then re-started asynchronously by the asynchronous input with *known* value but *unknown* arrival time. For circuits that operate using this restricted interface model, this scheme avoids metastability because it avoids sampling a changing signal. However, stoppable clock circuits require careful delay analysis in their design to ensure correct circuit operation [9, 31]. Often redesign of the synchronous part itself is required as well. Thus stoppable-clock based interface schemes are applicable only in certain restricted cases.

Stretchable clock circuits can be used in the more general case when neither the value nor the arrival time of the incoming asynchronous signal is known in advance. In stretchable clock circuits [9, 10, 22, 28], a comparator-based metastability detector is used to delay the next clock edge when a potentially changing input is sampled. Again, critical timing conditions must be met in the design and operation [24].

3 Q-Module-Based Interfaces

The goal is to build value-safe interfaces for combining synchronous and asynchronous designs. The main difficulty with the stoppable/stretchable clock based approaches is that they require specialized and somewhat hard-to-build circuits. Yet interfaces based on these methods would facilitate the decoupled mixed-design style we are advocating, allowing us to incorporate synchronous parts into an asynchronous design without redesign of the synchronous parts. Because they are the more general of the two schemes, we focus primarily on stretchable clock techniques for our interfaces.

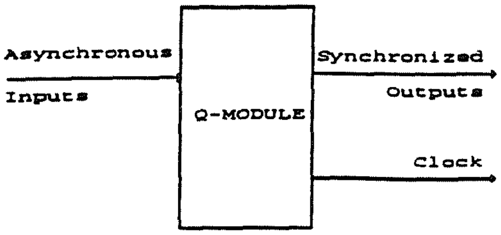


Figure 2: Q-Module-based interface.

We implement a stretchable clock interface where the metastabillty detection circuitry is encapsulated in a special module called a Q-module, shown in Figure 1 and described in Rosenberger, et al., [24]. Q-modules are internally clocked, delay-insensitive state machines. Data inputs are sampled on one clock event using Q-flops: these special flip-flops use comparator-based circuits to acknowledge when metastability has resolved and values are stable. The acknowledgements from the Q-flops and the state registers are used to generate the next clock event.

We extend the Q-module organization by altering the Clock Generation element to provide the appropriate clock signal to the synchronous side of the interface. Note that in the normal case the period of this exported clock signal is regular and set by the delay through the clock generator element (with any additional delay needed to model the combinational delay time, or to achieve a particular minimum clock frequency). If the internal Q-flops stall, waiting to resolve metastability due to asynchronous inputs, the clock signal will be stretched automatically until the data resolve.

In contrast to conventional stretchable clock implementations, because Q-flops acknowledge when they are ready to accept new inputs as well as acknowledging when the storage operation is completed, the clock generation utilizes a delay-insensitive protocol. Clock generation and distribution are also simpler: delay through the combinational logic implementing the state function provides the single delay constraint that must be satisfied in the clock generation and distribution for the Q-module itself. For our interfaces, we use Q-modules both to synchronize the asynchronous inputs and to act as clock generators. The internal two-phase single wire clock, used for internal sequencing, is exported to form the synchronous part clock input, as is shown in Figure 2. Thus we have to adjust the single delay constraint in the Clock Generation element to provide the minimum clock frequency required by the synchronous part.

The goal for this work is to demonstrate an interfacing method to allow value-reliable synchronization between self-timed asynchronous and conventional synchronous systems. In contrast to the stretchable and stoppable clock methods discussed above, Q-modules provide the safety, generality, and simplicity required of an easily designed interface. Our Q-module interface/clock generation circuits avoid synchronization failure by ensuring that storage elements have resolved before using the stored values by dynamically adjusting the additional delay.

Contrast this with the fixed time delay that is normally introduced for synchronization in conventional clocked circuits (for example, by including extra latching stages or dividing down the clock for sampling latches). In the Q-module-based interface, value-safety is assured at the expense of an occasional stretching in the clock period. In conventional clocked circuits, the clock period must always be long enough to allow sufficient time for metastability to resolve with as high a probability as possible. Yet because the resolution time is fixed and bounded in conventional techniques, there is always a finite probability of value failure and subsequent circuit failure.

4 An Example Mixed-System Design

Now we discuss a relatively small example - a mixed asynchronous-synchronous circuit implementation for an FIR filter. This circuit incorporates a Q-module-based interface for value-safe synchronization and clock generation. This circuit contains all the typical elements of a larger mixed-system design, but is simple enough to allow us to focus on the important interface details. For example, synchronous arithmetic parts are inexpensive, generally compact, and readily available. Their use can be very cost effective in a circuit which is otherwise asynchronous. The FIR filter circuit below is designed to operate in an asynchronous environment and so it must present an asynchronous interface to the external world. However, in addition to the asynchronous data buffering parts, it uses synchronous computation parts internally and so requires synchronization at that internal interface. The asynchronous control structure will allow the data-dependent processing time of the required arithmetic calculations to enhance the average-case performance of the filter.

The circuit described represents a class of circuits where mixed design is appropriate. For example, embedded control using standard synchronous controllers responding to asynchronous events are an excellent target for the synchronization strategy we describe here.

4.1 Circuit Description

We implement a universal FIR filter for asynchronous inputs. This is a sixth order filter, with input and output done in parallel. The symmetric coefficient algorithm is used which reduces the number of multiplications by a half over the standard filter. For the 4-bit input vector x_n and for three 4-bit coefficient vectors k_a, k_b, k_c, the 11-bit output vector y_n is computed as:

$$y(n) = k_a * [x_n + x_{n-5}] + k_b * [x_{n-1} + x_{n-4}] + k_c * [x_{n-2} + x_{n-3}]$$

The FIR filter circuit as shown in Figure 3 consists of an asynchronous data-fetching unit ASYNCH and a synchronous computation unit SYNCH. A Q-module-based interface part

ASYNCH-TO-SYNCH INTERFACE between the asynchronous module and the synchronous module links the control of both parts, synchronizes the data, and generates the clock for the synchronous part. The external interface is self-timed using two-phase transition signaling and data-bundling. The environment issues a Req when the next input data value is ready. The FIR returns an Ack when the FIR computation completes and a new output value is available. The asynchronous self-timed interface constrains event sequencing rather than event timing. The environment can supply data and use results at any rate as long as the sequence of "send-Req; receive-Ack" is obeyed.

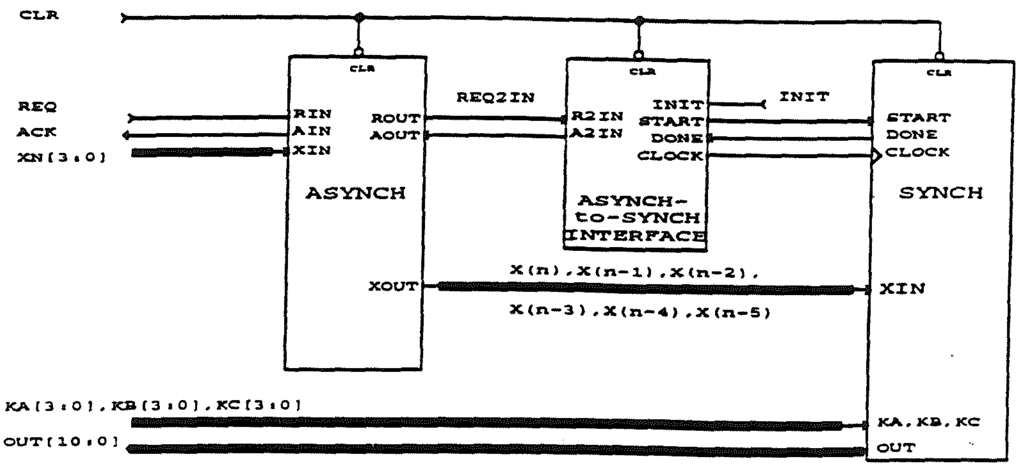


Figure 3: Top-level diagram for mixed asynchronous-synchronous FIR implementation.

The data-fetching part ASYNCH is an asynchronous shift register with 6 data taps implemented as a Sutherland micropipelined-based FIFO [29]. It uses two-phase self-timed control with data-bundling.

The Q-module-based interface part ASYNCH-TO-SYNCH INTERFACE, Figure 4, provides synchronized asynchronous data and the clock for use by the synchronous computation unit. A Q-flop is used to detect metastability when latching the "begin next computation" control signal REQ2IN is received from the asynchronous part. The clock generation circuit waits for metastability to resolve before initiating the next clock pulse and latching the outputs. The clock is exported to the synchronous part and is continuously running. While the maximum period may vary, the minimum clock period to match the synchronous part delay is guaranteed by the delay in the clock generation circuit.

Since the asynchronous interface communicates via a two-phase protocol, and the synchronous interface obeys a four-phase protocol, the interface circuit includes a 2-to-4 phase protocol converter part. This asynchronous converter circuit takes the place of the combinational logic implementing the state function in a regular Q-module. For this very simple interface. no additional logic or state-holding Q-flops are required. This is not usually the case. We are working on more complicated examples where state information and feedback within the Q-module are required [11, 16].

The synchronous computation unit SYNCH consists of five adders and three synchronous multipliers. The multipliers have data-dependent completion times, so the number of cycles

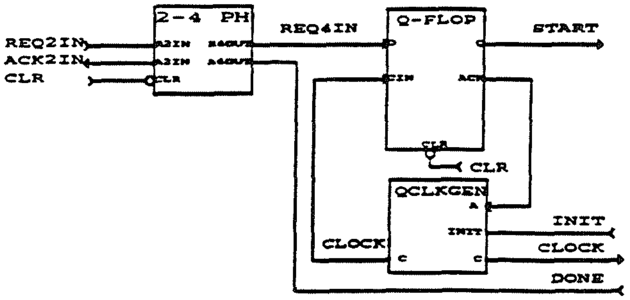


Figure 4: Asynch-to-Synch Interface Implementation

for each FIR computation will vary. Output data are latched, and so remain valid until the next FIR computation starts. For historical reasons, this module obeys a four-phase synchronous self-timed protocol.

4.2 Implementation Details

A prototype version of the mixed asynchronous-synchronous FIR filter has been designed for implementation in Actel FPGAs [1]. It was built and functionally simulated using the asynchronous macro cells described in [6] and the synchronous parts provided as Actel Macro Library soft macros, and contains 640 logic modules.

Q-flops used in the interface are not available as standard digital parts. Although Q-flops cannot be built using FPGAs because it is not possible to implement the required analog comparator circuit, we have designed circuits that mimic the behavior of a Q-flop using Actel FPGAs [6]. These FPGA circuits use a fixed delay to resolve metastability [1] and so are not value-safe, but they allow us to prototype designs quickly that can then be upgraded to use custom Q-flops as required.

We have built and tested actual Q-flops in a variety of technologies including CMOS [5, 11], and Gallium Arsenide [7]. The next step is to implement the asynchronous parts of this interface design in another technology where the actual analog metastability resolver parts can be used.

5 Results and Future Work

Q-flops are found in several circuit libraries [5, 7, 8, 19, 24, 26]. Combining Q-flops with control logic to form delay-insensitive state machine modules (Q-modules) has been extensively described by Rosenberger, et al., in [24]. Sutherland and Sproull suggest such an asynchronous state machine built using Q-flops as an asynchronous replacement for a conventional register-based synchronizer plus synchronous state machine in a three port memory controller in [30]. In this paper, we introduce a Q-module-based interface circuit for value-safe synchronization and clock generation. We demonstrate its use in a mixed asynchronous-synchronous FIR circuit implementation.

In general, these interface modules can be used in mixed systems whenever synchronization and clock generation are required and the synchronous element can tolerate some occasional clock elongation. Value-safety is assured. Chapiro [9] has done studies that indicate clock speeds can actually be increased, now that synchronization delay can be "dynamically adapting". In contrast to other recent work [2], no redesign of the synchronous part is required to accommodate it in the globally asynchronous self-timed environment. This makes the method especially suitable for correctly including previously designed synchronous parts in mixed systems.

There are many kinds of mixed systems where these interfaces can be used. This paper describes one example circuit containing asynchronous and synchronous regions that need to synchronize. Traver [31] describes a method for interfacing multiple locally-synchronous but mutually-asynchronous circuits. Synchronization problems arise in embedded controllers using synchronous circuits but responding to asynchronous events. These types of systems are often installed in remote locations and must be robust under a variety of environmental conditions that tend to make synchronization problems worse. Our technique could be used in both of these applications.

The FIR is a small mixed-system demonstration circuit. As asynchronous systems evolve in size, such an approach will become a necessity until suitable asynchronous computation components are commonly available. One system-size example we are working on in our lab [16] is interfacing a synchronous off-the-shelf math coprocessor chip to a fully self-timed asynchronous microprocessor [23]. Yet another example involves connecting synchronous analog-to-digital input converters and digital-to-analog output converters to an asynchronous neural net classifier system [14, 15] also being constructed in our lab. The issues involved in successfully combining more than two systems using these interfaces are being explored, as described in [11].

For a variety of reasons, including limitations in current asynchronous design tools, designing completely asynchronous circuits may not always be practical or desirable. With the addition of the interface method described here, the self-timed circuit design style we advocate allows us to design and implement globally asynchronous designs that incorporate correctly synchronized locally synchronous regions. Mixed systems retain many of the high-level advantages of asynchronous designs, while exploiting existing synchronous designs.

References

[1] Actel Corporation. *ACT Family Field Programmable Gate Array Databook*, March 1991.

[2] Naser Awad and David Smith. Automatic interfacing of synchronous modules to an asynchronous environment. In *International VLSI Conference*, 1991.

[3] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley, 1990. Chapter 8.

[4] Gaetano Borriello. Specification and synthesis of interface logic. In R. Camposano and W. Wolf, editors, *High-Level VLSI Synthesis*, chapter 7, pages 153–177. 1991.

[5] Erik Brunvand. Parts-r-us. *a chip aparts(s)....* Technical Report CMU-CS-87-119, Carnegie Mellon University, May 1987.

[6] Erik Brunvand. A cell set for self-timed design using actel FPGAs. Technical Report UUCS–91–013, University of Utah, 1991.

[7] Erik Brunvand, Nick Michell, and Kent Smith. A comparison of self-timed design using FPGA, CMOS, and GaAs technologies. In *International Conference on Computer Design*, Cambridge, Mass., October 1992.

[8] Erik Brunvand and Robert F. Sproull. Translating concurrent programs into delay-insensitive circuits. In *ICCAD-89*, pages 262–265. IEEE, November 1989.

[9] Daniel M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems.* PhD thesis, Department of Computer Science, Stanford University, October 1984.

[10] Daniel M. Chapiro. Reliable high-speed arbitration and synchronization. *IEEE Transactions on Computers*, C-36(10):1251–1255, October 1987.

[11] Ganesh Gopalakrishnan and Lüli Josephson. Towards amalgamating the synchronous and asynchronous styles. In *Tau 93: 1993 Workshop on Timing Issues in Specification and Synthesis of Digital Systems*, September 1993.

[12] M.R. Greenstreet and K. Li. Simple hardware for fast interprocessor communication. Technical Report CS-Tr-242-90, Princeton University, January 1990.

[13] S. Hayati and A. Parker. Automatic production of controller specifications from control and timing behavioral descriptions. In *Proceedings of the 26th ACM/IEEE Design Automation Conference*, pages 75–80, 1989.

[14] John F. Hurdle. Self-timed neural model implementation: an example using CMAC. In *Proceedings of the 26th Hawaii International Conference on Systems Science*, pages 369–378, January 1993.

[15] John F. Hurdle, Lüli Josephson, Erik L. Brunvand, and Ganesh Gopalakrishnan. Asynchronous models for large scale neurocomputing applications. In Jean-Claude Rault, editor, *Neural Nimes 92: Proceedings of the Fifth International Conference on Neural Networks and their Applications.* 1992.

[16] Lüli Josephson. Mixed asynchronous and synchronous circuits. Master's thesis, Computer Science Department, University of Utah, 1993. In preparation.

[17] Lindsay Kleeman and Antonio Cantoni. Metastable behavior in digital systems. *IEEE Design and Test of Computers*, pages 4–19, December 1987.

[18] David Ku. *Constrained Synthesis and Optimization of Digital Integrated Circuits from Behavioral Specifications.* PhD thesis, Department of Computer Science, Stanford University, June 1991.

[19] C. E. Molnar, T. P. Feng, and F. U. Rosenberger. Synthesis of dely-insensitive modules. In *Proc. 1985 Chapel Hill Conference on Very Large Scale Integration.* Computer Science Press, March 1985.

[20] Bryon I. Moyer. Has the metastability world stabilized? In *Proceedings of the Second Annual PLD Design Conference and Exhibit*, 1992. April.

[21] J. A. Nestor and D. E. Thomas. Behavioral synthesis with interfaces. In *IEEE International Conference on Computer-Aided Design, ICCAD-86: Digest of Technical Papers*, pages 112–115, 1986.

[22] Miroslav Pěchouček. Anomalous response times of input synchronizers. *IEEE Transactions on Computers*, C-25(2):133–139, February 1976.

[23] William F. Richardson and Erik Brunvand. The NSR processor prototype. UUCS Tech Report UUCS-92-029.

[24] Fred U. Rosenberger, Charles E. Molnar, Thomas J. Chaney, and Ting-Pein Fang. Q-modules: Internally clocked delay-insensitive modules. *IEEE Transactions on Computers*, 37(9):1005–1018, September 1988.

[25] Charles L. Seitz. System timing. In C. Mead and L. Conway, editors, *Introduction to VLSI Systems*, chapter 7. Addison Wesley, Reading, MA, 1980.

[26] Robert F. Sproull. Qflop modules. Technical Memo 4472, Sutherland, Sproull and Associates, June 10, 1986.

[27] Robert F. Sproull and Ivan E. Sutherland. Stoppable clock. Technical Memo 3438, Sutherland, Sproull and Associates, Jan. 9, 1986.

[28] M.J. Stucki and J.R. Cox, Jr. Synchronization strategies. In *Proceedings of the Caltech Conference on VLSI*, pages 375–393, January 1979.

[29] Ivan Sutherland. Micropipelines. *CACM*, 32(6), 1989.

[30] Ivan E. Sutherland and Robert F. Sproull. Three port memory. Technical Memo 4497, Sutherland, Sproull and Associates, Sept 1, 1986.

[31] Cherrice Ann Traver. A testable model for stoppable clock ASICs. In *Proceedings of the Fourth Annual IEEE International ASIC Conference and Exhibit*, pages P6–3.1, 1991.

Design Methods for Fault-Tolerant Finite State Machines[†]

Shailesh Niranjan and James F. Frenzel
Department of Electrical Engineering
University of Idaho, Moscow, ID 83844-1023
Office: 208-885-6554 E-mail: j.frenzel@ieee.org

Abstract — **VLSI electronic circuits are increasingly being used in space-borne applications where high levels of radiation may induce faults, known as single event upsets. In this paper we review the classical methods of designing fault-tolerant digital systems, with an emphasis on those methods which are particularly suitable for VLSI-implementation of finite state machines. Four methods are presented and will be compared in terms of design complexity, circuit size, and estimated circuit delay.**

1 Introduction

Fault tolerance in recent years has become an essential feature in the design of many computing systems. Very Large Scale Integrated (VLSI) circuit technology with its attractive features of increased circuit density and low power consumption has been used in the design of modern electronic equipment and is especially appealing in fault tolerant system designs. As the limits of VLSI technology are pushed towards submicron levels the probability of errors being introduced due to fabrication, wear, and environmental factors increases. In this paper we concentrate on radiation induced errors caused by high energy particles, referred to as single even upsets, or SEU's. This has a serious consequence on the operation of VLSI circuits used in space-borne applications as they are continuously bombarded by high energy cosmic ions.

The SEU's are generally transient in nature, temporarily altering the voltage level at the circuit node where the particle strikes. However, when the affected node belongs to a storage element, such as a memory cell or flip-flop, the fault may change the contents of that element and become permanent in the sense that it may affect the outcome of operations performed later in time. Sequential state machines used as controllers in VLSI circuits contain memory cells thus making them susceptible to SEU induced errors.

Different methods have been used to reduce the occurrence of SEU-induced errors, such as radiation-immune storage elements, radiation-immune processes, and conventional fault-tolerant design methods, each with advantages and disadvantages. Radiation-immune storage elements often have additional components, such as resistors or transistors, which reduce density and, in some designs, increase power dissipation or limit switching speed. Radiation-immune processes such as silicon on insulator (SOI) are more expensive [15]. In contrast, classical fault-tolerant design techniques can be implemented in standard technologies such

[†] This research was supported in part by NASA under Space Engineering Research Center Grant NAGW-3293.

as low power CMOS, using standard logic gates without any special modifications. However, such designs always employ some form of redundancy, resulting in larger circuits.

The advances made in VLSI design and technology, namely reduced feature size, pass transistor circuits, and design automation tools, warrant a new analysis of the tradeoffs associated with the different techniques for achieving fault-tolerance for the problem discussed. Consequently, in this paper we will focus on various classical design methods to create fault tolerant state machines; specifically, methods for correcting single errors in the storage devices (flip-flops), as might occur in a space environment. Comparisons will be made in terms of the complexity of the design technique, resultant circuit size, and circuit speed.

Definitions

The following are definitions which will be used throughout this discussion.

State Machine: A circuit for which the current outputs depend upon past inputs as well as current inputs, described in terms of a set of states.

State Table: Describes the operation of the state machine in a symbolic fashion.

State Encoding Table: Gives the binary codewords representing the symbolic states of the system after they are encoded by a set of state variables. The minimum number of state variables needed for encoding is $log_2(N)$, where N is the total number of states in the machine.

Next State Map: A Karnaugh map (k-map) which describes the states the system enters depending on the present state and the input. The next state map is used in designing the excitation circuit of the state machine.

Fault Tolerant System: A system that continues to correctly perform its specified tasks in the presence of failures [13].

Hamming Distance: The minimum number of bit positions in which any two codewords differ [14].

Generator Matrix: A k by n matrix with the property that any codeword is a linear combination of the matrix rows. The generator matrix is a concise way to describe a linear code. It consists of two parts, a k by k identity matrix and k by $(n-k)$ parity matrix [14].

2 Fault Tolerant State Machine Design

Fault tolerant designs require the use of some form of redundancy. Redundancy can be in terms of hardware, information, or time [13]. A well known example of hardware redundancy is triple modular redundancy. This uses three copies of the original circuit and a voter to perform error masking at over a three-fold increase in the circuit size. It has been shown in the literature that single fault tolerant state machines can be designed using only two copies

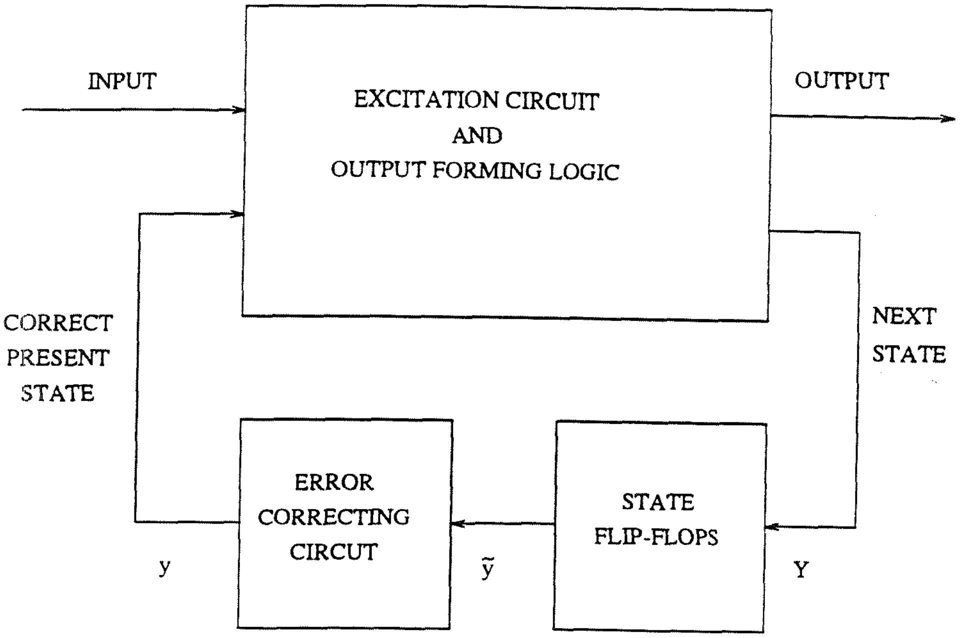


Figure 1: Fault Tolerant State Machine using Explicit Error Correction

of the nonredundant machine and some additional hardware [5, 6]. The approach used in these methods is to detect and suppress the output of the faulty copy. A time redundant system performs operations more than once in order to detect and possibly correct the errors. Because of the penalties associated with these methods in terms of size and operational speed, neither are ideal for implementing high performance, fault tolerant state machines.

The use of information redundancy to achieve fault tolerance on the other hand is very attractive, especially considering the design procedure [12] for synchronous state machines. Hence, in this paper we will focus on the various techniques which use information redundancy in the design of single fault tolerant state machines. Each of these methods involves adding additional *parity bits* to the existing set of state variables which are used to encode the states of the sequential machine. In order to have single bit error correcting capability, the Hamming distance between the encoded states should be greater than or equal to three [14]. Different design methods which use this fundamental principle of coding theory are presented in the next few sections and will be explained using the state machine shown in Table 1 as an example.

2.1 Explicit Error Correction

This method was first proposed by I.S. Reed and A.C. Chiang and is illustrated in Figure 1 [3]. It is a direct application of the Hamming distance three codes. Hamming linear block codes, also known as (n, k) codes, are used to encode the states and provide error correction capability. The Hamming code uses $(n-k)$ parity bits to protect the original k state variables.

Three state variables are required to encode the six states of the benchmark state machine. The encoding of states is based on design principles in [12] and the encoded states are as shown in Table 2. Three parity bits have to be appended to the state variables to maintain

State	I_1	I_2
A	C, 0	B, 0
B	D, 1	E, 0
C	E, 0	F, 1
D	A, 1	A, 0
E	A, 0	A, 1
F	A, 0	A, 0

Table 1: State Machine Description

State	y_1	y_2	y_3
A	0	0	0
B	0	1	0
C	0	1	1
D	1	1	1
E	1	1	0
F	1	0	0

Table 2: State Encoding Table

State	y_1	y_2	y_3	y_4	y_5	y_6
A	0	0	0	0	0	0
B	0	1	0	1	0	1
C	0	1	1	1	1	0
D	1	1	1	0	0	0
E	1	1	0	0	1	1
F	1	0	0	1	1	0

Table 3: State Encoding using Hamming Codes

the minimum distance requirement needed to design a single fault tolerant state machine. If y_1, y_2, and y_3 are the original state variables, then the parity bits are given as

$$\begin{aligned} p_1 &= y_1 \oplus y_2, \\ p_2 &= y_1 \oplus y_3, \text{ and} \\ p_3 &= y_2 \oplus y_3. \end{aligned}$$

The resultant state encoding table is shown in Table 3.

The excitation circuit is a combinational logic block which generates the next state based on the present state and the inputs. It is designed using the next state maps under each input [12]. Part of the symbolic next state map for the example under consideration is shown in Table 4. The entry C^A means that for present state A and under input I_1, the system enters the next state C; the d entries represent don't cares. A similar next state map is required to design the excitation circuit for input I_2. The excitation circuit which generates the next state is designed from the next state maps similar to one shown in Table 4 and replacing the symbolic entries by state variables. The logic equation for state variable Y_1 is

$$Y_1 = y_1' y_2 I_1 + y_1' y_2 I_2.$$

		y_3y_4			
		00	01	11	10
	00	C^A	d	d	d
	01	d	d	d	d
y_5y_6	11	d	d	d	d
	10	d	d	d	d

Table 4: K-map for $y_1y_2 = 00$ under input I_1

				y_3y_4	
		00	01	11	10
	00	C^{A}	C^{A4}	d	C^{A3}
	01	C^{A6}	D^{B2}	d	d
y_5y_6	11	d	d	d	d
	10	C^{A5}	A^{F1}	E^{C2}	d

Table 5: K-map for $y_1y_2 = 00$ under input I_1

Equations for the remaining next state variables may be similarly obtained.

The explicit error correction circuit shown in Figure 1 is designed based on the syndrome generation principle. The parity bits are calculated from the output of the state flip flops and compared with the original parity bits to form the syndrome. If the syndrome is not equal to zero then the state stored in the flip flops is incorrect. The syndrome indicates which bit needs to be inverted to correct the error.

2.2 Implicit Error Correction

This method of single fault tolerant state machine design is also based on Hamming distance three codes. The excitation circuit which generates the next state in the sequential machine is designed such that even if one of the memory elements (flip flops) is in error, a correct next state is generated. In other words, the error correction circuit is built into the excitation circuit of the sequential machine. This method was proposed by R.L. Russo in the early 1960's [1] for designing fault tolerant counters. Analysis of this method was carried out by J.F. Meyer in [4].

The encoding of state variables is done as explained previously and is shown in Table 3. This method differs from the previous one in the design of the excitation circuit. Here the present state and all its adjacent states are forced to proper next state. This is accomplished by making appropriate state entries in the next state map. For example, the entry C^A in the Table 5 means under present state A and input I_1, the system enters the next state C. The entry C^{Az} means that when bit z of the state encoding of A is in error the system goes to proper next state, C. In this example z has value between one and six. In the previous method, the next state map had a large number of don't care conditions and hence the excitation circuit could be designed efficiently by exploiting this. In this method, the number of don't care states are reduced and the excitation circuit becomes complex. For example, the next state equation obtained using this design method to generate variable Y_1 illustrates the fact. The equations for variables Y_2 through Y_6 are similarly designed.

$$Y_1 = y_1'y_4y_6 + y_1'y_2'y_3y_5 + y_1'y_2(y_6y_5' + y_4 + y_5y_3) + y_1y_2(y_4y_5'y_6 + y_3y_4y_5)(I_1 + I_2)$$

2.3 Permutation Codes

It is desirable in any fault tolerant design to minimize the redundancy without affecting the fault tolerant capabilities of the system. The number of parity bits that has to be added when

State	y_1	y_2	y_3	y_4	y_5
A	0	0	0	0	0
B	1	1	0	1	0
C	1	1	0	0	1
D	0	1	1	1	0
E	1	0	1	1	1
F	0	1	1	0	1

Table 6: State Encoding using Permutation Codes

encoding the states of a single fault tolerant state machine can be reduced by using permutation codes. This method suggested by E.A. Tractenberg in [8], is based on the assumption that under no input is it possible to reach all states of the state machine. Consequently, to achieve single fault tolerance, a minimum distance of three has to be maintained only between the states that are reached under the same input. For input X_i, a code C_i is used to encode the states of machine, where C_i stands for a linear block code (n, k). For another input X_j, a coset [14] of C_i is used to encode the states. The cosets of C_i are formed by permuting the columns of the generating matrix **G** which forms the code C_i. This method of encoding the states reduces the redundancy in both the number of flip flops used and the combinational logic which generates the next state equations.

In the example under consideration, there are four distinct states under each input. To encode them we need two bits and to maintain a distance of three between the states three parity bits must be added [14]. Thus, a (5, 2) linear code is needed to encode the states. The first step in the encoding process is to form the generator matrix [14]. The generator matrix for a (5, 2) code is shown in the Equation 1.

$$\mathbf{G} = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{1}$$

The fault tolerant state encoding is formed by multiplying the original state encoding by a generator matrix, **G**. Now, by interchanging the columns of the of the generator matrix we get a permutation code. For the given generator matrix shown in Equation 1 there are 5! permutations. Not all permutations result in new codewords. For example, by interchanging columns one and three of the generator matrix shown in Equation 1 we get a new generator matrix, G_{13}. Using the same original state encodings and the new generator matrix, a different set of code words can be generated.

In the example under consideration there are six states and we need six distinct codewords. This is satisfied by just one permutation of the columns of the generator matrix. Depending on the problem need might arise to generate codes using more number of permutations or even add few more parity bits. The six distinct codewords represented by the decimal equivalent values are 0, 13, 14, 23, 25 and 26. The problem now is to assign the codewords to the states of the machine such that any two states under a particular input are separated by a minimum distance of three. This is based on the algorithm shown in [8]. The state assignments obtained are shown in Table 6.

The excitation circuit for the state machine is designed as shown in Section 2.1. The error correction is based on verifying that a state reached under input I_x belongs to the set

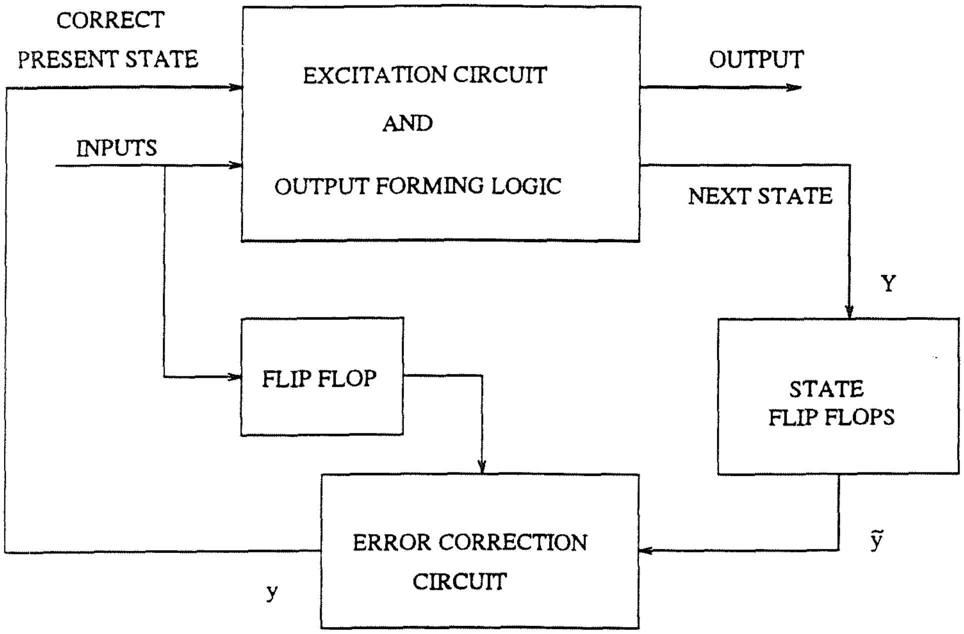


Figure 2: Fault Tolerant State Machine using Permutation Code

of states reachable under that input. Based on this principle, under input I_1 the system can go only to states A, C, D and E. If the system reaches any other state, it is forced to the proper state under input I_1 by a combinational circuit similar to that shown in Section 2.2. The conceptual block diagram for such a fault tolerant sequential machine is shown in Figure 2.

2.4 Two-Hot Codes

The design methods suggested in the earlier sections are efficient in terms of the number of redundant state variables used. However, the design of the excitation circuit is complicated and tedious. The use of one-hot code for encoding the state variables allows a state machine to be designed by inspection [12]. The number of state variables required to encode the states is equal to the number of states in the sequential machine. The use of one-hot code for state assignment results in a Hamming distance of two between any pair of states in the machine. This does not satisfy the Hamming distance requirement needed to *correct* single bit error.

Two-hot code can be used for encoding the states of the sequential machine to make it single fault tolerant. Here, each state is represented by two state variables which are active. The procedure to design the single fault tolerant state machine using this method is illustrated using the example shown in Table 2. The six states of the machine are encoded using twelve state variables. The state encoding obtained using this method of encoding is shown in the Table 7.

The excitation circuit of the state machine is designed by inspection as in the one-hot

State	y_{12}	y_{11}	y_{10}	y_9	y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1
A	1	1	0	0	0	0	0	0	0	0	0	0
B	0	0	1	1	0	0	0	0	0	0	0	0
C	0	0	0	0	1	1	0	0	0	0	0	0
D	0	0	0	0	0	0	1	1	0	0	0	0
E	0	0	0	0	0	0	0	0	1	1	0	0
F	0	0	0	0	0	0	0	0	0	0	1	1

Table 7: State Encoding using Two-Hot Code

code method. The logic equation for the state variable Y_{12} is given as

$$Y_{12} = (y_6 + y_4 + y_2)I_1 + (y_6 + y_4 + y_2)I_2.$$

The block diagram of a fault tolerant state machine using this method is similar to that shown in Figure 1. An explicit error correction circuit is used after the state flip flops. The logic equation of the error correction circuit for state variable Y_{12} is

$$y_{12} = \tilde{y}_{12}\tilde{y}_{11} + (\tilde{y}_{12} \oplus \tilde{y}_{11})\tilde{y}'_{10}\tilde{y}'_9\tilde{y}'_8\tilde{y}'_7\tilde{y}'_6\tilde{y}'_5\tilde{y}'_4\tilde{y}'_3\tilde{y}'_2\tilde{y}'_1.$$

3 Simulation Results

The benchmark state machine was simulated and synthesized using UC-Berkeley OCT-TOOLS for the different methods discussed. Layouts using standard cells were generated and the maximum delay through the next state and error correction logic was found. These results are shown in Table 8.

Method 1 using explicit error correction circuit is the most straight forward of designing a single fault tolerant synchronous state machine. A fault tolerant counter using a modified Reed-Muller code and Hamming code was designed in [3] based on similar design principles. The error correction circuitry in [3] was designed using majority gates. The use of CAD tools for synthesis of larger state machines which use Method 2 becomes a necessity. J. Beister in [2] had suggested the use of threshold logic elements in the design of excitation circuits which is capable of correcting single errors. However this does not reduce the complexity of the design procedure. The permutation code method is a combination of explicit and implicit error correction. The size of the excitation and error correction circuits is reduced in this method. The limitation is the complexity of the state encoding algorithm. Also the error correction circuit requires delayed primary inputs, resulting in the need for flip flops. This effectively offsets the advantages gained with reduced number of parity bits. The two-hot code method is very simple and can be designed by inspection. The price is the increased number of flip flops in the state machine. This was applied to the design of fault tolerant asynchronous sequential machines in [9] and is more suitable for implementation in pass transistor technology than in standard cells. Efforts are currently underway to replicate the results for Method 4 using pass transistors.

	Nonredundant	Method 1	Method 2	Method 3	Method 4
Dimensions (λ)	472 x 228	536 x 904	616 x 836	672 x 820	816 x 804
Area (λ^2)	107616	484544	514976	551040	656064
Max Delay (ns)	34.0	109.0	95.7	106.3	82.0

Table 8: Results of Simulation

4 Conclusions

Research these days is focussed on concurrent error detection in sequential state machines as in [10], [11] and [7]. These design principles are not suitable for achieving on-line fault tolerance which is required for hardening state machines against SEU's. But other techniques used to design on-line fault tolerant sequential machines and advantages of using information redundancy methods were discussed. Various methods which use information redundancy to design fault tolerant state machines along with the strength and weakness of each method were presented. The results of the simulation for a simple six state and single output state machine show that explicit error correction method is most efficient in terms of area. MCNC logic synthesis benchmark state machines are being synthesized to further verify the results.

References

[1] R.L. Russo, "Synthesis of error tolerant counters using minimum distance three state assignments," *IEEE Trans. on Computers,* vol. EC-14, pp. 359, June 1965.

[2] Jochen Beister, "Implementation of failure tolerant counters," *IEEE Trans. on Computers,* vol. C-17, pp. 885–886, September 1968.

[3] I.S. Reed, "Coding techniques for failure tolerant counters," *IEEE Trans. on Computers,* vol. C-19, pp. 1035–1038, November 1970.

[4] J.F. Meyer, "Fault tolerant sequential machines," *IEEE Trans. on Computers,* vol. C-20, pp. 1167–1177, October 1971.

[5] Gary Maki, "Reliable sequential circuit design," *Circuits and Systems,* pp. 6–10, June 1975.

[6] A.S. Gupta, "Realization of fault tolerant and fail safe sequential machines," *IEEE Trans. on Computers,* vol. C-26, pp. 91–96, January 1977.

[7] M. Alahmad and S. Whitaker, "Fault tolerant sequential circuits using sequence invariant state machines," in *3rd NASA SERC Symposium on VLSI Design,* pp. 2.5.1–2.5.12, October 1991.

[8] M. Chen and E.A. Tractenberg, "Permutation codes for state assignment of fault tolerant sequential machines," in *Proc. IEEE /AIAA 10th Digital Avionics system conference,* pp. 85–90, 1991.

[9] J. Teeter, “N-Fault tolerant sequential circuits,” *Master’s thesis,* Univ. of Idaho, 1975.

[10] R. Leveugle, “Optimized synthesis of concurrently checked controllers,” *IEEE Trans. on Computers,* vol. C-39, pp. 419–425, April 1990.

[11] L.P. Holmquist and L. L. Kinney, “Concurrent error detection for restricted fault sets in sequential circuits and microprogrammed control units using convolution codes,” in *1991 International Test Conference Proceedings,* pp. 926–935, October 1991.

[12] David J. Comer, *Digital logic and state machine design.* Saunders College Publishing, 1990.

[13] B. Johnson, *Design and analysis of fault tolerant digital systems.* Addison-Wesley Publishing Company, 1989.

[14] R.E. Blahut, *Theory and practice of error control codes.* Addison-Wesley Publishing Company, 1984.

[15] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design.* Addison-Wesley Publishing Company, 1988.

Low-Latency Digital Frequency Synthesizer Using The Residue Number System

William A. Chren, Jr. School of Engineering
Grand Valley State University
Grand Rapids MI 49504
chrenb@gvsu.edu

Abstract - **A low-latency frequency synthesizer using the Direct Digital Synthesis (DDS) technique has been designed. Called the Residue Assisted Frequency Synthesizer (RAFS), it exhibits frequency switching times which are reduced by more than 50% below previously published designs. The switching speed advantage is made possible by the use of the Residue Number System, which allows the pipeline lengths in the Phase Accumulator and other circuitry to be reduced significantly.**

1 Introduction

In the last several years Direct Digital Synthesis (DDS) techniques have become an increasingly popular method of frequency synthesis. This has happened in response to the increasingly stringent performance specifications required of frequency synthesizers for many different applications such as spread spectrum and frequency hopping communication systems, Doppler and chirp radar systems, EW and jammer systems, magnetic resonance spectroscopy/imaging, and test equipment design. Direct Digital Frequency Synthesizers (DDFSs) offer many important advantages over Phase-Locked Loop (PLL)-based synthesizers, including ultra-small frequency resolution, fast and phase-continuous frequency switching, exceptional linearity and excellent temperature and aging stability [1].

The Residue Number System (RNS), invented in the late 1950s, offers the pleasing prospect of "one-step" addition, subtraction and multiplication. It allows carry-free and partial-product-free operations which are performed digit-on-digit, in parallel. It has acquired significant interest lately for use in high-performance applications such as signal processing and image processing.

This paper is a presentation of an RNS-based Digital Frequency Synthesizer called the Residue Assisted Frequency Synthesizer (RAFS). It is shown that the RAFS has reduced frequency switching time by more than 50% when compared with traditional binary arithmetic-based designs.

The organization of this paper is as follows. Section 2 is a discussion of the architecture of traditional DDS systems as well as the fundamentals of the Residue Number System. Section 3 presents the RAFS and Section 4 contains a model and an analytical estimate of its performance. Section 5 contains some conclusions.

2 Background

This section is a presentation of Direct Digital Frequency Synthesis and several figures of merit for DDFS designs. It also contains a brief introduction to the Residue Number System.

2.1 Direct Digital Frequency Synthesis

All direct digital frequency synthesizers use essentially the same method, called the Sine Table Lookup Method. It was first published by Tierney, Rader and Gold in 1971 [2]. The idea is to generate a sinusoidal output by periodically accessing ROM-stored digital samples of a sine wave, and converting the samples by means of a Digital-to-Analog converter (DAC). The ROM addresses are computed using a digital Phase Accumulator. The basic structure of traditional Digital Frequency Synthesizers is shown in Figure 1. The Phase Accumulator consists of the adder/latch feedback subsystem.

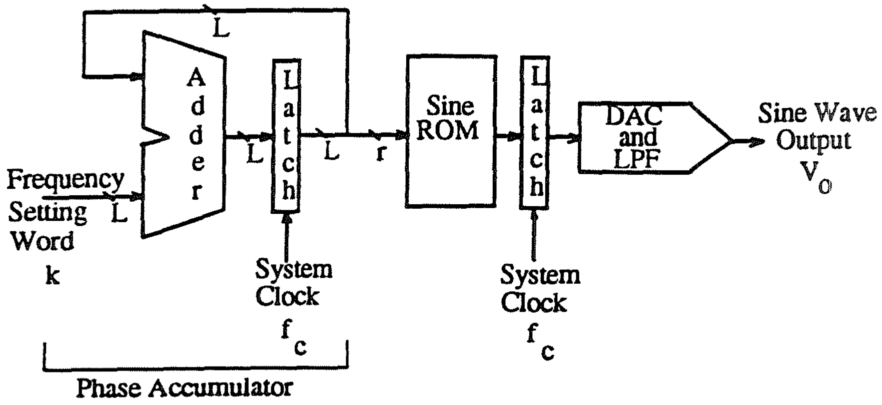


Figure 1: Traditional Digital Frequency Synthesizer Architecture

It computes L-bit multiples (the "phase sequence") of the frequency setting word k by successive addition modulo 2^L. The sum is latched and the r most significant bits of it are used as an address to enter the ROM. The accessed sample is then latched for input to the DAC. It can be shown that the output of the DAC is the signal $V_0(lT) = \sin(2\pi \frac{f}{f_c} l) = \sin(2\pi f l T)$, $1 = 0, 1, 2, ...$ where $f_c = 1/T$ is the system clock frequency and l is the sampling index. This signal is of frequency f, which is related to f_c, the Phase Accumulator width 2^L and the L-bit frequency setting word k by the equation $f = \frac{f_c}{2^L} k$. Clearly, the frequency f of the output sinusoid can be controlled by the value of k.

2.1.1 DFS Figures of Merit

The most important DFS figure of merit for our discussion is the frequency switching time. The frequency switching time (sometimes called the "latency") of the synthesizer is the amount of time required to switch the output from one frequency to another. This is a particularly important figure of merit for frequency hopping and digital FM communication systems because it determines the maximum hopping rate and modulation bandwidth, respectively.

The Phase Accumulator addition time is the largest of the processing delays in a DFS and is the largest contributor to the latency. This is because of the carry propagation time, which is relatively large for accumulators of practical size (as large as 48 bits). Most synthesizers use pipelining to increase throughput. The level of pipelining is dependent on the relative sizes of ROM access time and the speed of the logic and fabrication process used. Typical values of the number of accumulator bits per pipeline stage can be lower than one bit per stage [3]. The latency of such a pipelined accumulator is the number of stages multiplied by the clock period. Typical values can be as low as 34 nsec (24 bits at 700MHz) [4].

2.2 The Residue Number System

The Residue Number System (RNS) is a number system in which an integer is represented as the vector of its residues modulo a fixed, specially chosen set of integers called moduli. In the RNS the operations of addition, subtraction and multiplication can be performed without carries, borrows, or partial-products. These operations are done "digit-on-digit" concurrently, in a single-step computation, using small processors in each digit position. The operation of division, however, cannot be done nearly so simply because of the necessity of rounding, which is expensive in the RNS [5].

Mathematically, letting m_i denote the ith modulus, the residue representation of the integer X is $X \Longleftrightarrow (x_1, x_2, \ldots, x_n)$, where $X_i = |X|_{m_i} \doteq X$ modulo m_i. If the moduli are chosen to be pairwise relatively prime, then the residue representation of X is unique for $0 \leq X \leq M - 1$, where M is the product of the moduli and is called the range of the representation.

If $X \Longleftrightarrow (x_1, x_2, \ldots, x_n), Y \Longleftrightarrow (y_1, y_2, \ldots, y_n)$ and $Z \equiv X * Y$ where $*$ represents any of the three operations of addition, subtraction or multiplication, then $Z \Longleftrightarrow (|x_1 * y_1|_{m_1}, |x_2 * y_2|_{m_2}, \ldots |x_n * y_n|_{m_n})$. These operations are very fast because they are done in digit-parallel fashion, modulo small integers m_i.

A special residue operation called "scaling" is the equivalent of "right-shifting" (radix division) in the binary number system. It computes the truncated quotient of integer X with modulus m_i, denoted $\lfloor x/m_i \rfloor$. This quantity is found by subtracting the ith digit from all others and then multiplying the result by the multiplicative inverse of m_i , denoted m_i^{-1} . The subtraction and multiplication are done modulo m_j for $j \neq i$. Formally, this is $\lfloor x/m_i \rfloor \Longleftrightarrow (|m_i^{-1}(x_1 - x_i)|_{m_1}, |m_i^{-1}(x_2 - x_i)|_{m_2}, \ldots |m_i^{-1}(x_n - x_i)|_{m_n}), j \neq i$. The result has no ith digit because the operation is not defined on the modulus i. Truncation by a product of moduli can be done by sequential scalings, one for each modulus in the product. One digit is eliminated for each scaling operation, and this fact will be used shortly to reduce Sine ROM size in the RAFS. More RNS information can be found in [6].

3 Design and Implementation of the RAFS

3.1 Basic Architecture

A block diagram of the Residue Assisted Frequency Synthesizer is shown in Figure 2 and will now be discussed.

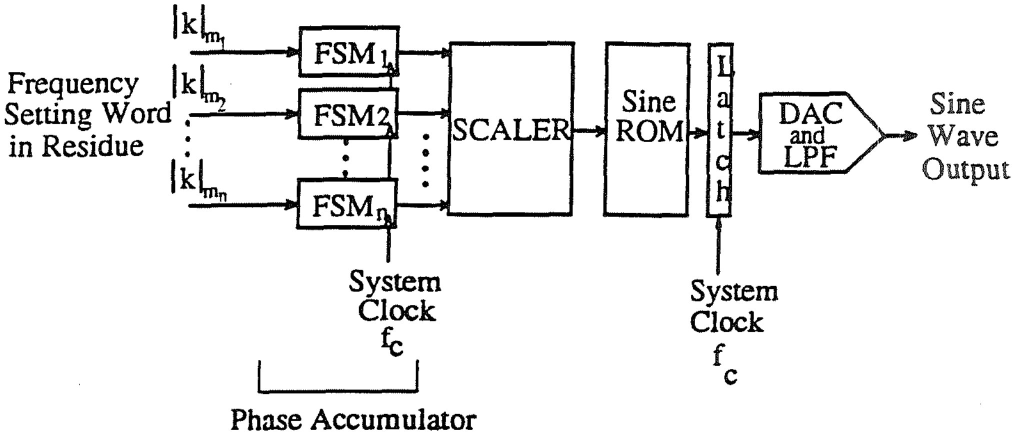


Figure 2: Residue Assisted Frequency Synthesizer

3.1.1 The Phase Accumulator

The Phase Accumulator is a residue arithmetic version of its binary counterpart. In Figure 2 its modular structure is evident. It consists of n independent finite state machines (FSMs), one per residue digit, clocked in unison. The ith FSM performs accumulation modulo m_i. The input to FSM$_i$ is the binary encoded ith residue digit $|k|_{m_i}$ of the frequency setting word. The state of FSM_i at any time is the binary encoded ith residue digit of the (accumulated) phase at that time. The validity of using FSMs instead of modulo m_i adders is clear from the fact that, modulo m_i, the value of $|k|_{m_i}$ and the present value of the ith residue digit of the phase determine uniquely all subsequent values of the ith digit of phase. The FSMs are used instead of modulo m_i adders because they are usually faster, since they can be implemented using only two logic levels of delay with PLAs. Section 4 contains an analytic expression for the delay using PLAs as well as modulo m_i adders.

The Phase Accumulator performs addition modulo the product of the moduli, that is, modulo $M = \prod_{i=1}^{n} m_i$. It therefore has an effective bit length of $\log_2 M = \sum_{i=1}^{n} \log_2 m_i$, that is, approximately equal to the sum of the bit lengths of the moduli.

The choice of the number and size of the moduli is made with regard to latency and frequency resolution, defined as the smallest nonzero synthesizeable frequency. As will be made clear in Section 4, if residue adders (rather than state machines) are used to implement the Phase Accumulator, then latency will increase with the size of the largest modulus. Increasing the number of moduli also increases the latency assuming that the number of residue digits used to address the Sine ROM is constant. Frequency resolution increases with increasing M, the product of the moduli.

3.1.2 The Scaler

The Scaler computes the residue scaling operation on the output of the Phase Accumulator which, as discussed above, consists of the residue encoded value of (accumulated) phase.

The Scaler is needed to eliminate some of the residue digits so as to decrease the size of the Sine ROM, and in effect performs the residue equivalent of truncation rounding. The architecture of the scaler is shown in Figure 3. In the figure, the input data consists of the n Phase Accumulator output digits $|lk|_{m_i}$, where the variable l is the integral "time index" mentioned in Section 2.1. The Scaler is designed to reduce the number of residue digits to a value $r < n$. This value is chosen so that the number of binary bits used to represent all r digits is equal to the number of address inputs to the Sine ROM. The removal of the $n-r$ residue digits (chosen without loss of generality to be digits $r+1$ through n) is done by scaling by the product of the moduli $m_{r+1}, m_{r+2}, \ldots, m_n$ as explained in Section 2.2. That is, the output of the Scaler is the residue encoded value R, where $R = \lfloor ik/m_{r+1}, m_{r+2}, \ldots m_n \rfloor$, encoded using the moduli $m_1, m_2, \ldots m_r$.

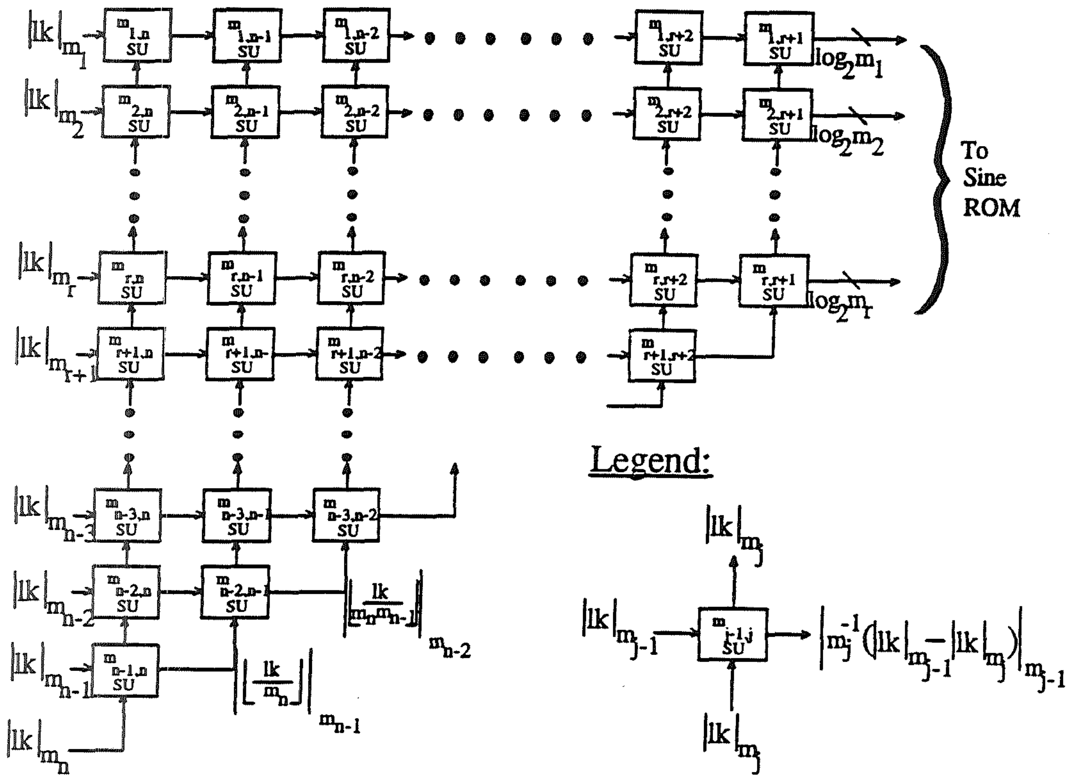


Figure 3: Scaler Architecture

As the Legend in Figure 3 shows, the $m_{j-1,j}$ scaling unit (SU) computes the modulo $j-1$ value of truncated lk. That is, the output of the $m_{j-1,j}$ SU is $|\lfloor \frac{lk}{m_j} \rfloor|_{m_{j-1}}$. This computation involves a "premultiplied" (by m_j^{-1}) subtraction modulo m_{j-1} of the lower operand from the left operand. It is clear from the figure that the scaling units are arrayed so as to implement a direct "flowthrough" computation of equation (8).

The SUs can be implemented in several ways. The obvious way is to build them as modulo m_{j-1} adders (with postmultiplying by m_j^{-1} "built in"). Another possibility is to use PLAs or ROM look-up tables. More will be said about these implementations in Section 4.

Note that there is one datapath level of SUs for every removed residue digit. This means that in the design of the RAFS, latency and Sine ROM size can be traded off against each other. That is, assuming that the n moduli used in the Phase Accumulator cannot be changed (i.e., its effective bit length is fixed), then the removal of a residue digit necessitates a larger latency, but decreases the size of the required ROM. Conversely, removing fewer digits produces smaller latency but increases the ROM size. The fact that the designer has a great deal of flexibility in selecting the "tradeoff point" is an attractive feature of the RAFS architecture. A corresponding advantage for traditional binary-based architectures does not exist. In the latter case, once the size of the Phase Accumulator is fixed, the latency is also and is independent of ROM size.

3.1.3 Sine ROM and DAC/LPF

The Sine ROM and DAC/LPF (without the latch) are shown in Figure 4. From the figure, it can be seen that the r residue digits generated by the Scaler are used as address bits. The total number of address bits is given by the expression $\sum_{i=1}^{r} \lceil \log_2 m_i \rceil$. The ROM stores p-bit periodic samples of one complete period of a sinusoid. The number of stored samples is equal to the product of the moduli $m_1, m_2, \ldots, m_r$. Note that this value will be less than the size of the memory (except in the case of a single power-of-two modulus, which is the traditional architecture!) so therefore a small number of memory locations will not be used. The fraction of unused memory locations is small for modulus sets consisting of moduli which are large and approximately equal to a power of 2.

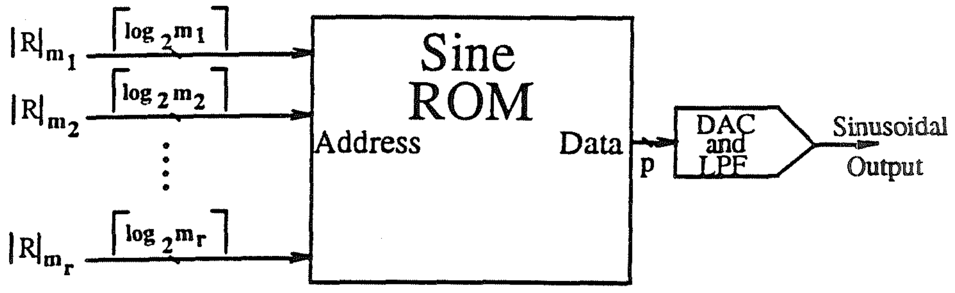


Figure 4: Sine ROM and DAC/LPF

4 Latency Estimate of the RAFS

This section is a derivation of analytical estimates of the latency of both the traditional architecture and the RAFS. It will be shown that the RAFS implemented with PLAs exhibits less than half the latency of the traditional design. The latency estimate of the RAFS will be found under the approximation that the n moduli in the residue representation are of equal size m. This assumption is justified because most practical residue-based signal processing systems use relatively few, large moduli. This section also includes a reduced area design which has at least 20% less latency than the traditional architecture.

It will be assumed that the Sine ROM sizes and DAC/LPF implementations are identical in both the RAFS and traditional architectures, and therefore the designs differ only in their "front ends", consisting of the Phase Accumulator and the Scaler (in the RAFS). This assumption is reasonable because it gives both architectures the same values of phase and amplitude resolution. Furthermore, since for the highest speed applications the pipeline stage delay time (i.e., the system clock period) is determined by the combination of ROM access time and DAC settling time, this assumption forces these delay times to be equal for both architectures. Latency estimates can therefore be made using numbers of pipeline stages. Furthermore, only the number of stages in the datapath up to the address port of the Sine ROM will be counted. This is done for simplicity since the ROM access time and DAC settling time are the same for both architectures.

The strategy of performance evaluation is to find the length (measured in gates) of the critical path in both architectures. Interconnection delays will be ignored for simplicity. Because modulo m adders and subtracters will occur frequently, it is advantageous to introduce a "level of pipelining" parameter $d, d > 0$, which is defined as the number of pipeline stages per bit of ripple carry addition or subtraction. For example, for a 6-bit ripple carry adder with three stages, $d = .5$ stages per bit. The parameter d is a measure of the gate speed of the process used to fabricate the architectures because as discussed above the clock speed is the same for both designs. In high speed applications where greater than bit-level pipelining is used [7], $d > 1$.

It will further be assumed that the bit lengths of both Phase Accumulators is the same. That is, the bit length of the traditional architecture is the same as that of the RAFS, namely $\log_2 m^n = n \log_2 m$.

4.1 Latency Estimate of the Traditional Architecture

An estimate of the latency of the traditional architecture can be derived by observing in Figure 1 that the delay from input k to the ROM address port is all associated with the Phase Accumulator adder. It will be assumed that its architecture is ripple carry. Since it is $n \log_2 m$-bits wide, we conclude that its estimated pipeline stage delay is $D_{trad} = dn \log_2 m$.

4.2 Latency Estimate of the RAFS Using PLAs

In this section a latency estimate for a PLA-based version of the RAFS will be derived. PLAs are used to implement the FSMs in the Phase Accumulator as well as several scaling units.

The input to the modulo m_i FSM in the Phase Accumulator (see Figure 2) is $|k|_{m_1}$ (the ith residue digit of the frequency setting word k). The output is the ith digit of the present value of the accumulation and is updated every clock transition. A PLA can be used to implement the state feedback combinational logic of the FSM. It will have a total of $2\lceil \log_2 m_i \rceil$ bits of input (half for the present state and half for $|k|_{m_i}$) and $\lceil \log_2 m_i \rceil$ bits of output. The critical path in the PLA is two gates long. In order to relate gate delays to d, we can employ a commonly used architecture of a full adder [8] which has four gate delays. We conclude that there are $.25d$ pipeline stages per gate in a ripple carry architecture. We therefore conclude that the PLA-based Phase Accumulator has a latency of $.5d$ pipeline stages. The PLA-based version of the Scaler is shown in Figure 5 for the case $n = 6$ and $r = 2$. It can be seen that

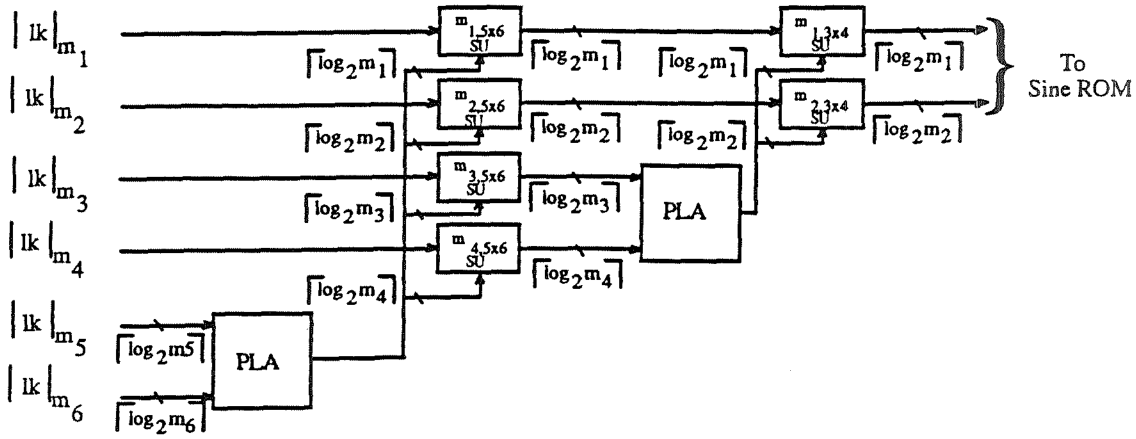


Figure 5: PLA-based Implementation of Scaler

the PLAs are used to convert two residue digits $|lk|_{m_i}, |lk|_{m_j}$ into the residue digits for the additive inverse of the quantity $|lk|_{m_i m_j}$, encoded using the remaining moduli. For example, the lower PLA in Figure 5 outputs the quantity $|-|lk|_{m_i m_j}|_{m_0}$, where m_0 is any of the moduli m_1, m_2, m_3 or m_4. This allows the scaling operation to be performed with one fewer column of scaling units (see Figure 3) for each PLA used. The delay of the Scaler can be found as follows. A scaling unit performs an addition (modulo m) of operands which are premultiplied by m^{-1}. The premultiplication can be considered a "renaming" of both operands and can therefore be done with the same number of pipeline stages as used by a "non-renaming" ("regular") modulo m adder. We will use a model for a modulo m adder that is due to Shanbhag [9], shown in Figure 6. It is readily seen that it consists of a cascade connection of an l-bit binary ripple carry adder, a ripple borrow subtracter and a 2:1 multiplexer.

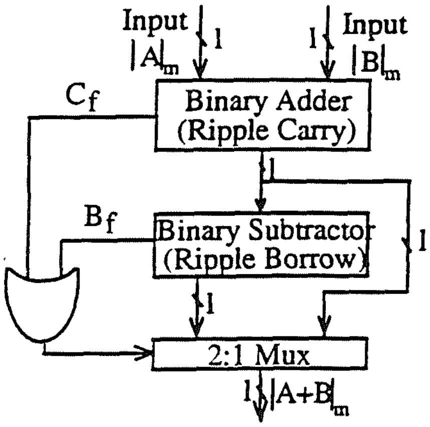

Input
|A|m
l
l
Input
|B|m
Cf
Binary Adder (Ripple Carry)
l
Bf
Binary Subtractor (Ripple Borrow)
l
l
2:1 Mux
l
|A+B|m

Figure 6: Model of a Modulo m Adder

The value of l is $l = \lceil \log_2 m \rceil$. The processing in the subtracter is overlapped with that in the adder [10], so that the delay of the cascade connection is one bit larger than in the adder alone. Therefore, the number of pipeline stages incurred up to the Mux inputs is $d + d\lceil \log_2 m \rceil$. The delay of the multiplexer is one gate ($.25d$) if we use a transmission gate-based design [11], and therefore an estimate of the number of pipeline stages needed by the adder architecture in Figure 6 is $d(1.5 + \lceil \log_2 m \rceil)$. For every two moduli that are used for scaling, the delay is one PLA plus that of the adder architecture in Figure 6, or $d(2 + \lceil \log_2 m \rceil)$. Since the Scaler must scale by the $n - r$ moduli $m_{r+1}, m_{r+2}, \ldots, m_n$, the overall delay of the Scaler therefore consists of $d(\frac{n-2}{2})(2 + \lceil log_2 m \rceil)$ stages (if $n - r$ is even), or $d\lfloor \frac{n-r}{2} \rfloor (2 + \lceil \log_2 m \rceil) + d(1.5 + \lceil \log_2 m \rceil)$ if $n - r$ is odd. Adding this result to the $.5d$ delay of the Phase Accumulator gives a total delay for the PLA -based RAFS of $D_{PLA} = d\lfloor \frac{n-r}{2} \rfloor (2 + \lceil log_2 m \rceil) + d(1.5 + \lceil \log_2 m \rceil)|n - r|_2 + .5d$. The PLA-based RAFS improves the latency of the traditional design by the fraction

$$\frac{D_{trad} - D_{PLA}}{D_{trad}} = 1 - \frac{\lfloor \frac{n-r}{2} \rfloor (2 + \lceil log_2 m \rceil) + (1.5 + \lceil log_2 m \rceil)|n - r|_2 + .5}{n \log_2 m} \tag{1}$$

For the case $n = 5$, $r = 3$, $m = 32$, this is a latency improvement of 70%. The price of this significant performance improvement is a larger chip area due to the PLAs. ROMs are not a small area alternative because in most implementations they require more silicon area than PLAs. However, residue adders are a viable alternative. It can be shown that when the residue adders above are used in place of the FSMs in the Phase Accumulator and in place of the PLAs in the Scaler, the RAFS improves the traditional latency by the fraction

$$\frac{D_{trad} - D_{ra}}{D_{trad}} = 1 - \frac{(1 + n - r)(1.5 + \lceil log_2 m \rceil)}{n \log_2 m} \tag{2}$$

For example, for $n = 5$, $m = 32$ and $r = 3$ the decrease is 22%.

The expression is a lower bound on the latency improvement for the chosen moduli and is also the lowest area implementation. Equation 1 has much lower latency but requires several area expensive PLAs. An attractive compromise is to use a single PLA in the Scaler, for which the latency should substantially improve above Equation 2.

5 Conclusions

We have shown that the Residue Assisted Frequency Synthesizer (RAFS) has at least a 50% reduced latency below traditional binary number system-based direct digital frequency synthesizers. We also conclude that the RAFS exhibits a significant amount of flexibility in its latency/area tradeoff. This is due to the modular nature of the Residue Number System, which allows varying degrees of PLA usage.

References

[1] A. Edwin, "Direct Digital Synthesis Applications", Microwave Journal, v.33, no.1, January 1990, pp. 49-151.

[2] J. Tierney, C.M. Rader, B. Gold, "A Digital Frequency Synthesizer", IEEE Transactions on Audio and Electroacoustics, AU-19, no.1, March, 1971.

[3] F. Lu, H. Samueli, J. Yuan, C. Svensson, "A 700-MHz 24-bit Pipelined Accumulator in 1.2μ CMOS For Applications as a Numerically Controlled Oscillator", Proceedings IEEE 1991 Custom Integrated Circuits Conference, pp. 25.3.1-4.

[4] F. Lu, H. Samueli, J. Yuan, C. Svensson, "A 700-MHz 24-bit Pipelined Accumulator in 1.2μ CMOS For Applications as a Numerically Controlled Oscillator", Proceedings IEEE 1991 Custom Integrated Circuits Conference, pp. 25.3.1.

[5] W.A. Chren, Jr., "A New Residue Number System Division Algorithm", Computers and Mathematics With Applications, vol.19, no.7, pp.13-29.

[6] N.S. Szabo, R.I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, NY: McGraw-Hill, 1967.

[7] F. Lu, H. Samueli, J. Yuan, C. Svensson, "A 700-MHz 24-bit Pipelined Accumulator in 1.2μ CMOS For Applications as a Numerically Controlled Oscillator", Proceedings IEEE 1991 Custom Integrated Circuits Conference, pp. 25.3.2.

[8] N.H.E. Weste, K. Eshraghian, *Principles of CMOS VLSI Design, A Systems Perspective*, Reading, MA: Addison Wesley, 2nd Ed.,1993, p.516.

[9] N.R. Shanbag, R.E. Siferd, "A Single-Chip Pipelined 2-D FIR Filter Using Residue Arithmetic", IEEE Journal of Solid State Circuits, v.26, no.5, May 1991, pp.796-805.

[10] N.R. Shanbag, R.E. Siferd, "A Single-Chip Pipelined 2-D FIR Filter Using Residue Arithmetic", IEEE Journal of Solid State Circuits, v.26, no.5, May 1991, p.798.

[11] N.H.E. Weste, K. Eshraghian, "Principles of CMOS VLSI Design, A Systems Perspective", Reading, MA: Addison Wesley, 2nd Ed., 1993, p.294.

Using Crosscorrelation Techniques To Determine The Impulse Response Of Linear Systems

Michael J. Dallabetta Harry W. Li and Howard B. Demuth
Department of Electrical Engineering
University of Idaho, Moscow, ID 83844-1023
Office: 208-885-5548 FAX: 208-885-7579

Abstract — **A crosscorrelation method of measuring the impulse response of linear systems is presented. The technique, implementation and limitations of this method are discussed. A simple system is designed and built using discrete components and the impulse response of a linear circuit is measured. Theoretical and software simulation results are presented.**

1 Summary

There are many methods of obtaining the small signal dynamic response of linear systems. Of particular interest is the crosscorrelation method because the impulse response can be obtained while the system is being used under normal operation with no significant interference. The impulse response is measured by applying a noise level disturbance at the system input and crosscorrelating this disturbance with the system output. This method was developed in the early sixties to measure the impulse response of reactor systems [1]. However, at that time, its practical application was not possible because of difficulties with the computational techniques needed. More recently, this method has been implemented but never on a single chip [2] [3].

2 Mathematical Background

Examine Figure 1. The signal $f_i(t)$ is the normal system input and $f_n(t)$ is a noise level disturbance. The system output is

$$f_o(t) = h(t) * [f_i(t) + f_n(t)] = \int_0^\infty h(\lambda)[f_i(t-\lambda) + f_n(t-\lambda)]d\lambda \tag{1}$$

and the correlator output is

$$\Phi_{on}(\tau) = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} f_o(t+\tau)f_n(t)dt \tag{2}$$

From Eq.(1),

$$f_o(t+\tau) = \int_0^\infty h(\lambda)[f_1(t+\tau-\lambda)]d\lambda \tag{3}$$

Inserting Eq.(3) into Eq.(2) and rearranging the integrals,

$$\phi_{on}(\tau) = \int_0^\infty h(\lambda)\left\{\lim_{T\to\infty}\int_{-T}^{T}[f_n(t)f_i(t+\tau-\lambda) + f_n t f_n(t+\tau-\lambda)]dt\right\}d\lambda \tag{4}$$

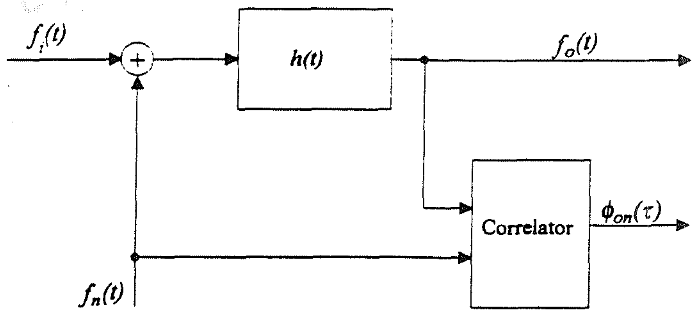


Figure 1: Determination of the unit impulse response of a system under normal use by the crosscorrelation method.

which reduces to

$$\phi_{on}(\tau) = \int_0^\infty h(\lambda)[\phi_{ni}(\tau - \lambda) + \phi_{nn}(\tau - \lambda)]d\lambda \tag{5}$$

where the term $\phi_{ni}(\tau - \lambda)$ is the crosscorrelation of $f_n(t)$ and $f_i(t)$ and $\phi_{nn}(\tau - \lambda)$ is the autocorrelation of $f_n(t)$. If $f_n(t)$ is white noise and is independent of $f_i(t)$ then

$$\phi_{nn}(\tau - \lambda) = \delta(\tau - \lambda) \tag{6}$$

and

$$\phi_{ni}(\tau - \lambda) = 0 \tag{7}$$

and Eq.(1) reduces to

$$\phi_{on}(\tau) = \int_0^\infty h(\lambda)\phi_{nn}(\tau - \lambda)d\lambda = \int_0^\infty h(\lambda)\delta(\tau - \lambda)d\lambda = h(\tau) \tag{8}$$

which is equal to the impulse response of the system under test.

3 Approximations

Practical considerations require approximations of Eq.(6) and Eq.(8). First, the correlation time must be finite and Eq.(8) reduces to

$$\hat{\phi}_{on} = \frac{1}{T}\int_0^T h(\lambda)\delta(\tau - \lambda)d\lambda \tag{9}$$

The integration time T is on the order of the settling time of the system under test which corresponds to the time for the impulse response, $h(t)$, to reach a negligible value. Second, the disturbance signal, $f_n(t)$, must have finite bandwidth and still retain an autocorrelation that approximates an impulse. Signals having this property can be generated from binary maximal-length shift register sequences (m-sequences). Binary m-sequences are very easy to generate and are well documented [4, 5, 6]. An example of a 15 bit m-sequence signal is shown in Figure 2.

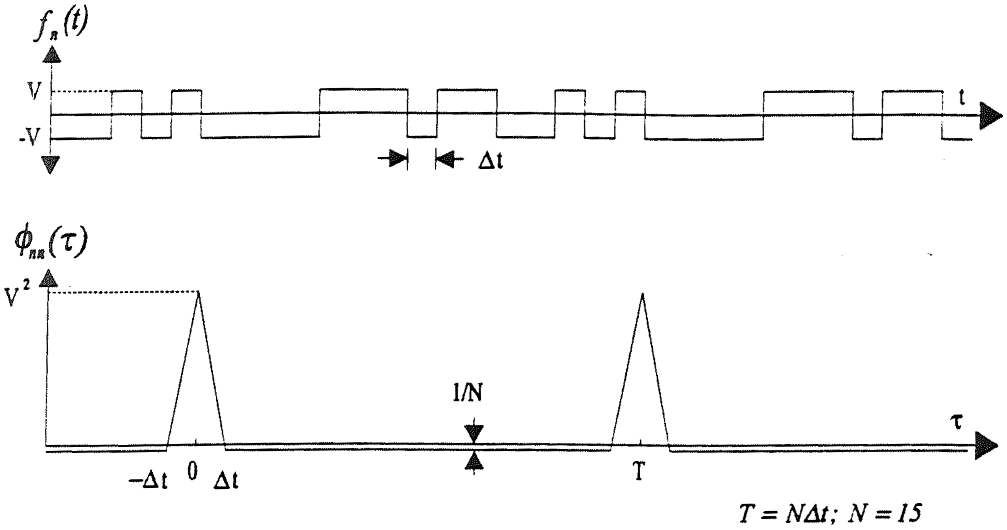


Figure 2: A 15 bit binary m-sequence signal and its autocorrelation function.

Note the following properties of the binary signal $f_n(t)$:

1. It has values of $+V$ and $-V$ with equal probability.
2. It may (but not necessarily) change sign only every Δt seconds.
3. It is periodic with period $T = N\Delta t$.

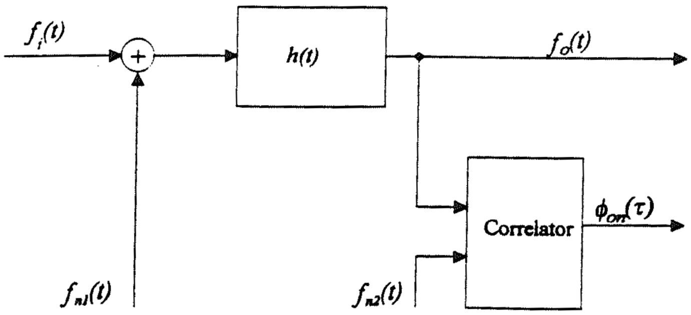


Figure 3: Determination of the unit impulse response of a system under normal use by the crosscorrelation method and two unique disturbance signals.

An improved system can be obtained by slightly modifying Figure 1. so that $f_n(t)$ is separated into two m-sequence signals as seen in Figure 3. The two m-sequence signals, shown in Figure 4, are derived from the classic binary m-sequences except that the signal voltage levels have been changed and their duty cycles can be adjusted. The modified voltage

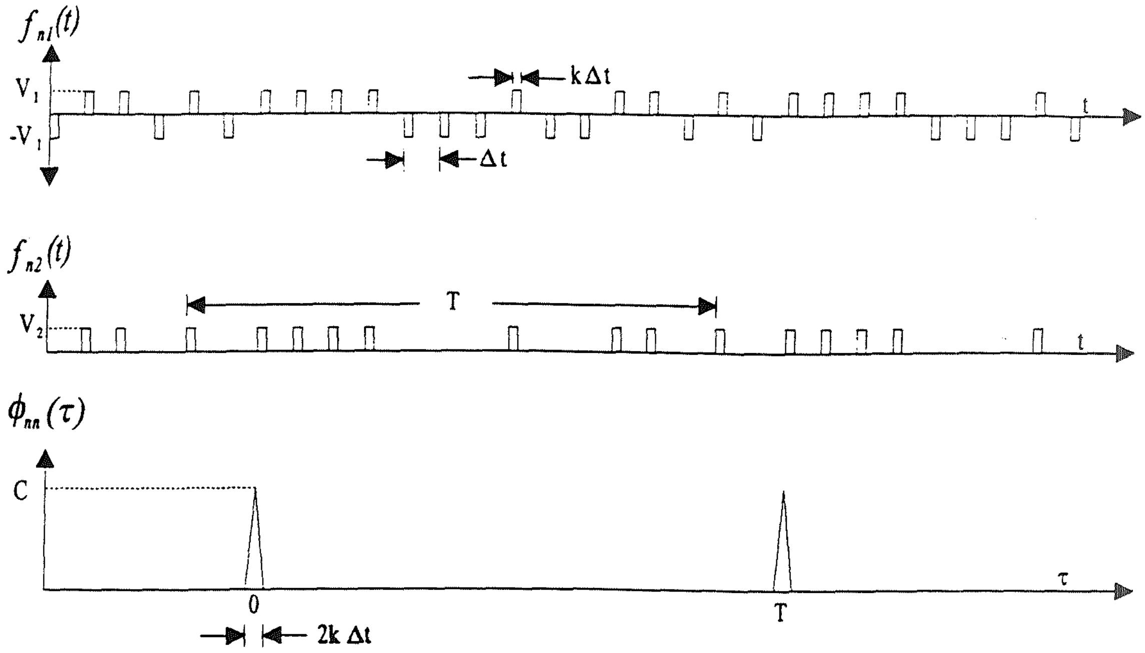


Figure 4: The cross correlation of two unique m-sequence random impulse trains.

levels eliminate the out of phase autocorrelation offset shown in Figure 2 and the duty cycle adjustment allows the pulse width, $k\Delta t$, to be adjusted independently of the period T.

To determine the correlator output, consider Eq.(8). One interpretation of Eq.(8) is that the correlator output is equal to the convolution of the system impulse response, $h(t)$, and $\phi_{nn}(t)$. The graphical description of Eq.(8) is shown in Figure 5. and it is clear that the period T of the random signal must be sufficiently long to allow the impulse response to decay to a negligible value which is the approximation required by Eq.(9).

If Eq. (8) is interpreted in the frequency domain, then

$$\phi_{on}(j\omega) = H(j\omega) \cdot \phi_{nn}(j\omega) \tag{10}$$

A graphical description of Eq.(10) is shown in Figure 6 and it is clear that the random signals approximate white noise over a limited bandwidth only. This bandwidth (the -3dB frequency) is easily calculated and is shown in Figure 6. as f_c. The parameter $k\Delta t$ must be sufficiently small so that the bandwidth of $\phi_{on}(j\omega)$ is much greater than the bandwidth of the system under test. The extracted impulse response can be made arbitrarily accurate, within the limits of the hardware used, simply by reducing $k\Delta t$. It should be clear why random disturbance signals with an adjustable duty cycle are desirable since this allows the period of the signals to be adjusted independently of the pulse width $k\Delta t$.

If the random signal period T is sufficiently large and $k\Delta t$ is sufficiently small, than $k\Delta t$.

$$\phi_{on}(\tau) = \frac{8}{15}(k)\Delta t V_1 V_2 \cdot h(\tau) \tag{11}$$

a result that is proportional to the system impulse response.

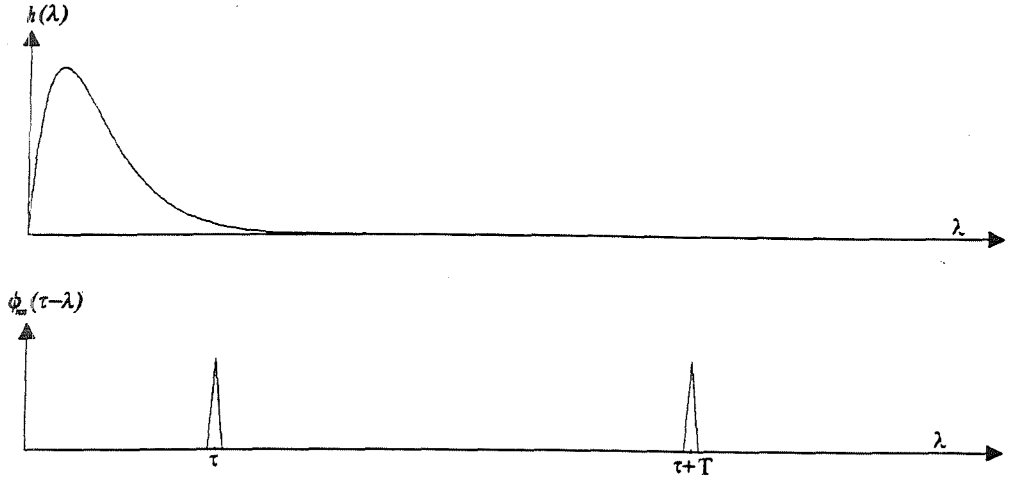


Figure 5: Graphical description of Eq. (8); The convolution of the system impulse response and the random signal crosscorrelation.

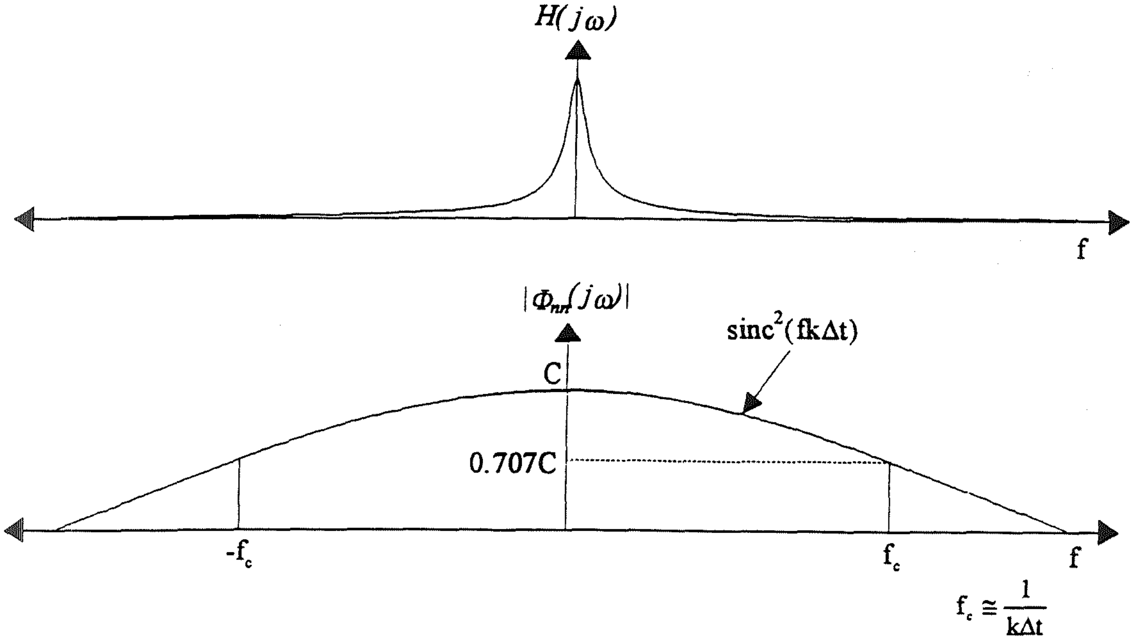

$H(j\omega)$
f
$|\Phi_{nn}(j\omega)|$
$\mathrm{sinc}^2(fk\Delta t)$
C
0.707C
$-f_c$
f_c
f
$f_c \cong \frac{1}{k\Delta t}$

Figure 6: Graphical description of Eq (10); The frequency domain representation of Eq.(8).

4 System Design

A simple INTeractive Impulse Response Extraction system (INTIRE) that utilizes a 15 bit m-sequence has been designed and is being built using discrete components. A high level block diagram of the system is shown in Figure 7. Although not shown, the system will utilize an external clock and independent analog and digital voltage supplies. The system is designed to provide discrete time samples of the impulse response of the system under test.

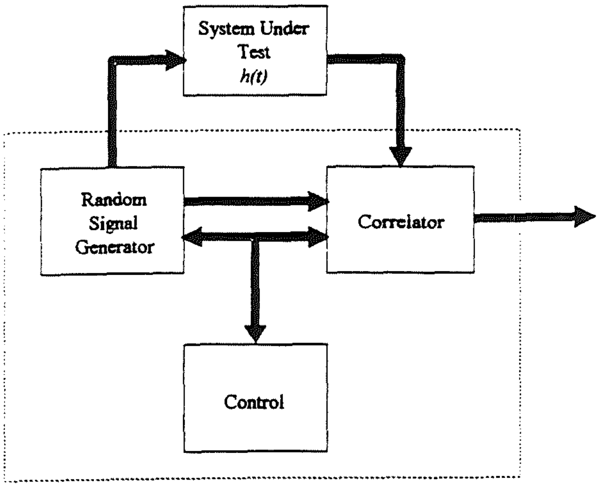


Figure 7: High level block diagram of the interactive impulse response extraction system.

4.1 Control Signals Block

A timing diagram of the Control Block output signals is shown in Figure 8 along with the system clock for reference. The Pulse Signal is simply a binary pulse train with an adjustable duty cycle that is synchronous with the system clock. This pulse signal is used to generate the random signals shown in Figure 4. The remaining three signals control the correlator so that the following sequence of events occurs:

1. The random signal $f_{n1}(t)$ is crosscorrelated with the output of the system under test for exactly one period of the random signal (Enable Correlator HIGH).
2. The correlator is disabled and the discrete time samples of the extracted impulse response are multiplexed to a single output pin (Mux Enable HIGH).
3. The outputs of the correlator are reset to 0 volts.

An updated extracted impulse response will be provided once every three periods of the random signal. Note that the period of the random signal, T, is equal to 15 cycles of the system clock since a 15 bit random sequence is used.

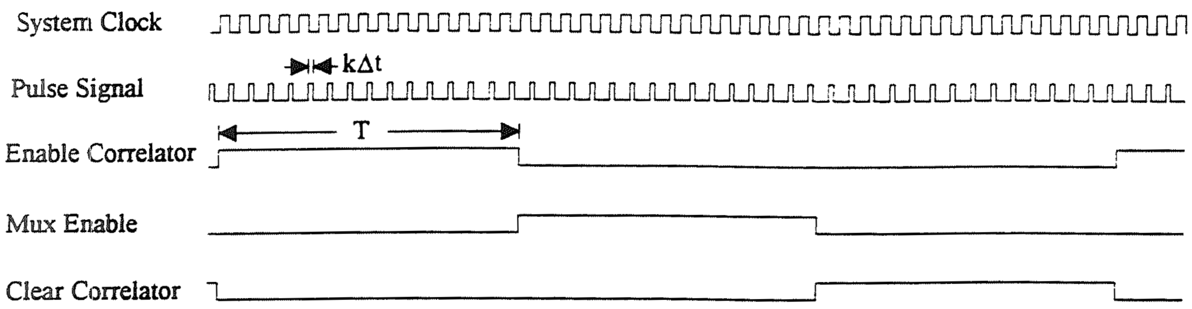


Figure 8: Timing diagram of the INTIRE system Control Block.

4.2 Random Signal Generator Block

The Random Signal Generator (RSG) block will provide the random signal pulse train $f_{n1}(t)$ shown in Figure 4. A schematic of the (RSG) is shown below in Figure 9. The Binary Random Signal Generator in Figure 9. is the classic m-sequence generator discussed in [4] and will provide the random signal $f_n(t)$ shown in Figure 2. This signal and the pulse signal provided by the Control block will drive the SW-1 and SW-2 switches to generate the desired signal. The output of SW-2 is then attenuated to noise level. A problem with this attenuation is that $f_{n1}(t)$ will be inherently be corrupted by white noise in the process. However, as will be demonstrated, a high level of noise corruption can be tolerated. Furthermore, since the duty cycle of the $f_{n1}(t)$ can be adjusted, noise level signals can be achieved with voltages on the order of 1 to 10 mV which sill minimize the noise.

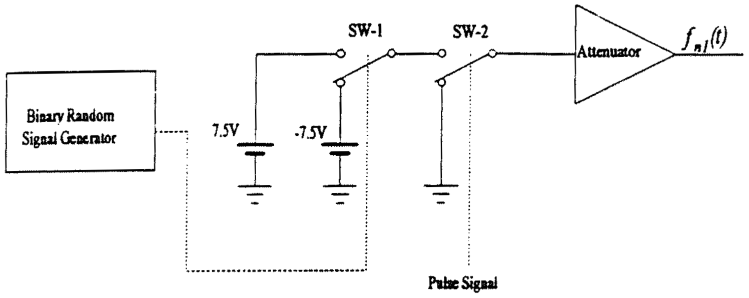


Figure 9: Schematic of the INTIRE system random signal generator

4.3 The Correlator Block

Generally, the realization of the correlation function is very difficult. However, in this case, the correlator design is very simple since the random signal $f_n(t)$ shown in Figure 2 is binary, This fact simplifies the delay and multiplication functions necessary in the correlator. To explain, consider the correlator schematic shown in Figure 10. To perform the correlation, delayed versions of the $f_n(t)$ must be stored. which is a simple matter for binary signals. The SW1 switches implement the multiplication of the output of the system under test, $f_o(t)$. and the delayed signals, $f_n(t-\tau_i)$. The signal $f_o(t)$ is multiplied by either a one, corresponding to a logic HI. or a zero, corresponding to a logic LOW. The SW2 switches multiply the output of SW1 by the pulse train shown in Figure 8. which converts the signal correlated with the system output from $f_n(t)$ to $f_{n2}(t)$ shown in Figure 4. The SW3 switches, controlled by the Enable Correlator signal shown in Figure 8. are cycled so that the product, $f_{n2}(t)f_o(t)$, is integrated over exactly one period of the random signal as required by Eq.9. At the end of the integration, the value of the integrator outputs is proportional to $h(\tau_i)$; the value of the impulse response of the system evaluated at τ_i. The integrator outputs can be thought of as discrete time samples of the impulse response $h(t)$. If there are a sufficient number of samples. all characteristics of $h(t)$ are preserved. For convenience, these discrete time samples are multiplexed out to a single pin.

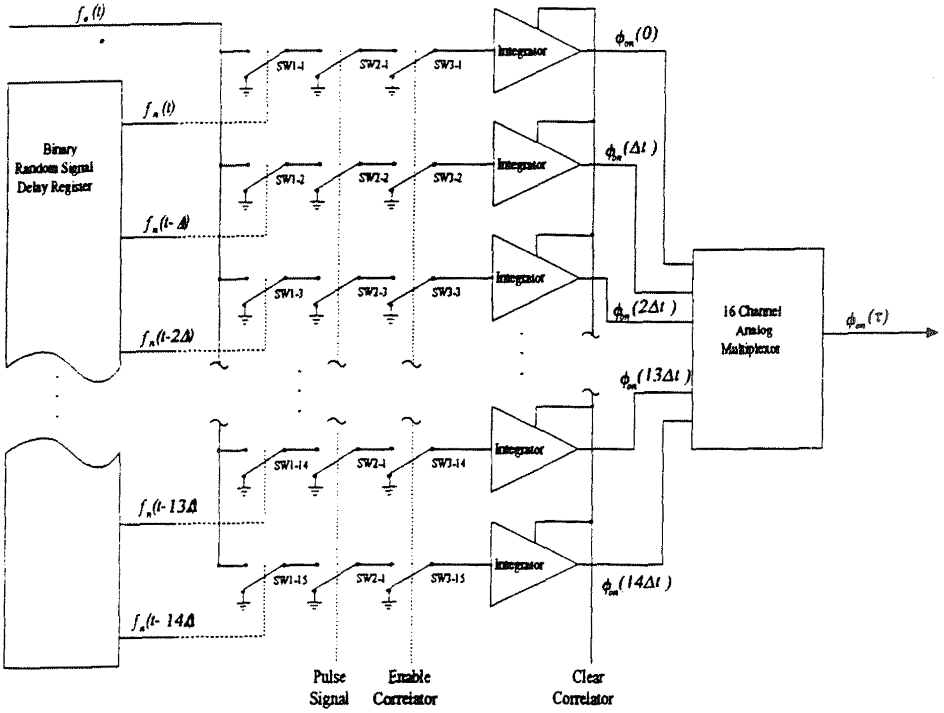


Figure 10: Schematic of the INTIRE system correlator

5 Simulation Results

The INTIRE system was modeled and simulated using Analogy's Saber program. The system was used to extract the impulse response from a simple RLC lowpass filter. The extracted impulse response samples are shown in Figure 11(a) along with the ideal impulse response. Clearly, The extracted samples are very accurate as can be seen by the standard deviation. Since the duty cycle of the random signal can be adjusted, the accuracy of the samples is only limited by the speed of the hardware used.

The results in Figure 11(a) were obtained using a perfect, noise free, random disturbance signal. However, a distorted random signal would yield similar results. To demonstrate, a random disturbance signal, shown in Figure 11(b), that was severely corrupted by white gaussian noise was applied to the RLC network. The Signal to Noise Ratio was 0 dB. The extracted impulse response samples are shown in Figure 11(c). Even with this severe noise distortion, the extracted signal is an accurate model of the RLC network impulse response. This is to be expected since the disturbance signal, $f_{n1}(t)$ and the white noise are independent. Hence, there crosscorrelation should equal zero.

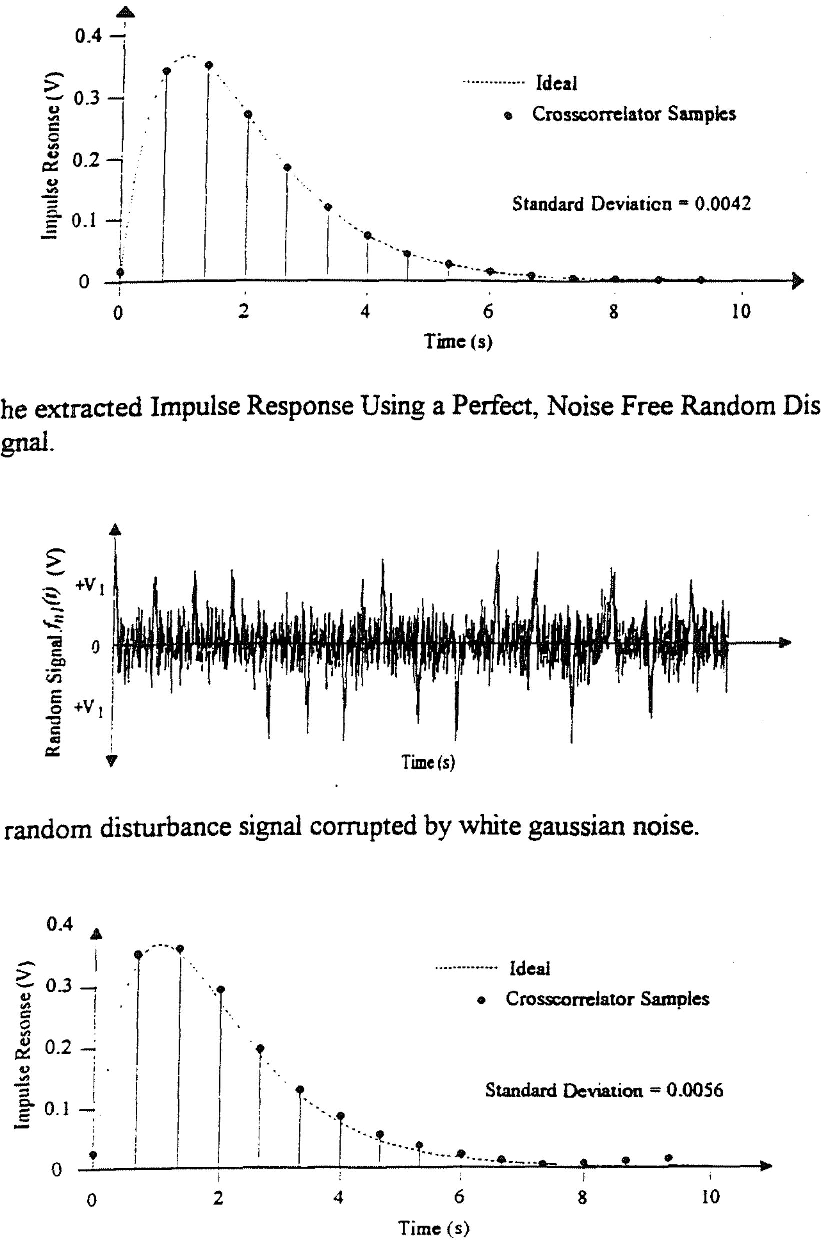


Fig. 11(a) The extracted Impulse Response Using a Perfect, Noise Free Random Disturbance Signal.

Fig. 11(b) A random disturbance signal corrupted by white gaussian noise.

Fig. 11(c) The extracted impulse response using a random disturbance signal corrupted by white gaussian noise.

Figure 11:

6 Conclusion

A method of extracting the impulse response of linear systems is presented. A system was designed and simulated and it was found that highly accurate samples of a system impulse response could be obtained. Furthermore, noise corruption of the disturbance signal had little effect on the extracted impulse response which makes the interactive use of this method more feasible. There are many applications for this method which include detection of linear system degradation, design of self-correcting automatic compensating analog systems, system self diagnostics and monitoring and testing of mechanical systems. Currently, a system is being designed that utilizes a 15 bit m-sequence disturbance signal. Using such a short m-sequence allows verification of the concepts while keeping the system small enough so that it can be constructed using discrete components. Once this design approach is verified, another system with wider applications will be designed and integrated so that its unique characteristics can be exploited.

Acknowledgments

This research was supported in part by NASA under Space Engineering Research Grant NAGW-1406.

References

[1] J.D. Balcomb, H.B. Demuth and E.P. Gyftopoulos, “A crosscorrelation method for measuring the impulse response of reactor systems,” Nuclear Science and Engineering, Volume 11, No.2, pp. 159-166, October 1961.

[2] D.A. Barr, S.J. Laverty, “Impulse response of a DCEL ZnS: Mn, Cu powder display determined by cross correlation, ”IEEE Proceedings - G, Volume 138, No.2, April 1991.

[3] C. Yap, “A background method for monitoring system performance based on cross correlation using the MC68HC11 single-chip microcomputer”, A Master of Science thesis in the discipline of Electrical Engineering, The University of Tulsa, 1987.

[4] F.J. MacWilliams, N.J.A. Sloane, “Pseudo-random sequences and arrays”, IEEE Proceedings, Volume 64, No. 12, December 1976.

[5] S.W. Golomb, Ed., Digital Communications with Space Applications. Englewood Cliffs, N.J.,: Prentice-Hall, 1964.

[6] K. R. Godfrey, “Three-level m-sequences”, Electron Letters. Volume 2, No.7, pp. 241-243, July 1966.

Omit

Session 11
Synthesis II

Sterling Whitaker

Automated Synthesis and Verification of Configurable DRAM Blocks for ASICs.[1]

M. Pakkurti, A. G. Eldin, S. C. Kwatra and M. Jamali
Electrical Engineering Department
The Univeristy of Toledo
Toledo OH 43606

Abstract - **A highly flexible embedded DRAM compiler is developed which can generate DRAM blocks in the range of 256 bits to 256 Kbits. The compiler is capable of automatically verifying the functionality of the generated DRAM modules. The fully automated verification capability is a key feature that ensures the reliability of the generated blocks. The compiler's architecture, algorithms, verification techniques and the implementation methodology are presented.**

1 Introduction

The use of Computer Aided Design (CAD) tools for the design and verification of custom VLSI circuits has become essential in their development process [1]. The present development of these tools has concentrated on automated synthesis and verification of designs [1]. Special-purpose generators and compilers for the design of structured systems with high regularity, like the Static RAM, have been developed [2],[3]. Majority of these generators lack an automated verification capability and are thus restricted to use only by expert IC designers. This is because the generated modules need to be accurately simulated and its layout checked for DRC and ERC violations. The DRC and ERC checking does not need significant IC design experience and can be performed easily by the system or chip designer. However, because these generators are capable of producing a very large number of different modules, the most reliable and commonly used verification method is by accurate simulation of the generated modules. Since the system designer does not know the details of the module's circuit design and the expertise to verify such a design, the verification process must be automated. Even for the expert IC designer, automating the verification part reduces dramatically, the development time and eliminates any possibility of human error in this process.

The design methodology, described in this paper, automates the synthesis of DRAM modules and the verification of their functionality [4]. The methodology, however, can be applied to any structured logic or memory block. The software for implementing the method is developed for a workstation environment using C and Genie/Lx languages. Generator Development Tools (GDT) from Mentor Graphics are used in the development of this DRAM compiler. A graphical interface is used to input the user specification, communicate any messages and provide the status of the design.

[1] This research is partially supported by NASA grant, NAG3-799 and is part of M.S. reseach of M. Pakkurti

2 Design Methodology

The successful implementation of a design methodology for automatic synthesis and verification of functional blocks has several basic requirements. These include the use of a flexible graphical user interface and reliable algorithms for synthesis and automatic verification.

The design methodology is shown in Figure 1 and Figure 2. The graphical user- interface is implemented in the Openwindows^{TM} system. The software utilizes the window and dialog box facilities to implement a user-friendly interface. The user- interface interacts directly with the organization program for the circuit to be synthesized. The organization program incorporates the structural decomposition algorithm for the module needed. The organization algorithm, in the case of the DRAM generator, takes as inputs the total number of words and the number of bits per word [4]. The program supports DRAM sizes in the range of 256 bits to 256 Kbits. The user input is checked by the organization program to verify that it is within this range. The input is then used to calculate the number of basic blocks, sense amplifiers per block, size of decoders and other structural parameters needed to synthesize the DRAM layout [4]. The parameters calculated are passed to lower hierarchical level generators of each sub-circuit in the DRAM module. For example in the case of the row decoder the number of the row address lines is passed to a generator which synthesizes the row decoder. The availability of all structural parameters enables the calculation of the physical size of the DRAM module. The access and cycle times are calculated using theoretical equations and reported together with the physical size to the user. If the initial estimates are acceptable to the user, then accurate simulation of critical data and address paths of the DRAM module is performed automatically. The circuit schematic generation is done using Genie/Lx programs which consist of the parameterized structural algorithms of the sub-circuits and the top level critical path circuit.

The circuit functional verification is done automatically. The netlist of the circuit is extracted from the schematic and this includes the size information of all transistors and parasitics in the circuit. The mixed-mode simulator Lsim^{TM} is used for transistor level simulation of the critical path circuit. The simulations are performed automatically with no user intervention using the non-interactive batch mode of simulation. The software generates all the initialization and control files for the simulator. Circuit specific initializations are created for each sub-circuit. The output of the simulation is stored as waveform files for all signals in the circuit and also tabular data is created for all signals as time and voltage pairs. The automatic verification stage is invoked after successful simulation of the circuit.

The methodology for automatic circuit verification is shown in Figure 3. The verification and analysis module takes the simulation output and a user input rule file to perform the verification. The rule file contains statements written in a special format to describe the intended behavior of the circuit. During the development stage of the DRAM compiler, a template rule file is created by the system into which the DRAM designer enters the rules using a text editor. These rules describe the expected sequence of events of critical signals that take place during the simulation of the DRAM circuit. The program checks the format of each rule in the file to ensure that they conform to the rule standard. In case of errors in a rule format, helpful messages are given to help track the format error. The first line in a rule file is present in the template file and it lists all the signals names available in the circuit.

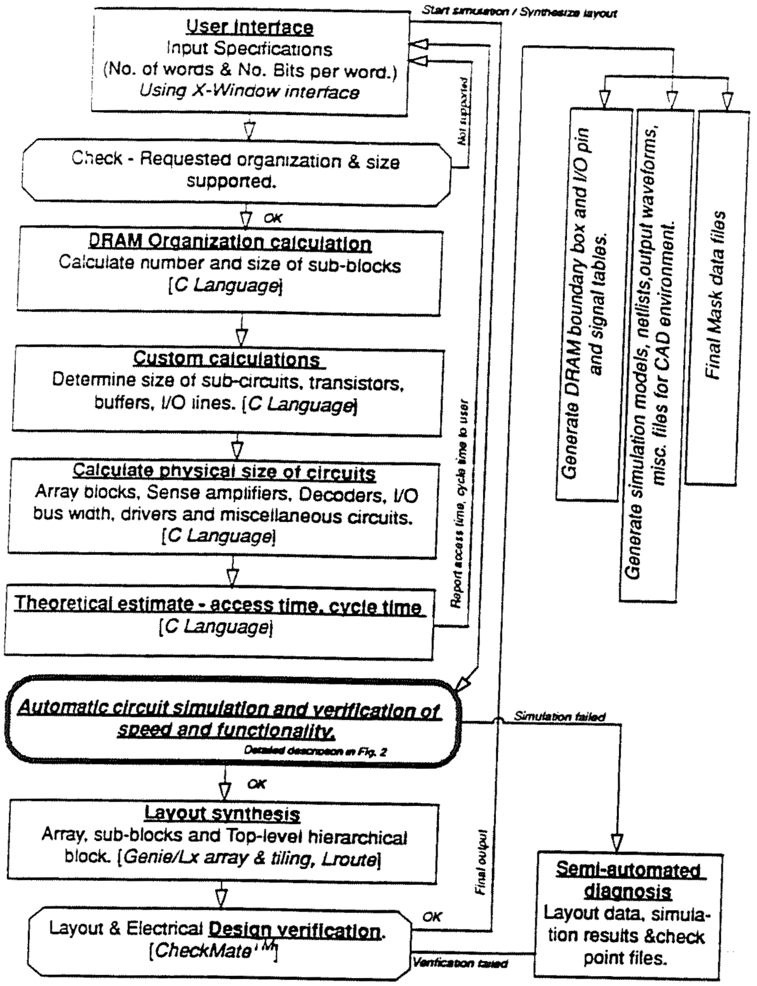

User Interface
Input Specifications
(No. of words & No. Bits per word.)
Using X-Window interface
Start simulation / Synthesize layout
Not supported
Check - Requested organization & size supported.
OK
DRAM Organization calculation
Calculate number and size of sub-blocks
[C Language]
Custom calculations
Determine size of sub-circuits, transistors, buffers, I/O lines. [C Language]
Calculate physical size of circuits
Array blocks, Sense amplifiers, Decoders, I/O bus width, drivers and miscellaneous circuits.
[C Language]
Theoretical estimate - access time, cycle time
[C Language]
Report access time, cycle time to user
Generate DRAM boundary box and I/O pin and signal tables.
Generate simulation models, netlists,output waveforms, misc. files for CAD environment.
Final Mask data files
Automatic circuit simulation and verification of speed and functionality.
Detailed description in Fig. 2
Simulation failed
OK
Layout synthesis
Array, sub-blocks and Top-level hierarchical block. [Genie/Lx array & tiling, Lroute]
Final output
Semi-automated diagnosis
Layout data, simulation results &check point files.
Layout & Electrical Design verification.
[CheckMate™]
OK
Verification failed

Figure 1: Detailed Compiler Methodology

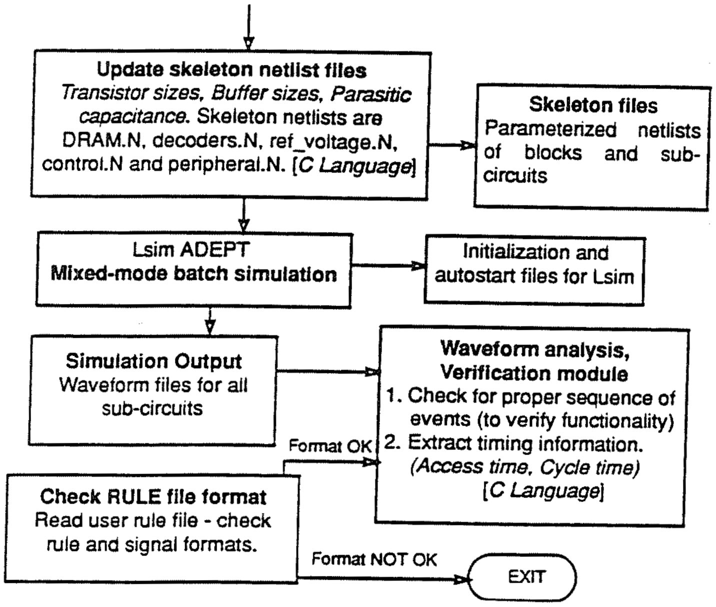

Update skeleton netlist files
Transistor sizes, Buffer sizes, Parasitic capacitance. Skeleton netlists are DRAM.N, decoders.N, ref_voltage.N, control.N and peripheral.N. [C Language]
Skeleton files
Parameterized netlists of blocks and sub-circuits
Lsim ADEPT
Mixed-mode batch simulation
Initialization and autostart files for Lsim
Simulation Output
Waveform files for all sub-circuits
Waveform analysis, Verification module
1. Check for proper sequence of events (to verify functionality)
2. Extract timing information. (Access time, Cycle time) [C Language]
Format OK
Check RULE file format
Read user rule file - check rule and signal formats.
Format NOT OK
EXIT

Figure 2: Automatic verification and waveform analysis

A sample rule file is shown in Figure 4. The signal names are used to specify the behavior. Each line starts with an optional word for the DRAM designer to keep track of the rules in the file. The rest of the line specifies the behavior. The basic entity in a rule is the signal event pair. Each signal can be at one of the four events LO, HI, HL, LH. The rule format at the present time can be used to specify sequential events. Each pair of signal name and event are processed by the algorithm during actual verification. The verification based on the signal name and event is done in a relative basis. This means that each pair specified in the rule is checked sequentially while all other unspecified intermediate events are ignored in the verification process without flagging any errors. The relative mode (REL) is the default. The absolute mode keyword (ABS) is used to describe a sequence of events, that matches exactly the sequence of all events stored in the simulation event table. This mode enables verifying time critical events with more strict timing constraints than in the relative mode during a part of the simulation. The REL and ABS keywords can be used in a single rule line to switch back and forth between the two modes. One other feature in the rule format is the MONITOR ONLY pair. The keywords can be used to monitor a particular signal for events which will not significantly affect the behavior of the circuit, but the DRAM designer at the debugging stage may need to monitor them for design optimization purposes. A rule file can contain any number of rules and the software attempts to validate each rule line.

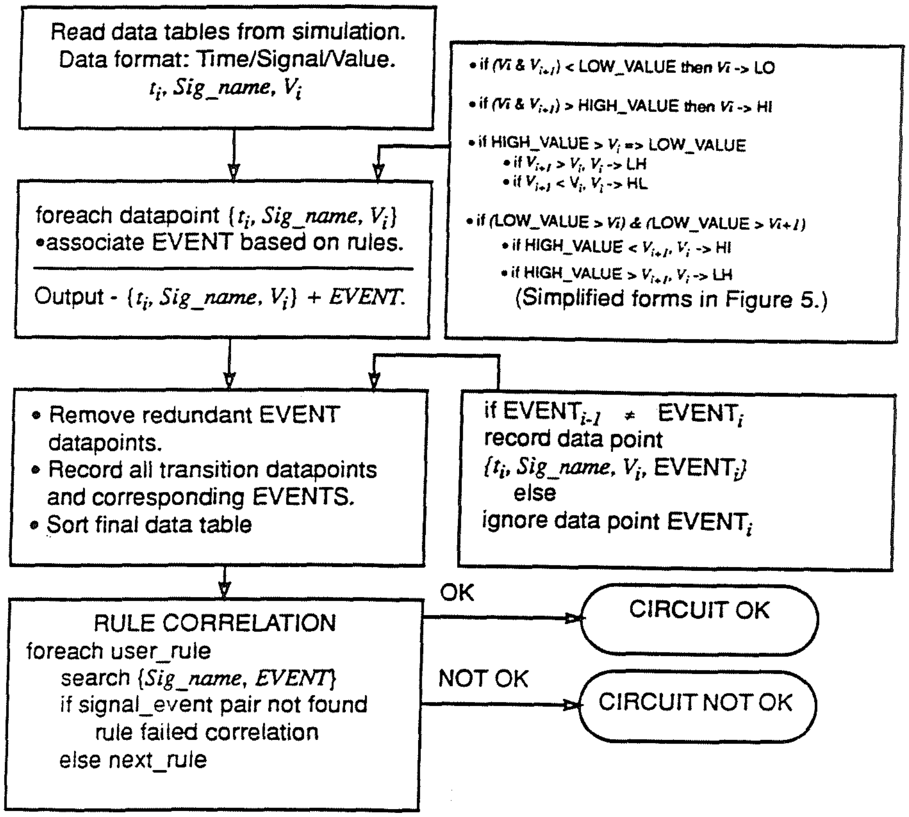
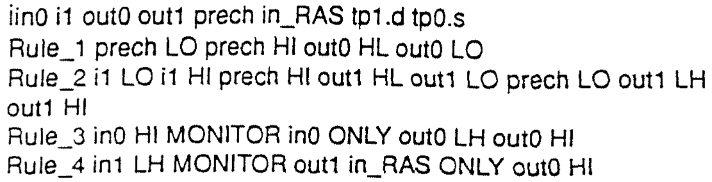

Read data tables from simulation.
Data format: Time/Signal/Value.
t_i, Sig_name, V_i
• if (V_i & V_{i+1}) < LOW_VALUE then V_i -> LO
• if (V_i & V_{i+1}) > HIGH_VALUE then V_i -> HI
• if HIGH_VALUE > V_i => LOW_VALUE
• if $V_{i+1} > V_i$, V_i -> LH
• if $V_{i+1} < V_i$, V_i -> HL
• if (LOW_VALUE > V_i) & (LOW_VALUE > V_{i+1})
• if HIGH_VALUE < V_{i+1}, V_i -> HI
• if HIGH_VALUE > V_{i+1}, V_i -> LH
(Simplified forms in Figure 5.)
foreach datapoint {t_i, Sig_name, V_i}
•associate EVENT based on rules.
Output - {t_i, Sig_name, V_i} + $EVENT$.
• Remove redundant EVENT datapoints.
• Record all transition datapoints and corresponding EVENTS.
• Sort final data table
if EVENT$_{i-1}$ ≠ EVENT$_i$
record data point
{t_i, Sig_name, V_i, EVENT$_i$}
else
ignore data point EVENT$_i$
RULE CORRELATION
foreach user_rule
search {Sig_name, $EVENT$}
if signal_event pair not found
rule failed correlation
else next_rule
OK
CIRCUIT OK
NOT OK
CIRCUIT NOT OK

Figure 3: Flow chart for EVENT classification and RULE correlation

The analysis program uses the data created by the simulator. The table data is created by Lsim and is stored in ASCII files. The data is pre-processed to remove some simulator



```
iin0 i1 out0 out1 prech in_RAS tp1.d tp0.s
Rule_1 prech LO prech HI out0 HL out0 LO
Rule_2 i1 LO i1 HI prech HI out1 HL out1 LO prech LO out1 LH
out1 HI
Rule_3 in0 HI MONITOR in0 ONLY out0 LH out0 HI
Rule_4 in1 LH MONITOR out1 in_RAS ONLY out0 HI
```

Figure 4: Rule file for row decoder

after pre-processing is available in data tables for processing by the analysis algorithms. The algorithm uses a set of rules which perform operations of smoothing the data and classify each data point as one of four events LO (low), HI (high), LH (low to high) and HL (high to low). The data points recorded during the simulation are now represented as a time value in nano-seconds, and signal name and value or voltage at that time. This data is recorded for each point in simulation time. The resolution of the time in the data is controlled in the earlier stage of simulation. Presently the time step is 10 pico-seconds. The raw data from the simulation after processing has been transformed into data points and events are associated with each point. Each data point is processed in the following method. The comparison and processing at this stage is done using the signal value or voltage. Two threshold values HIGH_VALUE and LOW_VALUE are used to represent the voltage for HIGH and LOW logical states in that technology. The algorithm used to associate events to the data points is shown in Figure 5.

The above event assignment is done for each data point. The data table after the events have been added is in an intermediate form. This intermediate form of data is processed further to remove redundant events. Since each data point has been associated with an event, signals have events even if there are no transitions. The software needs to record only the transitions in each signal to check the behavior. The redundant events and corresponding data are removed as shown in Figure 3. At this stage the data table contains all transition events and corresponding times have been recorded. The next stage is the actual behavior verification. The rule file is used as input and each rule is processed. The program extracts the first signal and event pair from a rule and correlates them with the final event table. If a match is found then the next pair is checked till the end of the rule file. If any pair fails to match then an error is flagged for the rule and the next rule is processed till all the rules have been done.

The row decoder is used to demonstrate the automatic verification process. The schematic and simulation waveform of the row decoder is shown in Figure 6. The rule file for the circuit is shown in Figure 4, and sample output from the program is shown in Figure 7. The first line in the rule file lists the signals in the circuit. Rule_1 specifies that signal prech has a state LO and it changes state to HI. This is followed by signal out0 changing state from HI to LO in relative (REL) mode.The algorithm compares the rule description with the simulation event table. An example of a monitor is shown in Rule_3. Sample output from the program is shown in Figure 7. The output shows the data and messages from the program which implements the algorithm described.

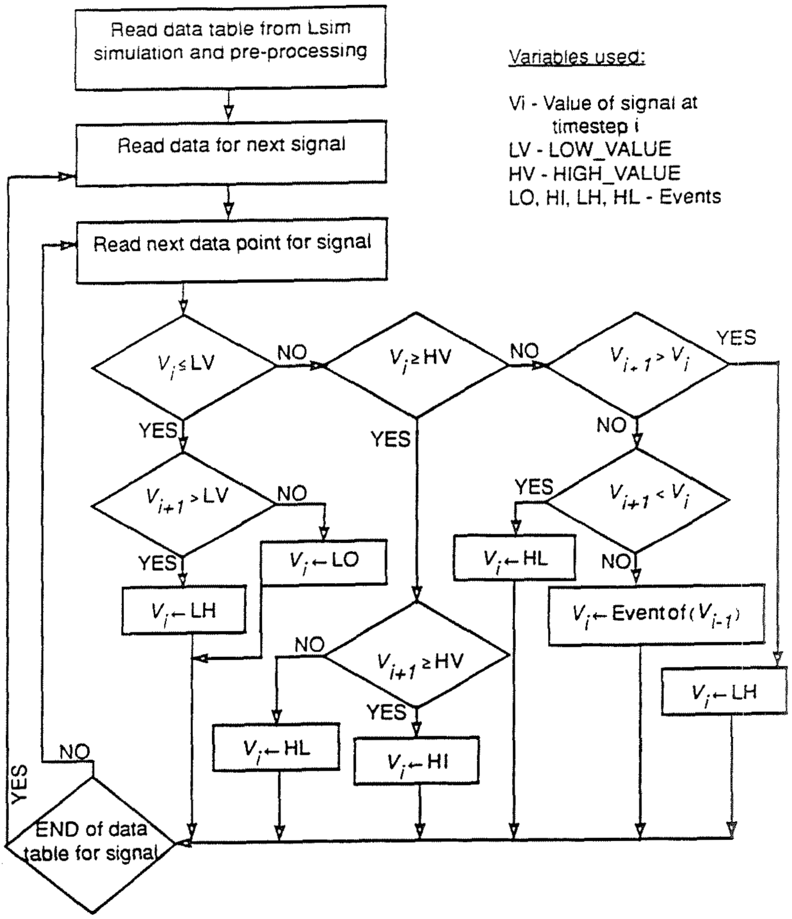


Figure 5: Algorithm for associating events to data points

This automatic verification process is essential to check the functionality of the top level circuit of the generated module. Once the synthesized module passes this test, The following stage is the actual layout generation. Genie/Lx programs have been written which can generate the layout of circuits based on the information calculated in the organization programs. DRAM cell array blocks and peripheral sub-blocks are synthesized by lower level generators. The block level routing is done using AutoRoute$^{\text{TM}}$. The complete layout of the DRAM is built using a combination of programs written in Lx and facilities provided in GDT. The layout of the completed module is checked for electrical and physical design rule violations.

The failure of any of the above stages triggers an error reporting mechanism which generates information for analysis by an expert designer of the module. This includes parameters calculated by the organization program, the input files for the simulator, the output of the simulation. ERC and DRC errors.

If the generated module passes all these tests, the different design files are provided to the user as shown in Figure 1. The results of the accurate simulation are provided as waveform files.

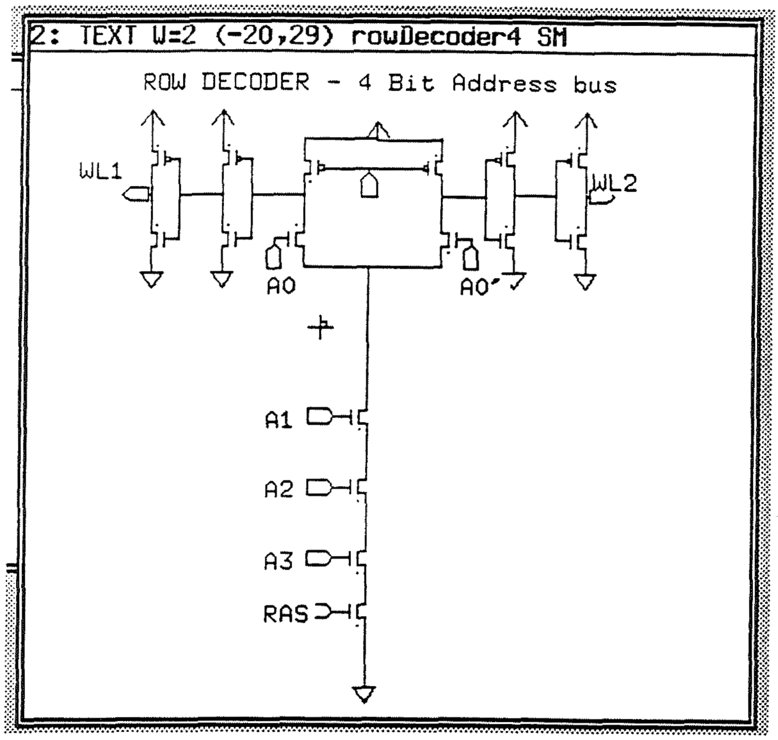

2: TEXT W=2 (-20,29) rowDecoder4 SM
ROW DECODER - 4 Bit Address bus
WL1
WL2
A0
A0'
A1
A2
A3
RAS

Figure 6(a). Schematic of the automatically generated row decoder.

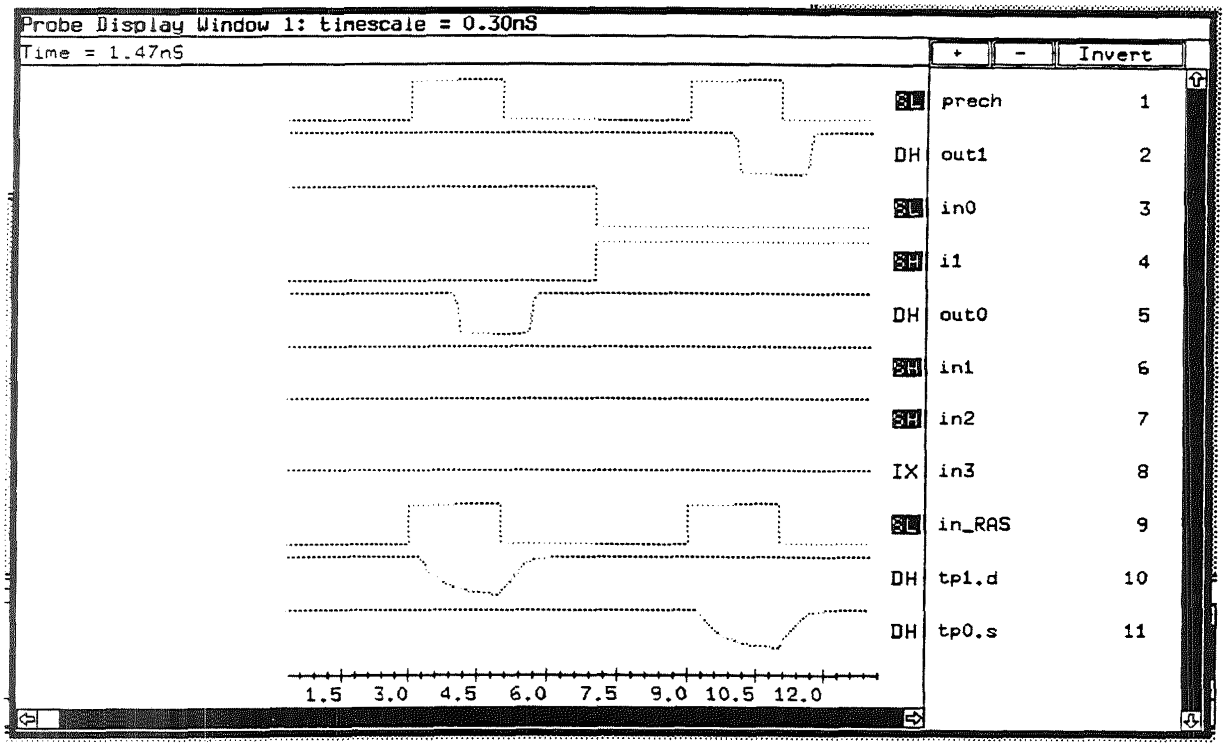
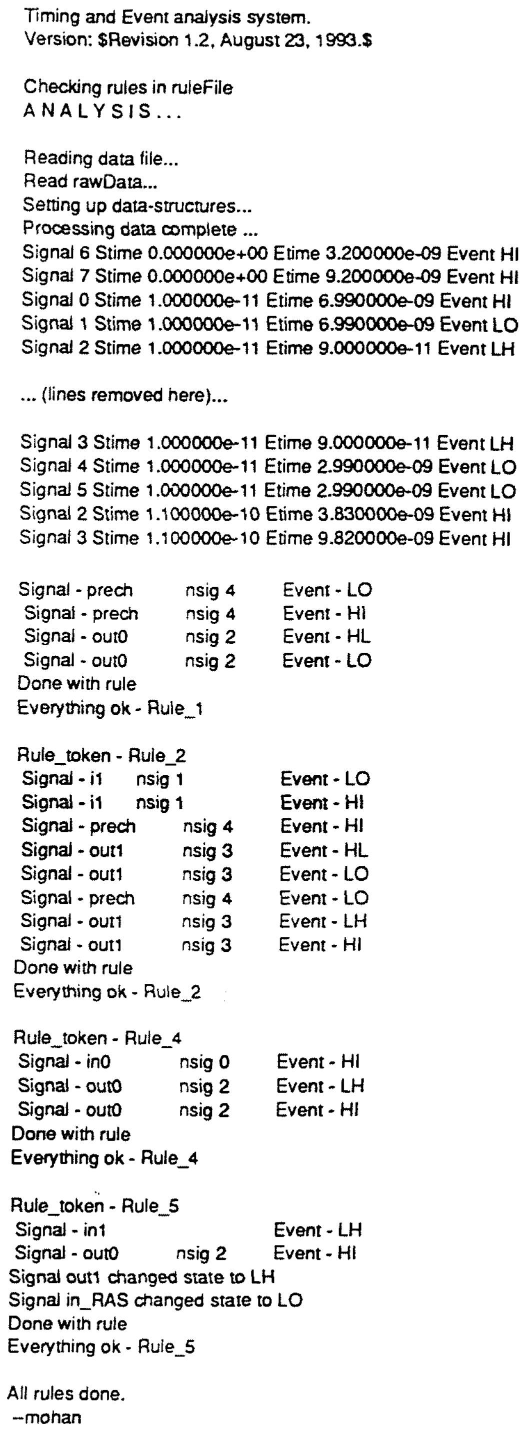


Figure 6(b). Simulation waveform of the row decoder.



```
Timing and Event analysis system.
Version: $Revision 1.2, August 23, 1993.$

Checking rules in ruleFile
ANALYSIS...

Reading data file...
Read rawData...
Setting up data-structures...
Processing data complete ...
Signal 6 Stime 0.000000e+00 Etime 3.200000e-09 Event HI
Signal 7 Stime 0.000000e+00 Etime 9.200000e-09 Event HI
Signal 0 Stime 1.000000e-11 Etime 6.990000e-09 Event HI
Signal 1 Stime 1.000000e-11 Etime 6.990000e-09 Event LO
Signal 2 Stime 1.000000e-11 Etime 9.000000e-11 Event LH

... (lines removed here)...

Signal 3 Stime 1.000000e-11 Etime 9.000000e-11 Event LH
Signal 4 Stime 1.000000e-11 Etime 2.990000e-09 Event LO
Signal 5 Stime 1.000000e-11 Etime 2.990000e-09 Event LO
Signal 2 Stime 1.100000e-10 Etime 3.830000e-09 Event HI
Signal 3 Stime 1.100000e-10 Etime 9.820000e-09 Event HI

Signal - prech      nsig 4      Event - LO
 Signal - prech     nsig 4      Event - HI
 Signal - out0      nsig 2      Event - HL
 Signal - out0      nsig 2      Event - LO
Done with rule
Everything ok - Rule_1

Rule_token - Rule_2
 Signal - i1     nsig 1         Event - LO
 Signal - i1     nsig 1         Event - HI
 Signal - prech         nsig 4  Event - HI
 Signal - out1          nsig 3  Event - HL
 Signal - out1          nsig 3  Event - LO
 Signal - prech         nsig 4  Event - LO
 Signal - out1          nsig 3  Event - LH
 Signal - out1          nsig 3  Event - HI
Done with rule
Everything ok - Rule_2

Rule_token - Rule_4
 Signal - in0           nsig 0  Event - HI
 Signal - out0          nsig 2  Event - LH
 Signal - out0          nsig 2  Event - HI
Done with rule
Everything ok - Rule_4

Rule_token - Rule_5
 Signal - in1                   Event - LH
 Signal - out0          nsig 2  Event - HI
Signal out1 changed state to LH
Signal in_RAS changed state to LO
Done with rule
Everything ok - Rule_5

All rules done.
 --mohan
```

Figure 7. Sample output from analysis of row decoder simulation data.

3 Conclusion

The design methodology for automatic synthesis and design verification of embedded DRAM blocks is presented. The details of a novel implementation of the automatic simulation and functional verification are presented. This includes assigning logical events for different simulation points. removing redundant events and comparison with designer's rules. Simplified rules language is developed to accurately describe the correct logical and timing behavior of the system. This automated simulation and verification capability is essential in developing highly reliable compilers. It also can be used in a feedback loop to automatically correct any design errors and optimize the performance of generated blocks. This methodology is applicable to any memory or other functional compiler.

References

[1] A. R. Newton, A. L. Sangiovanni-Vincentelli, "Computer-Aided Design for VLSI Circuits," IEEE Computer, Vol. 19, No. 4, April 1986, pp. 38-60.

[2] Jarvis C. Tou et al., "A Submicrometer CMOS Embedded Compiler," IEEE Journal of Solid-State Circuits, Vol. 27, No. 3, March 1992, pp. 417-424.

[3] H. Shinohara et al., "A Flexible Multi-port RAM Compiler for Data-path," IEEE Journal of Solid-State Circuits, Vol. 26, No. 3, March 1991, pp. 343-348.

[4] A. G. Eldin, "A DRAM Compiler Algorithm for High Performance VLSI Embedded Memories," Fourth NASA VLSI Design Symposium, Coeur d'Alene, Idaho, Oct 1992.

Correctness Properties for Iterated Hardware Structures

Phillip J. Windley
Laboratory for Applied Logic, 3370 TMCB
Brigham Young University
Provo UT 84602
Phone: 801.378.3722, FAX: 801.378.7775
email: windley@cs.byu.edu

Abstract – **Iterated structures occur frequently in hardware. This paper describes properties required of mathematical relations that can be implemented iteratively and demonstrates the use of these properties on a generalized class of adders. This work provides a theoretical basis for the correct synthesis of iterated arithmetic structures.**

1 Introduction

Iterated structures occur frequently in hardware. Often these structures are used to implement arithmetic operations. While the intuition behind why such operations can be implemented through iteration is well understood, correctly synthesizing iterated structures requires a well-developed mathematical theory of why such structures are correct. To be generally useful, this theory should be as general as possible.

In an effort to demonstrate dependent types, Keith Hanna developed a theory of iterated hardware structures in[4]. While Hanna's work sets out the direction such generalized theories should take, his theories apply only to iterated cells with only one input and one output.

This paper presents an extension to Hanna's work by generalizing the theory for i inputs and j outputs. In addition, we have completed our work in the HOL theorem prover [3] which is based on simple type theory. The use of simple type theory for this work, rather than the more complicated, and undecidable, dependent type theory of [4] means that users of our theory incur fewer proof obligations regarding types. We demonstrate the use of the theory by developing a general theory of adders.

2 A Brief Introduction to HOL.

To ensure the accuracy of our specifications and proofs, we developed them using a mechanical verification system. The mechanical system performs syntax and type checking of the specifications and prevents the proofs from containing logical mistakes. The HOL system was selected for this project because is has higher-order logic, generic specifications and polymorphic type constructs. These features directly affect the expressibility of the specification language. Furthermore HOL is widely available, robust, and has a growing world-wide user base. However, there is nothing our work that requires the HOL theorem proving system.

HOL is a general theorem proving system developed at the University of Cambridge [1, 2] that is based on Church's theory of simple types, or higher-order logic. Although Church developed higher-order logic as a foundation for mathematics, it can be used for reasoning about computational systems of all kinds. Similar to predicate logic in allowing quantification over variables, higher-order logic also allows quantification over predicates and functions thus permitting more general systems to be described.

HOL is not a fully automated theorem prover but is more than simply a proof checker, serving as a proof assistant. HOL has several features that contribute to its use as a verification environment:

- Several built–in theories, including booleans, individuals, numbers, products, sums, lists, and trees. These theories build on the five axioms that form the basis of higher-order logic to derive a large number of theorems that follow from them.

- Rules of inference for higher-order logic. These rules contain not only the eight basic rules of inference from higher–order logic, but also a large body of *derived* inference rules that allow proofs to proceed using larger steps. The HOL system has rules that implement the standard introduction and elimination rules for Predicate Calculus as well as specialized rules for rewriting terms.

- A large collection of tactics to support goal directed proof. Included in HOL are tactics which rewrite a goal according to some previously proven theorem or definition, remove unnecessary universally quantified variables from the front of a goal, and split equalities into two implicative subgoals.

- A proof management system that keeps track of the state of an interactive proof session.

- A metalanguage, ML, for programming and extending the theorem prover. Using the metalanguage, tactics can be put together to form more powerful tactics, new tactics can be written, and theorems can be aggregated to form new theories for later use. The metalanguage makes the verification system extremely flexible.

For the most part, the notation of HOL is that of standard logic: $\forall$, $\exists$, $\wedge$, $\vee$, etc. have their usual meanings. There are a few constructs that deserve special attention due to their use in the remainder of the paper:

- HOL types are identified by a prefixed colon. Built–in types include `:bool` and `:num`. Function types are constructed using $\longrightarrow$. HOL is polymorphic; type variables are indicated by a type names beginning with an asterisk.

- The HOL conditional statement, written `a` $\rightarrow$ `b` | `c`, means "if `a`, then `b`, else `c`."

- Tuples are formed using a comma. Parentheses are only required when the scope of the comma is ambiguous.

- The HOL list containing elements `a`, `b`, `c`, and `d` is represented as `[a;b;c;d]`. A list that contains elements with type `x` has the type `:(x)list` where `x` can be any valid type (including type variables since HOL is polymorphic).

Table 1: Restricted types used in iterated hardware structures.

Name	Meaning
`pos`	positive natural numbers
`nat`	natural numbers
`N n`	natural numbers less than n
`bit`	bits (i.e. 0 or 1)
`bvec`	bit vectors
`V m`	vectors with elements less than m
`vec (P,l)`	vectors of length l with elements all meeting the predicate P

- The function `MAP f lst` applies the function `f` to each member of the list `lst` returning a new list made from the results of each application.

3 Numbers and Numerals

Iterated arithmetic hardware structures are based on the relationship between numbers and numerals. We expect that the behavior of a hardware device will be to perform an arithmetic operation on numbers and produce numbers as the result. We implement such structures by iterating devices which can be thought of as working on the separate digits[1] of a numeral representing the number. This is true whether we are iterating a 1–bit adder 32 times to produce a device that adds two numbers (with magnitude less than 2^{32}) or iterating an 8–bit carry lookahead adder 4 times to get the same result. In the latter case, one must consider the numeral to have four digits of magnitude less than 256

4 Type Restrictions

In the definitions and theorems that follow we will frequently use type restrictions to restrict the the value that a variable may take. These restrictions are expressed as predicates. Inside quantification, we will append these predicates to the variable that they modify with two colons. So, ∀ `v::P. B` is syntactic sugar for ∀ `v. P` $\Longrightarrow$`B`. One can, for most purposes, however, read the term ∀ `v::P. B` as "for all v of type P..." We will use the restricted types defined in Table 1 throughout the rest of this paper.

5 A Generalized Theory of Iteration

Figure 1 shows how iterated hardware structures can be defined recursively. In this case an $n+1$–bit structure is implemented by a 1–bit cell and an n–bit cell. Textually, we can represent this same process (in a more general way) as follows:

[1] In this paper, we will refer to the seperate components of a numeral in any base as digits, even though technically, they are only called this for numerals in base 10.

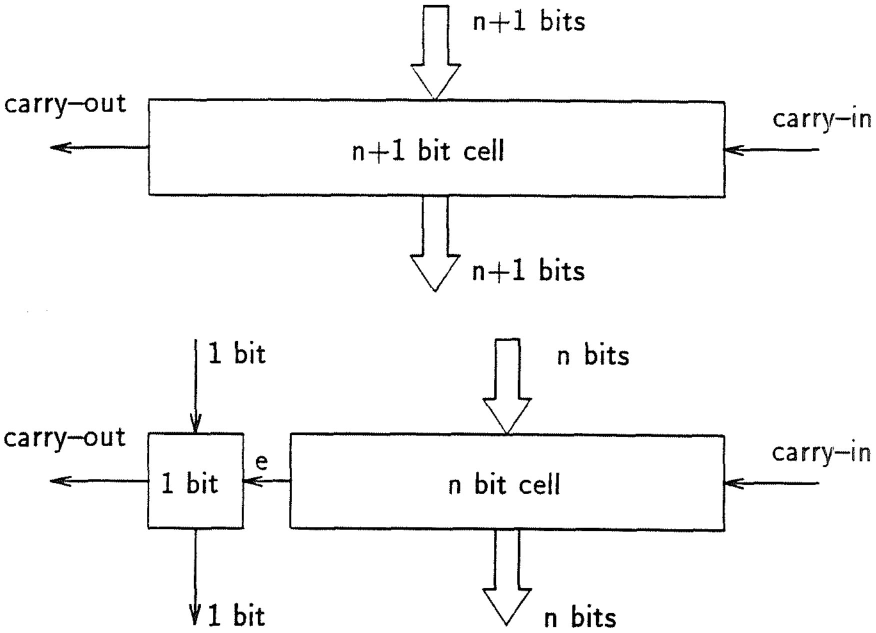


Figure 1: fold recursively implements iterated hardware structures.

```
(fold 0 cell in out cin cout = (cin = cout)) ∧
(fold (SUC n) cell in out cin cout =
    ∃ e . (fold n cell in out cin e) ∧
          (cell (MAP (λf.f n) in) (MAP (λf.f n) out) e cout))
```

Folding a cell 0 times results in a structure where the carry–in is wired directly to the carry–out. Folding a cell $n+1$ times is done by wiring a cell to a structure obtained by folding the cell n times. The existentially quantified variable, e, represents a hidden line connecting the carry–in of the most significant cell to the carry–out of the rest of the structure.

In order to make this definition as general as possible, in and out are lists of input and output busses respectively. We select the n^{th} line in the bus by applying the bus to n. Thus, (bus n) represents the n^{th} line in bus. So, the term (MAP (λ f. f n) in) selects the n^{th} member of each input in the list to be used by cell.

Using our definition of fold, we can define two properties that an arithmetic function (expressed in a relational form) must have in order to be iteratively implementable.

First, any arithmetic relation that we wish to implement iteratively must be *proper*.

```
⊢def proper R = ∀ a b (c:*) .
      (EVERY IS_ZERO a) ∧
      (EVERY IS_ZERO b) ⇒
      R 1 a b c c
```

The relation R takes a number giving the base it is to work in, 1 in this case; a list of inputs, a; a list of outputs, b; a carry–in, c; and a carry–out, also c in this instance. A relation is proper when it is true in base 1 with all its inputs and outputs are 0 and its carry–in is wired to its carry–out. This relationship, obviously, grows out of the base case in the recursive definition of `fold`. The proof that an arithmetic relation is proper is almost always trivial (see Section 5 for an example).

The second property that any arithmetic relation that we wish to implement iteratively must have is called factorizability. The following defines what it means for an arithmetic relation to be factorizable:

```
⊢def factorizable R =
      ∀ m k ::pos .
      ∀ l1 l2 ::pos .
      ∀ a0 :: vec(N m,l1) .
      ∀ b0 :: vec(N m,l2) .
      ∀ a1 :: vec(N k,l1) .
      ∀ b1 :: vec(N k,l2) .
      ∀ c d :* .
      let a = MAP (λ (x, y) . x + (y * m)) (a0, a1) and
          b = MAP (λ (x, y) . x + (y * m)) (b0, b1) in (
      (∃ e . (R k a1 b1 e d) ∧ (R m a0 b0 c e)) ⇒
      (R (k * m) a b c d))
```

The predicate is true of relations that can be implemented piecewise. That is, when a relation R on numbers less than $k \times m$ can be implemented by two instances of the same relation on numbers less than k and m respectively with a carry signal.

Using the factorization predicate, we can prove that a relation, R, that is both proper and factorizable can be iteratively implemented:

```
⊢ ∀ w .
  ∀ l1 l2 :: pos .
  ∀ m :: pos .
  ∀ (R:num -> (num)list -> (num)list -> * -> * -> bool) .
  ∀ A :: vec(V m,l1) .
  ∀ B :: vec(V m,l2) .
  ∀ c d .
  (factorizable R) ∧
  (proper R) ⇒
  ((fold w (R m) A B c d) ⇒
   (R (m EXP w) (MAP (VAL (m,w)) A) (MAP (VAL (m,w)) B) c d))
```

For a given relation R, this theorem states that when R is proper and factorizable, folding a base–m version of the relation w times implies the base–m^w version of the relation.

6 A Theory of Adders

In this section, we define an relationship for base–m addition, show that the relation is both proper and factorizable, and, using these results along with our theory, prove that an iterated adder has the correct behavior.

We can define a relation **add** as follows:

```
⊢def add m (w1, w2) w3 ci co =
          EVERY (N m) [w1;w2;w3] ⇒
          (w1 +  w2 + (bv ci) = w3 + (m * (bv co)))
```

In this definition, m is the base. w1, w2, and w3 are numbers less than m, and ci and co are boolean values representing the carry–in and carry–out. The function bv converts a boolean value to a number. Note that the relation is defined on base–*m* numbers, not numerals.

We have produced specialization of the general definitions and theorems given the last section for the most frequent cases. Using a specialization for 2 inputs and 1 output, we can show that add is proper. The proof comes down (after rewriting with the definitions) to showing that the following is true:

```
(0 + (0 + (bv c)) = 0 + (1 * (bv c)))
```

This proof is, of course, trivial. and establishes that

```
⊢ proper add
```

The second obligation, factorizability, while somewhat more complicated, is still surprisingly easy to prove. After expanding the definitions and some minor manipulations, we must show that

```
"(a0 + (a1*m)) + ((b0 + (b1*m)) + (bv c)) = (out0 + (out1*m)) + ((k*m)*(bv d))"
```

under the following two assumptions:

```
     [ "a0 + (b0 + (bv c)) = out0 + (m * (bv e))" ]
     [ "a1 + (b1 + (bv e)) = out1 + (k * (bv d))" ]
```

This is, of course, true by commutivity and associativity establishing that

```
⊢ factorizable add
```

The add relation is used to specify the correct behavior of an adder:

```
⊢def Adder (m,l) (A, B) OUT c d =
       add (m EXP l) (VAL(m,l) A, VAL(m,l) B) (VAL(m,l)OUT) c d
```

The definition operates on base–*m* numerals of length *l*. The function VAL is an abstraction function that converts numerals to numbers. Since we know that

```
⊢ (VAL (m,w) a) < (m EXP w)
```

The definition of Adder can be expanded to

```
⊢ Adder(m,l)(A,B)OUT c d =
    ((VAL(m,l)A) + ((VAL(m,l)B) + (bv c)) =
     (VAL(m,l)OUT) + ((m EXP l) * (bv d)))
```

The implementation of an adder is done iteratively:

```
⊢def Adder_impl (m,l) (A,B) OUT c d = (fold l (add m) (A,B) OUT c d)
```

The implementation iterates a base–m adder l times. Again, after expansion, this definition is like iterative specifications of adders done without our theory:

```
⊢ ∀m A B c d OUT.
   (Adder_impl(m,0)(A,B)OUT c d = (c = d)) ∧
   (Adder_impl(m,SUC l)(A,B)OUT c d =
       (∃e. fold_21 l(add m)(A,B)OUT c e ∧
            add m(A l,B l)(OUT l)e d))
```

Using our results about the factorizability of add and the theorems from the last section, we can show that the iterative structure correctly implements the desired behavior:

```
⊢ Adder_impl (m,l) (A,B) OUT c d ⇒  Adder (m,l) (A,B) OUT c d
```

The proof of this theorem is trivial and consists of expanding the definitions and applying a tactic based on the results of the last section.

This theorem is general enough to describe a large class of adders. This theorem applies equally whether we are implementing a base–2^{32} adder by iterating a 2–bit adder 32 times or iterating an 8–bit carry lookahead adder 4 times. This kind of generality is important in the synthesis of correct hardware structures since it gives the designer room to maneuver in the design space, trading off space and speed, while maintaining a correct design.

We have shown how this theory can be applied to adders, but it applies equally well to any arithmetic operation that is factorizable, including multiplication, comparison, etc..

7 Conclusion

This paper has presented a generalized theory of iterated hardware structures for implementing arithmetic operations. We have described properties that determine when an arithmetic operation *can* be correctly implemented by iteration. We have demonstrated the use of the theory by developing a theory of iterated adders that is general enough to be used to describe most adders in common usage. The theory forms the basis for correctly synthesizing adders based on constraints given by the designer.

References

[1] A. Camilleri, M. Gordon, and T. Melham. Hardware verification using higher order logic. In D. Borrione, editor, *From HDL Descriptions to Guaranteed Correct Circuit Designs.* Elsevier Scientific Publishers, 1987.

[2] M. J. Gordon. HOL: A proof generating system for higher-order logic. In G. Birtwhistle and P. Subrahmanyam, editors, *VLSI Specification, Verification, and Synthesis.* Kluwer Academic Press, 1988.

[3] M. J. C. Gordon and T. F. Melham. *Introduction to HOL.* Cambridge University Press, 1993.

[4] F. K. Hanna, N. Daeche, and M. Longley. Specification and verification using dependent types. *IEEE Transactions on Software Engineering*, SE-16(9), September 1990.

Partition Algebraic Design of Asynchronous Sequential Circuits

Gary K. Maki, Kristen Q. Chen and Suresh K. Gopalakrishnan
NASA Engineering Research Center
for VLSI System Design
University of New Mexico
Albuquerque, New Mexico 87106

Abstract - **Tracey's Theorem has long been recognized as essential in generating state assignments for asynchronous sequential circuits. This paper [1] shows that partitioning variables derived from Tracey's Theorem also has a significant impact in generating the design equations. Moreover, this theorem is important to the fundamental understanding of asynchronous sequential operation. The results of this work simplify asynchronous logic design. Moreover, detection of safe circuits is made easier.**

1 Introduction

Asynchronous VLSI design is starting to experience a renewed emphasis. The recent **VII Banff Workshop** was dedicated to Asynchronous Hardware Design, stressing both the applied and theoretical aspects of asynchronous design [1]. Besides the normal academics presenting new insights to the design of asynchronous VLSI, there was a set of major computing companies represented sharing latest design and chip implementation results. Steve Furber made this comment [2]:

There is a world-wide resurgence of interest in asynchronous logic design techniques. After two decades during which clocked logic has imposed its discipline across all corners of the world of digital logic, the older and more anarchic approach seems poised to made a come-back.

Virtually all digital design are based on the synchronous model and have controllers represented as synchronous sequential circuits. As Furber states [2], most asynchronous design has all but disappeared from sight. Moreover, many universities no longer teach many fundamentals associated with asynchronous sequential circuit design. However, global synchrony is becoming more and more difficult to maintain in high performance, high density chips implemented with new CMOS technologies.

As the integration density of VLSI circuits increase, synchronous circuits face clock distribution and power dissipation problems. Since an asynchronous circuit does not use a clock, the clock distribution overhead is eliminated. Further, the modules within an asynchronous circuit switch state only when required rather then at a potentially every clock edge, irrespective of whether a transition is needed or not. the instantaneous current and the power requirements are greatly reduced. In fact, asynchronous circuit technology is a major means to producing low powered VLSI [2, 3]. Moreover, asynchronous circuits usually produce the

[1] This research was supported in part by the NASA Space Engineering Research Center Program under grant NAGW-3293 Dr. Gopalakrishnan is with the California Design Center of Hewlett Packard

y_1	y_2	y_3		I_1	I_2	I_3
1	1	1	A	B	C	**A**
1	0	0	B	**B**	D	C
0	1	0	C	E	**C**	**C**
0	0	0	D	E	**D**	C
0	0	1	E	**E**	**E**	A

Table 1: Asynchronous Sequential Flow Table with Tracey Assignment

fastest possible circuits as they are not limited in speed by the clock [4]. These advantages present asynchronous controllers as a viable alternative. The major factor preventing the widespread use of asynchronous sequential circuits is that a more complex design procedure is required which must account for problems such as critical races. This paper presents material that allows for an easier method for the design of asynchronous sequential circuits, avoiding the use of normal state tables and K-maps.

Tracey introduced a fundamental theorem that define the necessary and sufficient conditions for realizing state assignments [5]. This work shows the relationship between Tracey's state assignment procedure and design equations, that can impact the process of generating design equations. A partition algebra can be defined that allows the design process to be simplier.

2 Asynchronous Sequential Circuit Fundamentals

It is assumed that the asynchronous sequential circuits operate in the fundamental mode [4] and are encoded with single transition time (STT) state assignments [5]. This implies that once an input change occurs the circuit is allowed to reach a stable condition before any further input change is allowed. Moreover, all state variables that must change during a transition are excited to change simultaneously at the beginning of the transition. STT assignments allow for the fastest possible state transitions. STT state assignment procedures have been advanced by Tracey, Liu and Tan [5, 6, 7].

In a column of a flow table, all the k-1 unstable states which lead to the corresponding stable state, together with the stable state form a *k-set.* For example in Table 1, the k-sets under input I_1 are AB and CED. The set of states that a circuit can assume during a transition between the states of a transition pair is called the transition path. Liu's and Tan's state assignments partition the *k-sets* under an input column [6, 7]. Tracey's state assignment procedure, on the other hand, partitions the transition pairs under a given input column [5].

Definition 1 *The two block partitions $\tau_1, \tau_2, \ldots \tau_n$, induced by the internal state variables $y_1, y_2, \ldots y_n$, respectively, are called the τ partitions of that assignment.*

For the state assignment given in Table 2, the τ partitions are as listed below.

$$\begin{aligned} \tau_1 &= \{CDF; ABE\} \\ \tau_2 &= \{BCE; ADF\} \\ \tau_3 &= \{EF; BCDA\} \end{aligned} \tag{1}$$

y_1	y_2	y_3		I_p
0	0	0	A	**A/1**
0	1	0	B	A
1	1	0	C	**C/0**
1	0	0	D	**D/1**
0	1	1	E	F
1	0	1	F	**F/0**

Table 2: Partial flow table

$\tau_1 = \{S_j \cdots ; S_k \cdots\}$ implies that the internal state Sj is coded '1' by the state variable y_1 and state S_k is coded '0'. Thus the first block of τ partitions are coded '1' by the corresponding internal state variables, and the second block of all τ partitions are coded '0' by the corresponding internal state variables.

Tracey's Theorem For STT operation in an asynchronous sequential circuit, all transition pairs of different *k-sets* under an input state must be partitioned by at least one state variable [5].

Tracey's theorem must be satisfied to eliminate critical races. Moreover, this theorem and the procedures used in this work [5] guarantees to produce all minimum variable state assignments. No other state assignment procedure can make such claims. Other assignment procedures may produce minimum variable assignments for some flow tables, but cannot guarantee to always produce the minimum variable assignment. Another feature of the Tracey assignment procedure is that it will generate all the STT assignments, not just the minimum variable assignments.

The design equations for an asynchronous sequential circuit can be formulated as follows:

$$\begin{array}{ccccccccccccc}
Y_1 & = & I_1(f_1^1) & + & I_2(f_1^2) & + & \cdots & + & I_p(f_1^p) & + & \cdots & + & I_m(f_1^m) \\
Y_2 & = & I_1(f_2^1) & + & I_2(f_2^2) & + & \cdots & + & I_p(f_2^p) & + & \cdots & + & I_m(f_2^m) \\
\vdots & & \vdots & & \vdots & & & & \vdots & & & & \vdots \\
Y_i & = & I_1(f_i^1) & + & I_2(f_i^2) & + & \cdots & + & I_p(f_i^p) & + & \cdots & + & I_m(f_i^m) \\
\vdots & & \vdots & & \vdots & & & & \vdots & & & & \vdots \\
Y_n & = & I_1(f_n^1) & + & I_2(f_n^2) & + & \cdots & + & I_p(f_n^p) & + & \cdots & + & I_m(f_n^m)
\end{array} \quad (2)$$

Definition 2 *A next state partition η_i^p partitions those states where $Y_i = 1$ from those states where $Y_i = 0$, under the input I_p.*

The η partitions corresponding to the partial flow table given in Table 2 are given below.

$$\begin{array}{ccl}
\eta_1^p & = & \{CDEF; AB\} \\
\eta_2^p & = & \{C; ABDEF\} \\
\eta_3^p & = & \{EF; ABCD\}
\end{array} \quad (3)$$

Definition 3 *An f_i^p is a sum of products expression of internal state variables that partition the states of η_i^p.*

The f_i^p expressions corresponding to the next state partitions of the partial flow table shown in Table 2 are given below.

$$\begin{aligned} f_1^p &= y_1'y_3' \\ f_2^p &= y_1y_2 \\ f_3^p &= y_3 \end{aligned} \tag{4}$$

Definition 4 *An internal state variable that partitions the states of one transition pair from the states of another is called a partitioning variable.*

It will be shown that the partitioning variables play an important role in the design equations.

3 Tracey Partitioning Variables

The relationship between the Tracey partitioning variables and the design equations has not been presented before. Consider the following development. Let S_i and S_j be states of a transition pair and S be the set of all states other than S_i and S_j.

Lemma 1

$$\begin{aligned} \textit{If} \quad \tau_1 &= S_iS_j;S_1 \\ \tau_2 &= S_iS_j;S_2 \\ &\cdot \\ &\cdot \\ &\cdot \\ \tau_k &= S_iS_j;S_k \end{aligned}$$

where $S_1 \cup S_2 \cup \ldots \cup S_k = S$, then $S_iS_j;S$ can be covered by the product term $y_1^*y_2^*...y_k^*$, where $y_i^* = y_i$ or y_i'.

Proof: If states S_iS_j are in the same block of each τ_i in the set of τ-partitions, then the intersection of partitions τ_i (or τ_i') will produce partition S_iS_j ; S. The product expression that covers S_iS_j ; S is $y_1^*y_2^*...y_k^*$.

Lemma 2

$$\begin{aligned} \textit{If} \quad \{\eta_1;S\} &\quad \textit{is covered by product term } P_1 \\ \{\eta_2;S\} &\quad \textit{is covered by product term } P_2 \\ &\quad \cdot \\ &\quad \cdot \\ &\quad \cdot \\ \{\eta_m;S\} &\quad \textit{is covered by product term } P_m \end{aligned}$$

then the partition $\{\eta_1\eta_2...\eta_m;S\}$ can be covered by a sum of products expression $P_1 + P_2 + ... + P_m$.

Proof: The proof follows directly from fundamental partition algebra.

Theorem 1 *The f_i^p terms of Eq. (1) are a function of the partitioning variables of input I_p.*

Proof: Consider a transition path T_0 where $Y_i = 1$ in T_0 and $Y_i = 0$ in the other transition paths T_i in I_p. The next state partition η_i^p can be expressed as T_0;S, where S contains all other transition pairs from different k-sets than T_0. The goal is to find an expression that partitions T_0 from S. From Tracey's Theorem [5] there exists a set of partitioning variables τ_i that partition T_0 from all other T_j. Therefore from Lemma 1, there exists an expression for Y_i covering T_0 that consists only of the partitioning variables.

In general, $Y_i = 1$ in several transition paths under a given input. A portion of the next state partition η_i for those transition paths where $Y_i = 1$ can be depicted as $\eta_i^p = T_k$; S. However, each transition path T_k is partitioned by partitioning variables and can be expressed as a product of the state variables that partition the transition path where $Y_i = 1$ from those where $Y_i = 0$. Therefore by Lemma 2, each f_i^p term can be expressed as a sum of products of the partitioning variables from I_p. **QED**

The following theorem yields a surprising basic result which shows that partitioning variables do not change state during circuit transitions.

Theorem 2 *The partitioning variables that partition a transition path T_0 from all other transition paths do not change state while the circuit proceeds from unstable to stable state of T_0.*

Proof: Since both unstable and stable states of a transition pair of T_0 are coded the same by the partitioning variables, these partitioning variables are not excited as the circuit transitions between the states of T_0. **QED**

To illustrate Theorem 1 and to show its impact on classical logic design, consider the flow table shown in Table 3. The state assignment partitions all the transition paths and is a Tracey assignment. The state variables $\{y_1, y_2\}$, $\{y_3, y_4, y_5\}$ and $\{y_4, y_5\}$ are the partitioning variables that partition the transition paths under I_1, I_2, and I_3 respectively. From Theorem 1, the expression representing each next state variable is a function only of the following partitioning variables under each input:

I_1	y_1, y_2
I_2	y_3, y_4, y_5
I_3	y_4, y_5

The state tables for each input need be constructed with only the partitioning variables associated with each input state. The significance of this example is that even though a 5 variable state assignment is needed in the traditional synthesis process, nothing larger than a 3 variable k-map need be constructed on a per input basis as shown in Fig. 1. The following design equations are obtained easily from Fig. 1.

I_1

y2 \ y1	0	1
0	GH 00000	CD 10010
1	EF 01100	AB 11111

I_3

y5 \ y4	0	1
0	EG 01100	BC 10010
1	DF 01001	AH 00011

I_2

y5 \ y3y4	00	01	11	10
0	G 01001	C 11110	B 11110	E 01001
1	F 10101	H 00011	A 10110	D 10101

Figure 1: State Tables for Each Input

$$\begin{array}{llllllllll}
Y_1 = & y_1I_1 & + & (y_4y_5' & + & y_3y_5 & + & y_4'y_5)I_2 & + & y_4y_5'I_3 \\
Y_2 = & y_2I_1 & + & y_5'I_2 & + & y_4'I_3 & & & & \\
Y_3 = & y_2I_1 & + & (y_4y_5' & + & y_3y_5 & + & y_4'y_5')I_2 & + & y_4'y_5'I_3 \\
Y_4 = & y_1I_1 & + & (y_4y_5' & + & y_3'y_4)I_2 & + & y_4I_3 & & \\
Y_5 = & y_1y_2I_1 & + & (y_4' & + & y_5)I_2 & + & y_5I_3 & &
\end{array} \tag{5}$$

One could easily come to the conclusion that satisfying Tracey's conditions are sufficient to realize STT state assignments. This notion is not always true as depicted from an example taken from [9]. Shown in Table 4 is the flow table and the design equations follow. The state assignment satisfies Tracey's theorem.

$y_1y_2y_3y_4y_5$		I_1	I_2	I_3
1 1 1 1 1	A	**A**	D	H
1 1 1 1 0	B	A	**B**	C
1 0 0 1 0	C	**C**	B	**C**
1 0 1 0 1	D	C	**D**	F
0 1 1 0 0	E	**E**	D	**E**
0 1 0 0 1	F	E	**F**	**F**
0 0 0 0 0	G	**G**	F	E
0 0 0 1 1	H	G	**H**	**H**

Table 3: Machine B

$(y_4\,y_5\,y_1\,\cdots y_3)$	I_0	I_1	I_2	I_3
(1 1 0 0 0) A	B	**A**	**A**	**A**
(1 0 0 0 1) B	**B**	A	–	C
(0 1 0 1 1) C	D	A	G	**C**
(1 0 0 1 0) D	**D**	A	–	A
(0 1 1 0 1) E	F	**E**	A	**E**
(0 0 1 1 1) F	**F**	G	–	E
(0 1 1 1 0) G	H	**G**	**G**	E
(1 0 1 0 0) H	**H**	E	–	E

Table 4: Flow Table with Modified State Assignment for Fail Safe Realization

$$
\begin{array}{llllllll}
Y_1 = & y_5y_2y_3I_3 & + & y_5y_1y_3I_3' & + & y_1y_2y_3 & & \\
 & & + & y_5y_1y_2 & + & y_4y_1 & & \\
Y_2 = & y_4y_3I_2 & + & y_5y_2y_3I_1' & + & y_4y_1I_0 & + & y_5y_1y_3I_0 \quad + \quad y_1y_2y_3I_2' \\
 & & + & y_5y_1y_2(I_1+I_3) & & & & \\
Y_3 = & y_4y_5I_0 & + & y_5y_3I_1' & + & y_5y_2y_3I_2 & + & y_5y_1y_3I_3' \quad + \quad y_1y_2y_3I_1' \\
 & & + & y_5y_1y_2I_2 & + & y_4y_1I_0' & & \\
Y_4 = & y_4y_5 & + & y_4y_3(I_0+I_1) & + & y_5y_2y_3(I_0+I_1) & + & y_4y_2 \quad + \quad y_5y_1y_3I_3 \\
 & & + & y_5y_1y_2I_0 & + & y_4y_1I_0 & & \\
Y_5 = & y_4y_5I_0' & + & y_4y_3I_0' & + & y_5y_3I_0' & + & y_4y_2I_0' \quad + \quad y_5y_1y_3I_0' \quad + \quad y_1y_2y_3I_0' \\
 & & + & y_5y_1y_2I_0' & + & y_4y_1I_0' & &
\end{array}
\tag{6}
$$

The partition variables for each input are:

I_1	y_1, y_2
I_2	y_3, y_4, y_5
I_3	y_4, y_5
I_4	y_4, y_5

Notice that the state variables used in the design equations are not a function of the partitioning variables. An analysis of the state tables shows that this design is replete with critical races simply because the design equations were improperly generated.

τ_1 = ABE;CDF
τ_2 = ABCF;DE
τ_3 = ADF;BCE
τ_4 = BCF;ADE

	I1	I2	I3
A	B	A/1	F
B	B/0	A	B/0
C	B	F	B
D	D/1	E	F
E	D	E/1	B
F	D	F/0	F/0

Table 5: τ-partitions for flow table

4 Design Equation Generation

The traditional method involves specifying the transition table (K-map) and then reading map for design equations. This often is cumbersome when there is a large number of state variables. All states in the transition paths must be properly specified. Following is the procedure for generating the design equations using partition theory which does not use K-maps or state tables.

Three theorems are used in the design of the next state equations [8].

Theorem 3 *If all the states of η_i^p are in one block, then f_i^p = 0 or 1.*

Theorem 4 *If η_i^p = tau_j, then f_i^p = y_j or $\overline{y_j}$.*

Theorem 5 *If η_i^p = η_j^p, then f_i^p = η_j^p or $\overline{\eta_j^p}$.*

4.1 Algorithm

1. Generate next state partitions
2. Using Theorems 3, 4, and 5 to identify as many f_i^p terms as possible.
3. Using partition algebra, generate the remaining f_i^p terms.

Example:

The π_i-partitions are:

π_1 = AB;DE π_2 = AB;DF π_3 = BC;DE π_4 = BC;DF π_5 = CF;AB
π_6 = CF;DE π_7 = AF;BC π_8 = AF;CE π_9 = DF;BC π_{10} = DF;CE

The table of next state partitions are shown in Table 6:

In column I1 all partitions are equal or complements. $\eta_1^2 = \overline{\eta_4^2} = Z^2$. Therefore the only η partitions that need to be determined are η_1^1 and η_1^2.

Transition pair partition T_{ip}^{wx} partitions transition pair wz, which is specified 1 for Y_i, from those states where $Y_i = 0$ under input I_p.

The basic idea is to find a logic expression which partitions the 1's of the transition path wz from the 0's of Y_i.

The transition pair partition for transition pair wz is a union of those π-partitions from I_p that:

	I1	I2	I3
η_1	ABC;DEF	ABDE;CF	ABCDEF;0
η_2	ABC;DEF	ABCF;DE	ABCDEF;0
η_3	DEF;ABC	ABCF;DE	ADF:BCE
η_4	ABC;DEF	CF;ABDE	ABCDEF;0
Z	DEF;ABC	ABDE;CF	ADF:BCE

Table 6: Next state partitions

1. partition wz and where

2. $Y_i = 1$ in wz and 0 in other transition pairs

Partition η_1^1 = ABC; DEF. The transition pair partitions T_{11}^{AB} = AB;DEF and T_{11}^{BD} = BC; DEF form η_1^1.

T_{11}^{AB} consists of π partitions π_1 = AB; DE and π_2 = AB; DF. π_1 is covered by τ_2 and π_2 is covered by τ_1. The product expression covering T_{11}^{AB} is y_1y_2. T_{11}^{BC} consists of π partitions π_3 = BC; DE and π_4 = BC; DF. π_3 is covered by τ_2 or τ_4, and π_4 is covered by $\overline{\tau_3}$. T_{11}^{BC} is covered by $y_2\overline{y_3}$ or $\overline{y_3}y_4$. The sum of products expression for f_1^1 is $y_1y_2 + y_2\overline{y_3}$ or $y_1y_2 + \overline{y_3}y_4$. $f_3^1 = \overline{f_1^1} = (\overline{y_1} + \overline{y_2})(\overline{y_2} + y_3) = \overline{y_2} + \overline{y_1}y_3$.

η_1^2 = ABDE; CF. The transition pair partitions are T_{12}^{AB} = AB;CF = π_5. T_{12}^{DE} = DE;CF = π_6. T_{12}^{AB} is covered by τ_1 and T_{12}^{DE} is covered by $\overline{\tau_2}$ or $\overline{\tau_4}$. $f_1^2 = (y_1 + \overline{y_2})$ or $(y_1 + \overline{y_4})$. $f_4^2 = \overline{f_1^2} = (y_2\overline{y_1})$ or $(y_4 + \overline{y_1})$.

The final next state equations are:

$$\begin{array}{rrrr}
Y_1 = & (y_1y_2 + y_2\overline{y_3})I_1+ & (y_1 + \overline{y_2}I_2+ & 1I_3 \\
Y_2 = & (y_1y_2 + y_2\overline{y_3})I_1+ & y_2I_2+ & 1I_3 \\
Y_3 = & (\overline{y_2} + y_3\overline{y_1})I_1+ & y_2I_2+ & y_3I_3 \\
Y_4 = & (y_1y_2 + y_2\overline{y_3})I_1+ & \overline{y_1}y_2I_2+ & 1I_3 \\
Z = & (\overline{y_2} + y_3\overline{y_1})I_1+ & (y_1 + \overline{y_2}I_2+ & y_3I_3
\end{array}$$

5 Safe Sequential Circuits

The above results can be applied to analyze the safeness of sequential circuits. Conventional asynchronous sequential circuit design procedures generate circuits that implement the flow table as per specification; under normal conditions of operation it is assumed that the circuit assumes only those states shown in the flow table. However, under abnormal conditions, like power fluctuations or noise spikes, the circuit can enter unspecified states. This condition may go undetected and may even be impossible to correct without the application of a master reset. Further, in some circuits there may be no reset signal available for resetting the circuit to a known stable state. Therefore, it is important that the designer be aware of the existence of such unspecified stable states in producing a safe design. A safe circuit is essentially a circuit in which it is always possible to assume specified stable states and the circuit is never allowed to remain indefinitely in unspecified states.

Unspecified stable states are formed when the design equations unintentionally specify the state as stable. The first study on safe circuit design was presented by Wickersham and

Maki [8]. In this paper, a procedure for safe circuit detection was introduced. They also introduce a safe design procedure which may require more than the minimum number of state variables.

5.1 Safeness Analysis

Definition 5 *Two internal states of a sequential circuit are in the same equivalence class under I_p, if they have the same next state entry.*

Definition 6 *A lock-up state under an input I_p, is any stable state under I_p that is not a member of a transition path under that input.*

Definition 7 *A lock-up state and the states of the flow table that have a lock-up state as a next state entry form a lock-up equivalence class.*

When a circuit assumes a lock-up state, there are several ways to attempt force the circuit back to the specified states of the flow table. One of the schemes is to apply an input or a sequence of inputs so that the circuit transitions through a set of unspecified stable states until it finally reaches a specified stable state. However, in some cases this may not be possible. Another approach is to turn the power off and on and hope that the circuit would come up in a specified stable state. Once a specified state is reached, further input changes keep the circuit in its specified transition paths, until an abnormal condition forces it out of the specified transition path.

Definition 8 *A critical situation exists if there are states from which it is impossible to re-enter a specified transition path by simply changing inputs.*

Definition 9 *A circuit is* safe *if a critical situation does not exist.*

This implies that a safe circuit can be designed by making sure that there is at least one input column under which there are no lock-up states. If the circuit ever enters a lock-up state, the input with no lock-up states can be applied to force the circuit in to one of the transition paths of the circuit. However, for some situations it may be desired to ensure that all the inputs are lock-up free. The safeness analysis procedure is a modified version of the procedure outline by Wickersham and Maki [8], the difference being that only the partitioning variables under each input are used for the analysis instead of all the state variables being used. Procedure 1 is based on the following theorems.

Theorem 6 *Equivalence classes under an input state can be represented as functions of the partitioning variables under that input state.*

Proof: Equivalence classes are specified by the next state entries under an input. From Theorem 1, the next state equations are a function only of the partitioning variables. Therefore equivalence classes are a function only of the partitioning variables. QED.

Theorem 7 *To identify the stable states under an input state, only the present state and next state value of the partitioning variables under that input need be checked.*

Proof: For any input state, product expressions of the partitioning variables under that input state uniquely represent the stable state and the associated equivalence class. If the circuit is in one of the stable equivalence classes then it assumes one of these unique product expressions. To check whether a circuit is in a stable state under an input state, only the present state and next state values of the partitioning variables under that input state need be checked. QED.

Procedure 1 *New safeness analysis procedure.*

Step 1: Map the design equations on to a K-map. The K-map for each input need contain only the partitioning variables. A stable state occurs whenever present and next state are equal. Only the f_i^p terms for the partitioning variables under that particular input need be considered in this step. This can be repeated for all the inputs.

Step 2: Identify all the lock-up states. Lock-states are those stable states that are not specified in the flow table.

Step 3: If lock-up states are found then determine if a critical situation exists. This can be accomplished by determining which equivalence class the circuit enters under other inputs from the lock-up equivalence class. If there exists a lock-up equivalence class which transitions only to other lock-up equivalence classes and specified equivalence classes cannot be entered, then a critical situation exists. If a critical situation exists then the circuit is not safe.

y_1	y_2	y_3	y_4	y_5	y_6		I_1	I_2	I_3
1	0	1	0	0	1	A	**A**	B	F
1	0	1	0	1	0	B	A	**B**	C
1	0	0	1	1	0	C	A	E	**C**
0	1	0	1	1	0	D	**D**	E	C
0	1	0	1	0	1	E	D	**E**	F
0	1	1	0	0	1	F	D	B	**F**

Table 7: Flow table for Safe Design

Example 1 *Consider the flow table shown in Table 2, the state assignment used is the Tan assignment.*

The design equations for implementing this circuit is given in Equation 7.

$$\begin{array}{lcccccc}
Y_1 & = & I_1(y_1) & + & I_2(y_3) & + & I_3(y_5) \\
Y_2 & = & I_1(y_2) & + & I_2(y_4) & + & I_3(y_6) \\
Y_3 & = & I_1(y_1) & + & I_2(y_3) & + & I_3(y_6) \\
Y_4 & = & I_1(y_2) & + & I_2(y_4) & + & I_3(y_5) \\
Y_5 & = & I_1(y_2) & + & I_2(y_3) & + & I_3(y_5) \\
Y_6 & = & I_1(y_1) & + & I_2(y_4) & + & I_3(y_6)
\end{array} \tag{7}$$

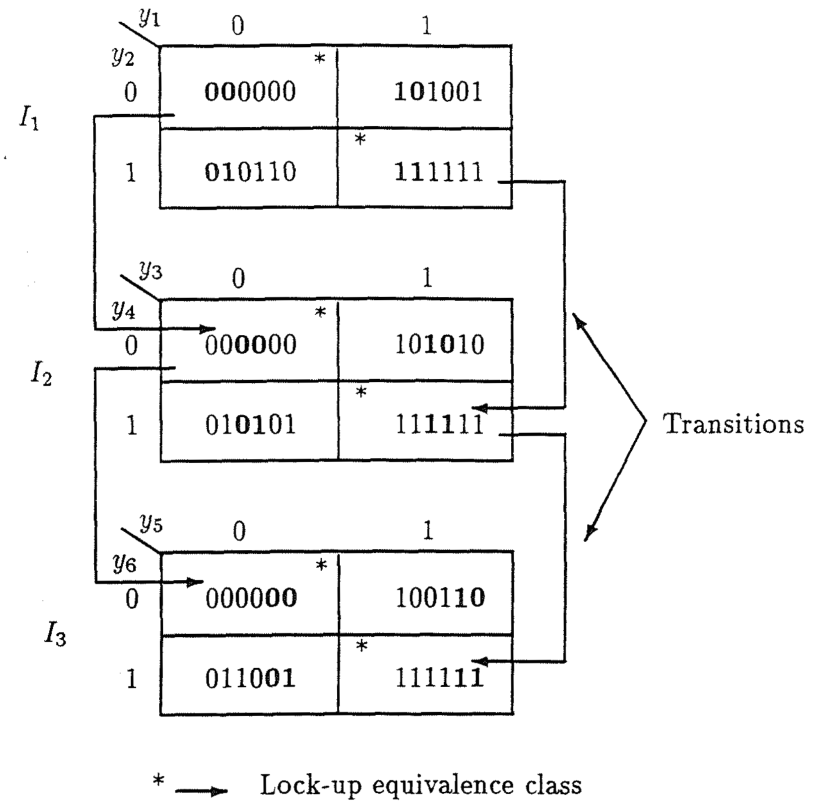


Figure 2: Safeness analysis

The partitioning variables under input I_1 are y_1, y_2, input I_2 are y_3, y_4 and input I_3 are y_5, y_6. The safeness analysis for this implementation is shown in Fig. 2. It can be seen that there are two lock-up states per input namely, 000000 and 111111. It can also seen that these lock-up states lead to a critical situation since once any of these states is assumed the circuit cannot re-enter a proper transition path by change of inputs.

6 Summary

The relationship between Tracey's fundamental state assignment theorem and the resulting design equations has been shown. The state variables that are partitioning variables to prevent critical races must appear in the design equations to insure STT operation. Classical design can be accomplished easier as a result of this work because k-maps with fewer state variables are required in representing the state tables. Showing state tables with only the partitioning variables allows easy analysis of asynchronous sequential circuits. It is also possible to derive the complete state equations without using any K-maps or state tables.

References

[1] Proceedings of the VII Banff Workshop: Asynchronous Hardware Design, Banff Canada, August, 1993.

[2] S. Furber, "The Return of Asynchronous Logic", Proceedings of the VII Banff Workshop: Asynchronous Hardware Design, Banff Canada, August, 1993.

[3] B. Coates, A. Davis, and K. Stevens, "Automatic Synthesis of Fast Compact Self-Timed Control Circuits", pp. 193-208, Proceedings of the IFIP Working Conference on Design Methodologies, Manchester, April 1993.

[4] S. Unger, *Asynchronous Sequential Switching Circuits*, New York, NY, Wiley-Interscience, 1969.

[5] J. Tracey, "Internal State Assignments for Asynchronous Sequential Machines", IEEE Transactions on Electronic Computers, Vol. EC-15, pp. 551-560, Aug. 1966.

[6] C. Liu, "A State Variable Assignment Method for Asynchronous Sequential Switching Circuits," J. ACM, Vol 10, pp. 209-216, Apr. 1963.

[7] C. Tan, "State Assignments for Asynchronous Sequential Machines", IEEE Transactions on Computers, Vol. C-20, No. 4, pp. 382-391, April 1971.

[8] R. Wickersham and G. Maki, "Safe Asynchronous Sequential Circuits", IEEE Transactions on Computers, Vol. C-23, No. 5. May 1974.

[9] S. Das and Y. H. Chuang, "A Unified Approach to the Realization of Fail-Safe Sequential Machines," IEEE Fault Tolerant Computing Symposium, pp. 3-2, 3-6, June, 1974.

Automated Radiation Hard ASIC Design Tool

Mike White, Bill Bartholet and Mark Baze
Boeing Defense and Space Group
Seattle, WA
(206)-657-6223

Abstract - **A commercial based, foundry independent, compiler design tool (ChipCrafter) with custom radiation hardened library cells is described. A unique analysis approach allows low hardness risk for Application Specific ICs (ASICs). Accomplishments, radiation test results, and applications are described.**

1 Introduction

Application Specific ICs (ASICs) are increasingly recognized by system designers as an option for reducing size, weight, power, and cost while increasing reliability. Their use however has been slower to appear in radiation hardened programs due to the circuit dependent aspects of hardness variation. Since hardness assurance testing can only occur after the substantial investment of fabrication, verification tools that can facilitate first pass success in radiation performance are crucial.

This paper describes the advantage of a foundry independent, commercial market driven, compiler tool approach to radiation hard ASIC design. Library development techniques to achieve hardness are described. The hardness verification methods for new ASIC designs are discussed. including the analysis and test structure support approaches. Finally, our actual uses of this tool in support of Boeing programs, and continued development plans are included.

2 Compiler

Boeing's radiation hardened design work is based on the ChipCrafter compiler from Cascade Design Automation. The use of a commercial compiler tool for radiation hard design allows a leveraging of commercial driven development and maintenance for radiation program use. The compiler is an integrated design environment that features a parameterized library of process independent function blocks addressing a range of capability across a broad selection of processes. Foundry independence provides process selection freedom, competitive bidding, and easy 2nd source options. In addition, prototype development in cheaper soft processes can reduce cost, with fabrication in a hardened process reserved for final production. The feature of process independent design methodology allows recompilation of a design to any of the hosted processes. Final process selection can be deferred, or changed through a simple procedure that automatically regenerates the entire design. This process independence allows easy design migration between processes and evolving technologies. The hosted processes accurately capture physical, electrical, and performance characteristics through the use of retargetable module generators.

The compiler approach provides the capability to support not only the design of radiation hardened ASICs but also provides tool support for the design of the cell libraries used in the ASIC design. Physical layout of the cells is characterized by over 150 design rule variables that customize the layout for the target process. The rules include everything from well type and number of routing layers available to inter-layer spacing and overlaps. The compiler applies the rules to the module generators (procedural level transistor layout descriptions) to generate the process specific layout geometry for each library cell at the time of compilation. The layout topology varies from rule set to rule set to optimally accommodate the demands of the specific process and user defined parameters, such as radiation requirements.

The process rules are used to optimize the physical and electrical performance of the modules. Process parameters are used to optimally size devices and characterize electrical performance in areas such as current density, resistance and capacitance. These values are used, in turn, to drive the physical layout of the design (e.g., power rail widths and buffer sizes) and to derive delay models. Transistor sizing parameters are created from process feature sizes and SPICE simulation models. Each rule set is individually optimized as opposed to using a fixed device width approach.

ChipCrafter includes the capabilities required for radiation hardened VLSI designs. Dose rate and SEU tunability have been incorporated into the robust library of process independent, parameterized modules including standard cells, data path cells, extensive set of I/O pads, MSI building blocks, and memory. Included in the enhancements are electromigration driven power, signal and clock routing width, clock skew minimization, performance biased layout, and probabilistic power estimation.

3 Rad Hard Library Development

The procedural level transistor descriptions for the cell library modules provide the capability to compile the design for a variety of foundry processes. This retargetability is accomplished by using the geometric rules as parameters for determining spacing during cell layout. The compiler also uses user parameters such as total dose, dose rate or SEU requirements during the device sizing portion of cell generation.

The module compilers take high level input and generate layout geometry and simulation models. Design of module compilers follows a top-down methodology that is shown in Figure 1. The process includes

1. module specification

2. gate-level representation

3. device level representation

4. layout geometry with landmarks

5. transistor level simulation model.

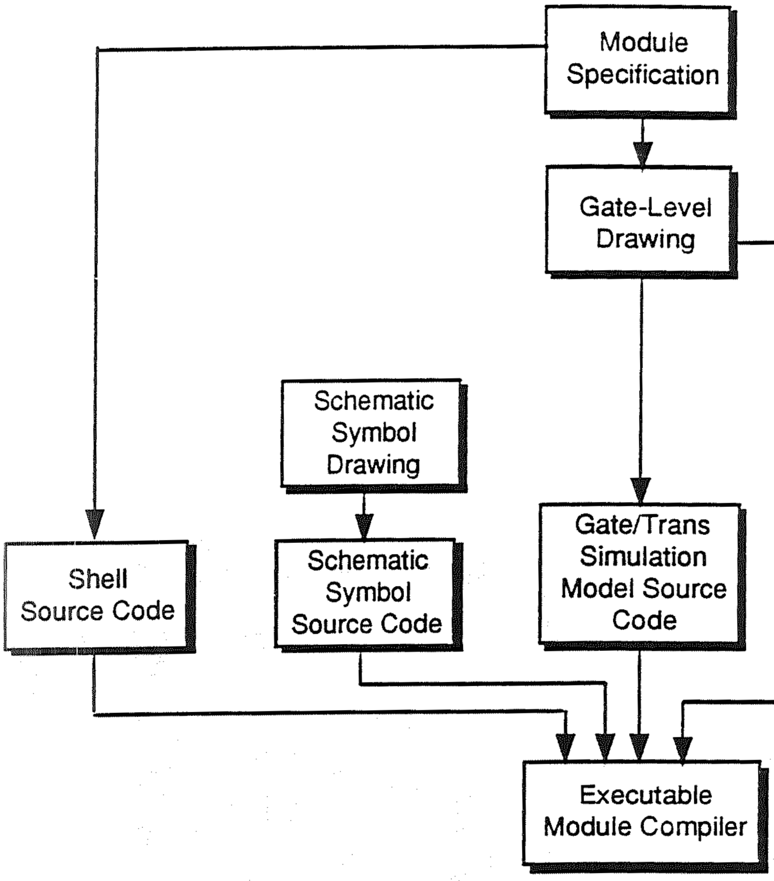


Figure 1: Module Compiler Design Process

diagram. approximate dimensions, and end-user input options.

Gate-Level Drawing

Gate-Level Drawing is the explicit logic of the lowest level cell(s) and also the preliminary representation of the gate-level simulation model.

Device Level Drawing

Device-level drawing is the gate-level drawing expanded to the level of discrete devices.

Layout Geometry

Layout Geometry is the device-level drawing translated into physical geometry. The structures are positioned with a virtual grid of "landmarks". Landmarks are x-y coordinates whose positions are not absolute but relative and dependent upon a design rule set. The layout drawing will reflect technology, connection requirements, and dimensions. Figure 2 shows a layout drawing with coordinate grid. This landmark technique serves two purposes. First it provides design-rule independent reference points. Second, it allows for easy stretching of a cell for transistor sizing or pitchmatching purposes or to accommodate options.

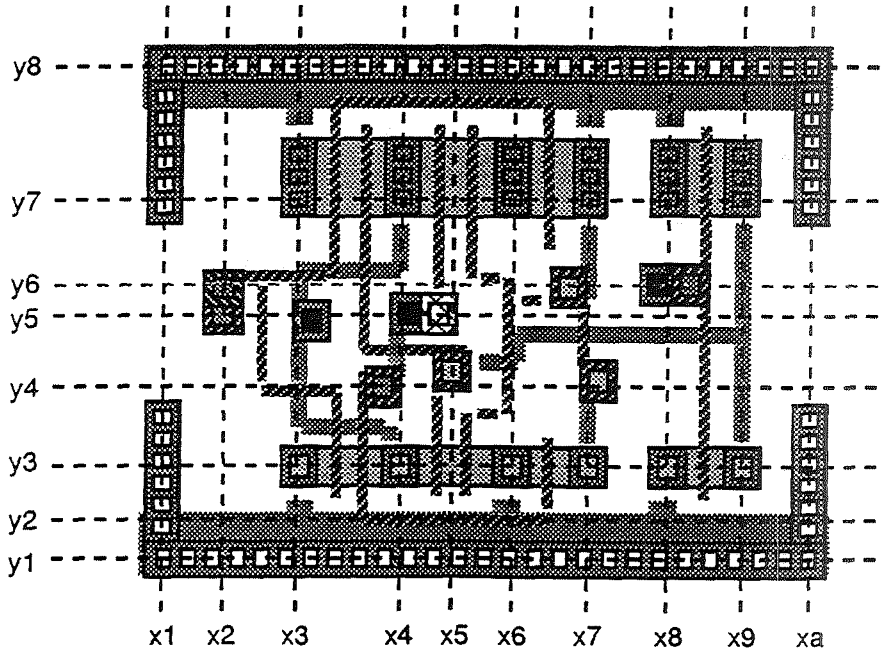


Figure 2: Layout Coordinate Grid

All dimensions are expressed in the source code as design rule variables and do not take on real values until the compiled code is run in a particular rule set to generate a cell. The "virtual grid" coordinate technique provides for geometry spacing based on design rule variables. The landmarks are free to move and are referenced outside the cell only to the x,y origin.

In the example, landmarks are drawn through the centers of the contacts because contacts have a certain intuitive appeal as reference points and because most design rule variable macros define distances from centers of contacts or the center of wires with contacts on them. Only xl and yl have absolute values (in this case xorigin and yorigin). All other x landmarks are defined, directly or indirectly, in relation to xl and all other y landmarks are defined, directly or indirectly, in relation to yl. This technique ensures that if the distance between two landmarks needs to be changed or offset, all landmarks above or to the left of it, as well as all objects attached to those landmarks, will move.

Transistor Level Simulation Model

Transistor Level Simulation Model is highly analogous to layout (geometry) source code. The hierarchy of the transistor simulation source code follows layout hierarchy as closely as possible, defining first the lowest level cells (these consist of calls to transistor primitives), then instantiating those cells to form the complete module.

4 Radiation Hardening

The first hardening variable with a multiple foundry design tool is the selection of the target process. The varying CMOS hardness choices of SOI, epi-layer protected bulk, or simple bulk processes span a wide hardness range. In addition, however, the details of transistor geometry further effect hardness. For rad hard design, the detailed definition of physical layout geometry of the commercial compiler design tool has been modified for each fundamental library cell to achieve significantly enhanced hardness for both Single Event Upset (SEU) and Dose Rate environments. Both the hardening design and the hardness assurance efforts are considered at both the individual gate geometry level and the overall circuit topology level. Degradation from total dose and neutron environments are predominantly process rather than geometry dependent, and therefore not strong risks for each new ASIC design in a given process.

Gate-level hardness

SEU hardening is considered for both direct hit of static bits, as well as direct hit of combinatorial logic. Logic hits propagate erroneous transients into static bits, where a stable upset can occur. In general, the time duration of SEU disturbances can be shortened by increasing the drive strength of transistors. A higher drive transistor more quickly drives the spurious SEU charge away. In addition, the increased gate capacitance of increased drive transistors also reduces the voltage excursion of SEU deposited charge in a circuit. Increasing transistor drive strength then, increasing the SEU charge required to exceed switching voltages for the required setup and hold time to cause an upset. This results in a harder design. Further, a deliberate time delay in a memory loop, with some RC element, can further increase setup and hold time yielding still more SEU hardness.

Dose Rate hardening requires control of transient voltage gradients from photocurrent flow within individual wells containing the logic gate transistors. This was achieved by both increasing the number and density of well contacts and re-arranging the transistor layout within the wells. The well contact strategy minimizes photocurrent voltage drops by terminating electric fields and providing robust current sinks. The transistor arrangement strategy is to avoid photocurrent bottle necks or pile up of IR voltage drops.

Circuit-level hardness For SEU. the circuit-level topology of gates also affects the ultimate upset rate. Ion hits in combinatorial logic propagate transients through logic potentially into and upsetting static bits. The exact propagation behavior will depend on the specific logic design. In the ASIC design environment, the logic flow is available for CAD analysis, and such an analysis tool for more SEU predictive accuracy is under development at Boeing.

Currently, without such detailed logic analysis, other simpler approaches have been very successful in at least bounding the worst case upset rate. One potential upper bound on the upset rate can be the minimum charge (or ion LET threshold) to propagate through a single gate. A worst case logic node is assumed as one connected to the minimum input capacitance for a maximum voltage transient. driven with the minimum drive gate for the longest transient duration. A minimum drive inverter string is representative of this worst case combination. This has been both SPICE modeled and directly SEU tested.

Another potential upper bound of SEU rate from logic transients can be the minimum transient pulse width that satisfies setup and hold times to successfully (erroneously) switch the destination bit (in fully synchronous designs). The worst case (minimum setup and hold time) of the only small number of static bit designs in the design library has been identified, modeled, and SEU tested.

In addition, design practice can have a significant impact on upset rate. For example, the logic upset behavior is both reduced and simplified if the chip clock signals are made SEU immune. Our designs achieve this by requiring only global clocks with high drive buffers and high capacitance extended buses. This means gated clocks within the chip are not allowed. Further, transient abatement is maximized by further limiting design practice to only fully synchronous architecture. This means all valid data progresses from one static bit to another in each clock cycle. Only transients coincident with clock edges can potentially cause upsets.

For Dose Rate, the main circuit-level issue is the resistive and inductive voltage drop across the power rails and bond wires caused by the dose rate photocurrent surge. The design tool automatically generates, sizes, and routes the power distribution network. Power routing takes into account the effects of dose rate, AC switching currents, and electromigration in sizing the power network. Each module is extracted to obtain the photocurrent collection volumes. The resulting photocurrent is added to the capacitive switching current and used in the power network generation and sizing. A multiple-tree/mesh power routing algorithm minimizes the area of the power rails by constructing a power tree for each source pad on the chip. The trees are generated by adding the internal power nodes of the core to the tree of the nearest source recursively until all nodes are included. Each tree is automatically sized to minimize area while meeting voltage drop and electromigration constraints for an input dose rate hardness requirement. Inter-tree connections occur through standard cell groups, which are internally meshed. For switching current calculations, individual switching frequencies may be specified at the block level in addition to the top level.

5 Modeling

In modeling the SEU direct hit of a static bit. different nodes, and different logic states all contribute different cross-sections with different critical charges for upset. SPICE modeling of SEU simulated injected charge is used to determine the critical charge of all sensitive bit nodes over all logic states for a complete upset rate calculation.

Similarly for SEU logic transients, SPICE modeling of SEU simulated injected charge is used to determine the critical charge for minimum propagation through a single worst case logic gate. For determination of the minimum setup and hold time for transients input to static bits, only logic pulse widths are required for modeling, since these transients arrive as fully driven square pulses through upstream logic.

For Dose Rate voltage gradients in well regions around transistors, a photocurrent production model is used to drive distributed current sources in a lumped 2-D resistive grid as a SPICE model.

6 Tunable Hardness & Test Results

One extremely useful advantage of the compiler approach is the application of hardness adjustable layout choices in compiled designs. Variable SEU hardness has been achieved with selection of variable transistor drive strengths and/or the addition of RC elements. Table 1 shows the CAD tool selectable, LET threshold adjustability for a basic latch in an epi-layer bulk CMOS process.

Latch Type	**Minimum Xstr drive**	**Maximum Xstr drive**
NAND	20	37
Master/Slave	30	60
Master/Slave w/RC	70	130

Table 1: Basic Latch LET threshold (MeV/(mg/cm2)) for various geometry selections in epi-CMOS

Direct radiation testing has also verified the adjustability of hardness by geometry selection at time of design compile. A similar hardness spread for an SOI implemented latch expressed as upset rate in GEO orbit, ranges from 1E−8 to <1E−12 upsets/bit-day. In dose rate, the SOI implemented selectable hardness range spans 1E8 to >2E11 rad/sec from softest to hardest geometries.

An example of the compiler's advantage in hardness flexibility was demonstrated with a recent microcontroller delivered for a LEO orbit program. Existing soft design investment was upgraded to a hardened microcontroller by re-compiling the design in the rad hard compiler, switching to a hard process, selecting SEU hardening geometry, but turning off un-needed dose rate hardness. Several foundry/process options allowed competitive bidding. A mid-range SEU hardening choice was sufficiently hard and allowed maximum speed by avoiding the RC slowed hardest option. The ability to forego dose rate modifications kept area growth to a minimum. SEU hardness of the final product on CMOS epi-layer bulk exceeds 20,000 years between upsets in its LEO environment, exceeding hardness (and speed) requirements. Figure 3 shows the analysis predicted cross-section vs. LET (deliberately conservative) very favorably agreeing with final test results (1 of 5 samples anomalously displayed a lower LET threshold).

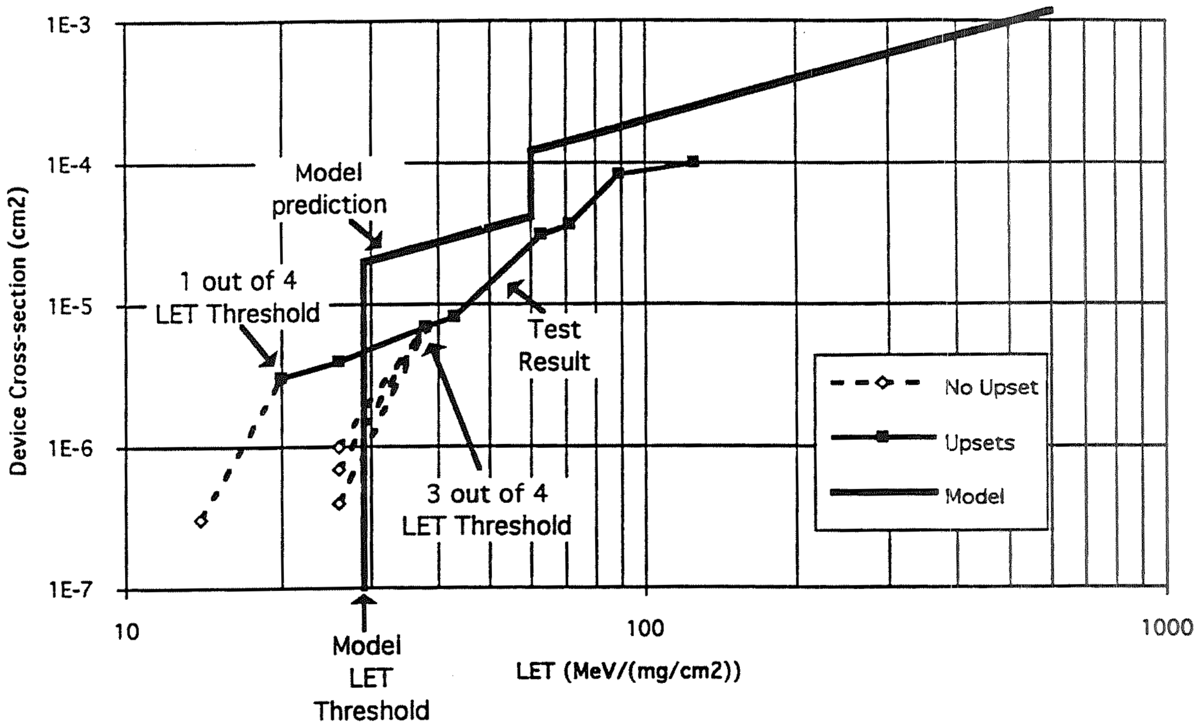


Figure 3: Microcontroller SEU Radiation Test Result

7 Applications

At Boeing, the rad hard compiler has already supported several program ASICs including a tactical 1553, a tactical memory management unit, a space hard 8051, and a space hard 1773. Associated modeling and analysis tools verified by rad testing gives low hardness risk for new ASIC designs.

8 Conclusion

The radiation hardened compiler has proven to be a very successful design tool. Over one hundred library cells have been modified as appropriate for SEU & Dose Rate hardness. Further, these modifications are selectable for different hardness levels. Two different foundries, with a third expected by conference time, are accessible with the rad hard libraries covering strategic (CMOS SOI) to tactical/space (epi-CMOS). Testing has already verified SEU hardness of the libraries at both cell and VLSI levels. Dose Rate hardness has already been verified at cell level with VLSI testing planned 1Q94.

A Unified Approach to VLSI Layout Automation and Algorithm Mapping on Processor Arrays

N.Venkateswaran, S.Pattabiraman, Vinoo N.Srinivasan
N. Vijaykrishnani and S.Balamurugan
Dept. of Computer Science and Engineering
Sri Venkateswara College of Engineering
Pennalur, Sriperumbudur - 602 105
Tamilnadu, INDIA

Abstract - **Development of software tools for designing supercomputing systems is highly complex and cost ineffective. To tackle this a special purpose PAcube [1] silicon compiler which integrates different design levels from cell to processor arrays has been proposed in [2]. As a part of this, we present in this paper a novel methodology which unifies the problems of Layout Automation and Algorithm Mapping.**

Index Terms - Layout design/Algorithm Mapping, Placement/Assignment, Modules/ Partitions, Wafer area/Processor array, gate arrays/SIMD array (Single Instruction Multiple Data), general cells/MIMD array (Multiple Instruction Multiple Data), Routing, routing channel/switch lattice, process synchronization, GIPOP - Generalized Inner Product Outer Product, IP - Inner Product, CM - Chain Multiplication.

1 Introduction

VLSI Layout Automation involves placement of hierarchially organized functional modules of the system design onto the wafer and interconnecting them according to the net-list specification. An efficient automation algorithm aims at minimizing the chip area and overall interconnection cost. Advances in automation procedures has brought down drastically the cost involved in custom and semi-custom VLSI system design. A detailed review of VLSI layout schemes is found in [3] [4] [5].

Algorithm Mapping is the process of embedding the computation and communication structure of an algorithm onto the processor array. An efficient mapping strategy minimizes the communication complexity and maximizes the processor utilization. A detailed review of Algorithm Mapping strategies is found in [6] [7].

Conventionally, the problems of layout automation and algorithm mapping are treated as two extremities in the design hierarchy of an supercomputing system. Interestingly, our analysis reveals the existence of many functionally identical steps in these problems. Based on this we have developed a unified approach to tackle these problems, reducing the complexity of the special purpose PAcube silicon compiler.

In Section I and II the problems of Layout Automation and Algorithm Mapping on Processor Arrays are discussed in detail. The concept of unification is presented in section III. Section IV describes the unified algorithm, with examples for layout automation and algorithm mapping using the unified approach.

2 Layout Design Automation

Layout Automation can be defined as the process of embedding the functional modules and their interconnect structure of the system topology onto the wafer. Formally, if G = ¡N,E¿ represents the module level interconnect structure (netlist) of the system and T = ¡n,e¿ represents the grid structured image of the wafer, then the F(G) → T is defined as the process of layout automation.

Layout Automation is applied to both full-custom and semicustom based design of VLSI chips. In semi-custom design (eg. gate arrays), the wafer has a structured image with fixed areas assigned for gates/standard cells, wires (interconnection routes) and I/O pads. Here, the modules are designed to by interconnecting set of gates of the array. In full-custom design, the wafer is an empty piece of silicon. The full-custom approach permits the wiring area to expand or contract providing more flexible utilization of chip real estate. Here, the modules (called General Cells) are hand optimized layouts. In both these approaches the modules are in general rectangles of different aspect ratios.

The steps involved in Layout Automation (Figure 1) are

1. Modular partitioning of VLSI system.
2. Intra module computational decomposition.
3. Design of different sub-modules that form the functional module.
4. Topological Placement (relative positioning) of functional modules on the wafer and Routing.
5. Process (module level) synchronization.
6. Programming the VLSI system controller.

Steps 1, 2 and 3 form the pre-layout process. System partitioning involves hierarchical breaking the system topology into different functional modules (F1,F2,F3,F4 in Figure 2.b). The functionality of the modules are then decomposed into computations to be performed by the sub-module. The next step is to design the sub-modules so as to balance the computational complexity across different functional modules. The balancing is achieved by fixing the degree of parallelism in each module. This helps in synchronizing the module level operations. Four and five form the major steps in the layout automation process.

Placement involves usually a constructive placement followed by the iterative improvement. Constructive placement provides a foundation or an initial placement of the modules which may be suitably interchanged by the stages to follow. The Iterative improvement refines the constructive placement by placing the densely interconnected modules in the neighborhood of each other. An efficient constructive placement and iterative improvement algorithm requires proper ordering of net- list and the modules sizes. The net-list ordering in particular helps in minimizing the layout area(full-custom approach)/unutilized gates(gate arrays) with an overall perspective of reducing the connection cost.

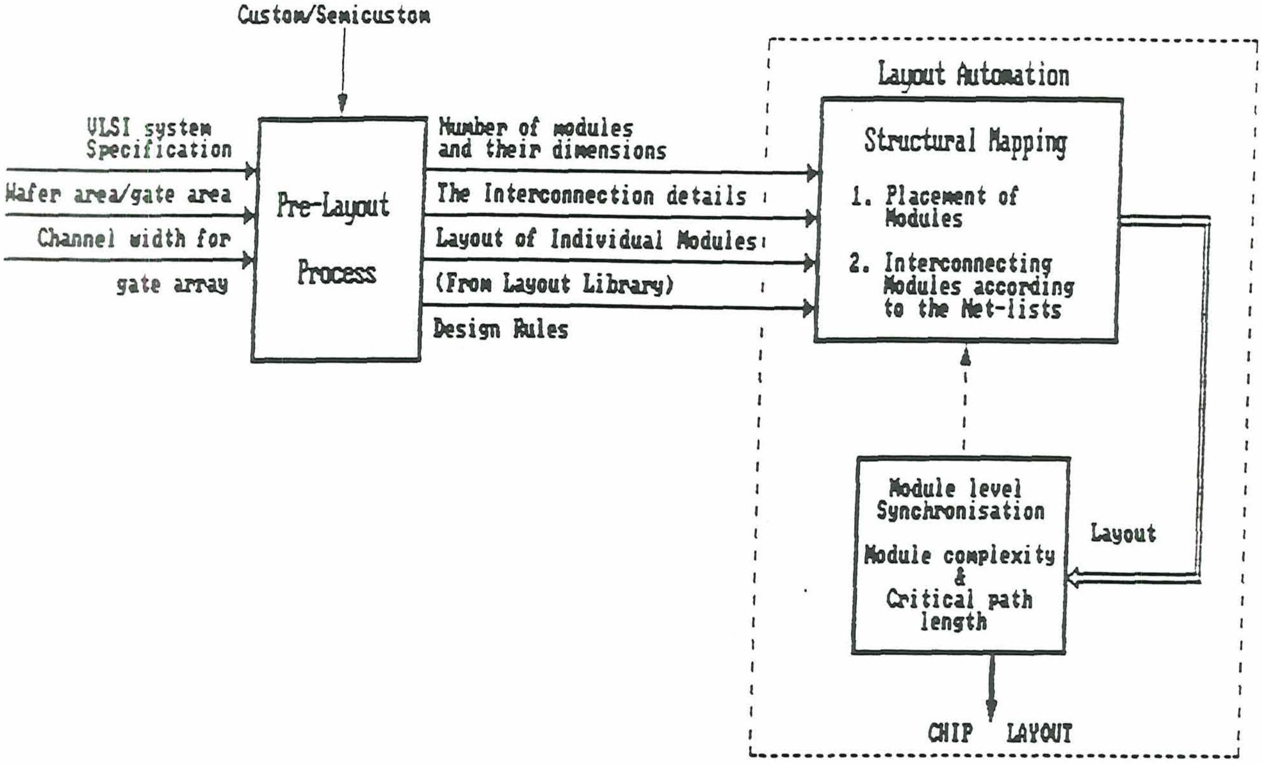


Figure 1: Layout automation problem

The Routing phase has two stages, Area estimation/Global Routing followed by Detailed Routing. In the Global Routing stage one of the several alternative routing paths available is selected. In General- Cell layouts this involves an area estimation phase during which routing space is allocated by shifting the modules as required. This ensures maximum routing completion and avoids allocation of redundant routing space. There is no such shifting process in gate arrays as the channel width (routing area) is fixed. In the Detailed Routing Phase the final wiring is implemented. If there are any incomplete nets, the process of Global and Detailed Routing is done for these nets alone. In gate array based layout automation the area estimation phase corresponds to estimating the channel capacity and the gate count required to choose the particular gate array size for maximum routing completion.

Module level synchronization becomes simpler if the sub-modules are of same type. However this may not be the case in general, requiring proper balancing of computational complexity among different sub- modules. Clock skewing due to critical path violation also leads to synchronization problem. An efficient router takes care of this by proper placement.

3 Algorithm Mapping on Processor Arrays

Algorithm mapping can be defined as the process of embedding the communication graph of an algorithm on to a target machine. Formally, if G = ¡N.E¿ represents the communication graph of an algorithm and T = ¡n.e¿ represents the target machine model then the F(G) → T is defined as mapping.

The target machine can be either fixed or reconfigurable processor array architecture (SIMD or MIMD). In the fixed array configuration the inter processor connections are fixed and communication takes place through special I/O buffers in processors. In reconfigurable arrays the processor interconnections are programmable through switches.

The steps involved in Mapping (Figure 2.a) are

1. Algorithm partitioning
2. Intra partition Problem size and Wordlength decomposition
3. Assignment of intra partition computations to processors.
4. Topological placement (relative positioning) of partitions on the processor array and establishing inter partition communication.
5. Process(partition level) synchronization
6. Programming of the processor and the switch lattice (for reconfigurable arrays).

Algorithm partitioning, Problem decomposition and assignment form the pre-mapping stage. Partitioning involves hierarchical breaking of the algorithm structure into different logical partitions (Functions F1,F2,F3,F4 in Figure 2.b) suitable to the processor functionality. Wordlength and Problem size decomposition involves breaking down each these complex functions into computations suitable to the processor grain size. 3, 4 and 5 form the major steps in mapping process.

Strategic assignment of these intra partition computations to different processors is essential for proper load balancing, maximize processor utilization and minimize communication complexity. Placement of algorithmic partitions on the processor array is done in such a way as to minimize the inter partition communication complexity. Establishing inter partition communication involves selecting data paths through the switches.

The next important step is to synchronize the different inter and intra partition level operations. This involves computation and communication level synchronization effected by proper load balancing and efficient placement. Further synchronization is effected by introducing proper delays in the communication path. Instruction and operand loading, programming the array forms the final stage of mapping.

The output of pre-mapping stage are the number of partitions, partition size (number of processors that make up the partition) and their inter and intra communication details. In algorithms like matrix multiplication where all the partitions do similar operations, balanced load assignment and placement of these partitions on the array becomes simple. In a complex algorithm like matrix LU decomposition different partitions perform different complex functions which makes load balancing difficult. This necessitates unifying the computation and communication structure of different algorithms to simplify the mapping process. While unifying the communication structure of different algorithms is a difficult process, the computation structure of a wide class of numeric algorithms can be (unified) decomposed in terms of GIPOP functions. Based on this we have proposed a novel mapping strategy in which the complex computation involved in each partition is expressed in terms of IP and CM blocks (Figure 2.c) [1][8].

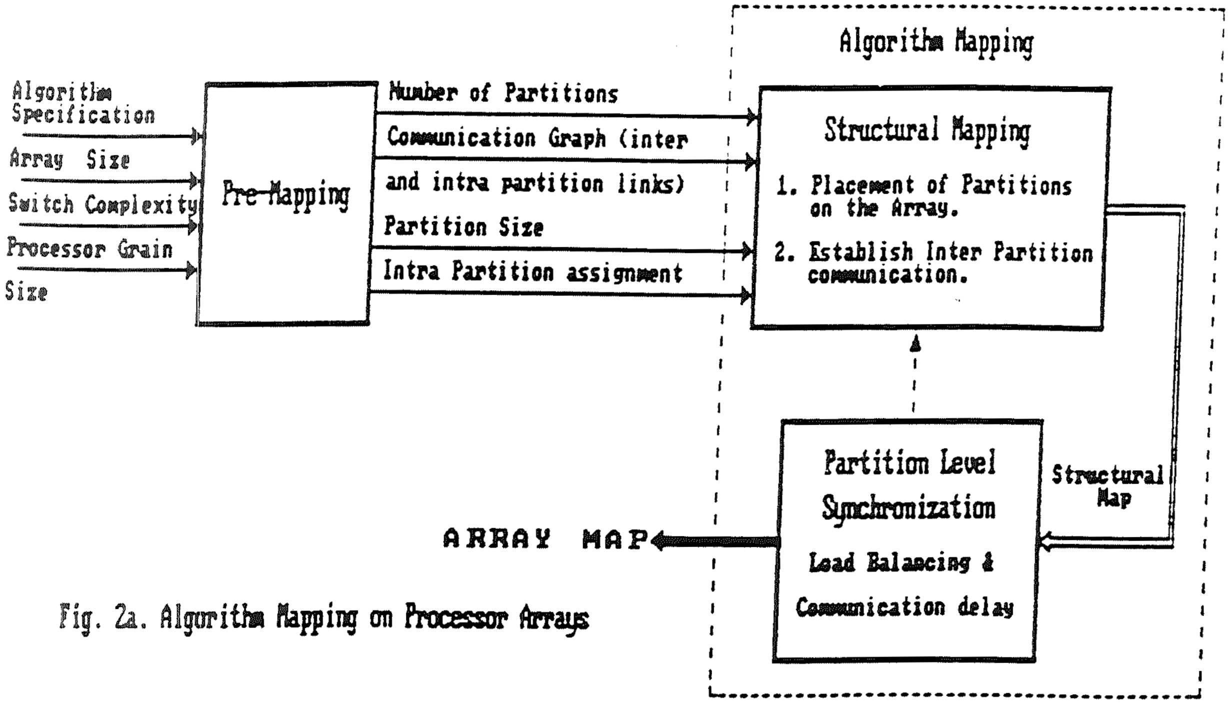

Algorithm Mapping
Algorithm Specification
Array Size
Switch Complexity
Processor Grain Size
Pre-Mapping
Number of Partitions
Communication Graph (inter and intra partition links)
Partition Size
Intra Partition assignment
Structural Mapping
1. Placement of Partitions on the Array.
2. Establish Inter Partition communication.
Structural Map
Partition Level Synchronization
Load Balancing & Communication delay
ARRAY MAP

Fig. 2a. Algorithm Mapping on Processor Arrays

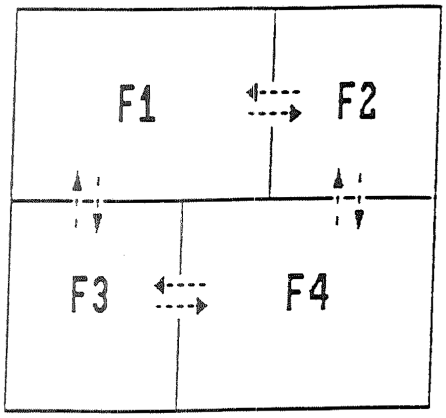


Fig.2b. Algorithm Decomposition

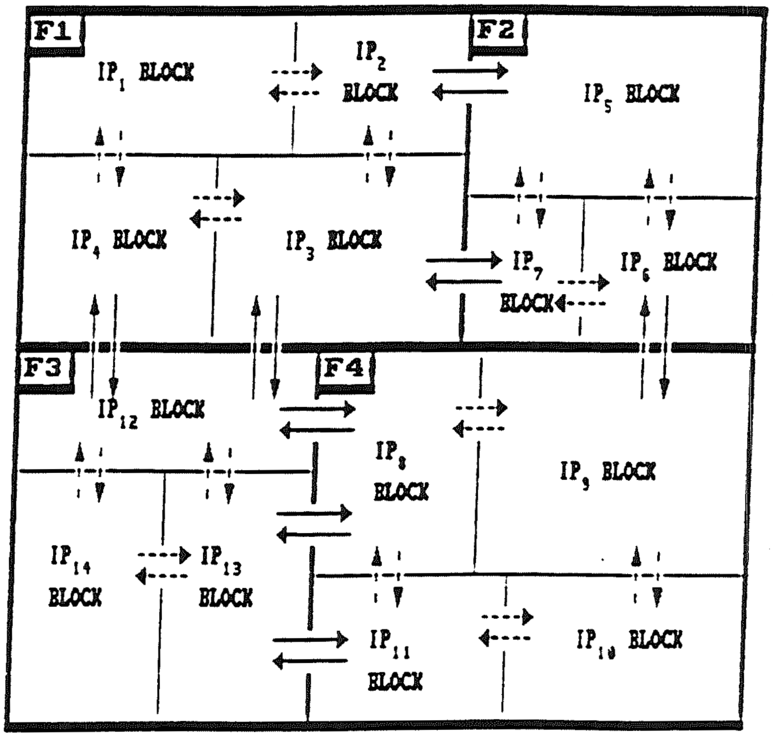


Fig.2c. Algorithm Decomposition in terms of GIPOP functional units

4 Unification of Layout Design Automation and Algorithm Mapping

The functional similarities in the problems of Layout Automation and Algorithm Mapping is explained below.

4.1 Layout Design/Algorithm Mapping Automation

Steps 1 and 2 in the layout design and algorithm mapping processes are identical but of different complexities. Assignment of intra partition computation in algorithm mapping is analogous to the design of sub- modules. While intra partition assignment to processors can be varied dynamically, the modules once designed are fixed.

Module/Partition Placement

The problem of placing algorithm partitions on the processor array is identical to the placement of layout modules on the wafer. The algorithm partitions are analogous to the layout modules and the processor array is analogous to the wafer. However while the processor array size is fixed during placement and routing process, the wafer area can be fixed as in gate arrays or varied dynamically as in custom design. Placement of primary I/O modules on the periphery is more preferable in layout design compared to that in algorithm mapping. The above criteria should thus be considered in the block-interchange step.

Module Routing / Inter and Intra Partition Communication

Establishing inter and intra partition communication is identical to the problem of routing in layouts. Both these problem involves finding the shortest path between the source and target nodes. Each node is either a gate/processor or a routing channel/switch. The switch capacity is analogous to the channel width in gate arrays. In the custom design approach there is no such routing area constraint. The obstacles in processor interconnections is similar to that in the gate arrays. In routing through channels/switches on the gate/processor array, the wire cannot enter a node if (i) the node is a gate/processor or (ii) the node forms a channel/switch but its capacity is filled by the existing routes. This is different in the custom design approach. Here the obstacles are modules themselves and wires can take any path that falls outside the boundaries of the modules (except in over-the-cell routers).

Module Level/Partition Level Process Synchronization

Balanced assignment of intra partitions computations to processors is similar to the problem of designing the sub-modules of balanced complexity in the layout design. With respect to different problems effecting process synchronization is simpler in algorithm mapping due to the flexibility in assigning the load to the processors. This is more critical in layout design as the modules once designed are fixed.

5 PALMAP - The Unified Algorithm

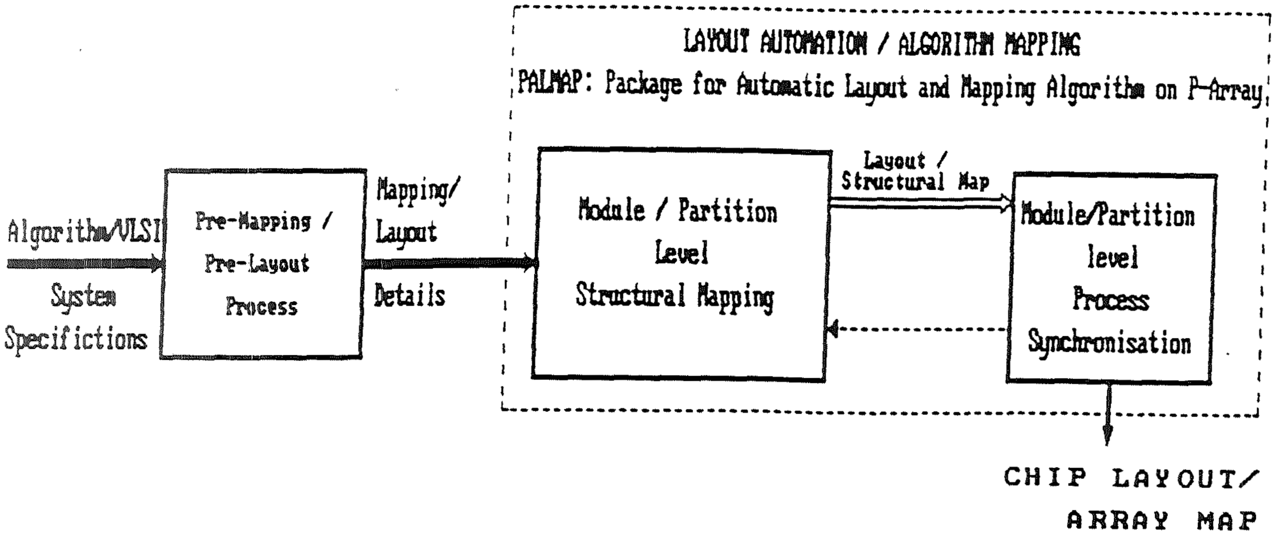

LAYOUT AUTOMATION / ALGORITHM MAPPING
PALMAP: Package for Automatic Layout and Mapping Algorithm on P-Array
Algorithm/VLSI System Specifictions
Pre-Mapping / Pre-Layout Process
Mapping/ Layout Details
Module / Partition Level Structural Mapping
Layout / Structural Map
Module/Partition level Process Synchronisation
CHIP LAYOUT/ ARRAY MAP

Fig. 3. The UNIFIED Approach.

There are several automation algorithms for VLSI system design based on custom and semi-custom approaches. The placement and the routing phase are same for these two approaches. But the constraints, heuristics applied may differ. The common heuristics in the iterative improvement phase are pairwise interchange, neighborhood interchange, force-directed technique. All these heuristics can be applied for partition placement on the processor array. Also the different wiring algorithms like Wavefront, Maze-runner etc., can be applied for establishing the communication between the partitions on the processors array. The number of routing levels in the layout design can be one or more. In algorithm mapping there is no such routing levels.

We have developed a unified package PALMAP (Package for Automatic Layouts and Mapping Algorithms on Processor-arrays) which solves the functionally identical steps in the problems of Layout Automation and Algorithm Mapping by using common procedures. PALMAP uses the same data structure for input specification of both these problems. The iterative improvement phase does a pairwise interchange based on greedy approach. The PALMAP uses the Rubin's modification of Lee's algorithm.

5.1 Example For Layout / Algorithm Mapping

A sample layout model obtained using the above algorithm is shown in Figure. 4a and 4b. The numeric structure of the FFT algorithm [8] can be decomposed in terms of the GIPOP functions as explained above. The exact decomposition process is not presented here. Figure. 5a,5b,5c,5d and 5e shows the mapping of radix 4, 16 point, 16 bit FFT using the unified approach. Figure. 6a,6b,6c.6d and 6e shows the mapping of LU decomposition of a 4 × 4 matrix.

```
MODULE DIMENSIONS (length * breadth)  4*5 , 5*3 , 4*5 , 5*3 , 4*5 , 5*3
                                      4*5 , 5*3 , 6*5 , 8*5 , 9*5 ,11*5 ,
                                      4*5 , 4*5 , 4*5 , 4*5
```

```
#Connect {
         0  r 2   1 l 2
         9  r 1  10 r 1
         5  r 0   6 b 1
         4  r 2   5 l 1
         9  r 3   8 l 1
        14  b 2  15 t 1
        13  l 1   2 r 1
         7  r 1   1 l 1
        12  l 2  14 r 1
         1  l 0   5 l 2
         1  r 0  11 t 3
         0  t 2   7 l 2
        13  l 0   5 b 2
         7  t 3   3 l 2
        12  r 2   6 r 2
        11  b 4   9 l 2
         9  t 3   8 b 4
        15  l 2   7 b 1
        14  r 0   9 t 6
         3  r 1  14 l 3
        11  r 3   8 b 2
         9  t 2  12 b 1
         9  r 0  12 b 3
        14  t 2   6 r 3
        12  r 1   6 r 4
         3  r 0   6 l 0
         5  r 1   6 l 3
         3  r 2   6 l 4
}
```

```
#ioinfo {
input :   1 b 2
input :   0 l 2
input :  10 l 2
input :  11 l 2
input :   9 l 0
input :   9 l 1
output:   7 t 1
output:   7 t 3
output:   5 t 2
output:   5 t 4
output:  14 t 3
}
```

(l-LEFT , r-RIGHT , t-TOP , b-BOTTOM)

NETWORK SPECIFICATION

Fig.4a.Module dimensions & Network Specification

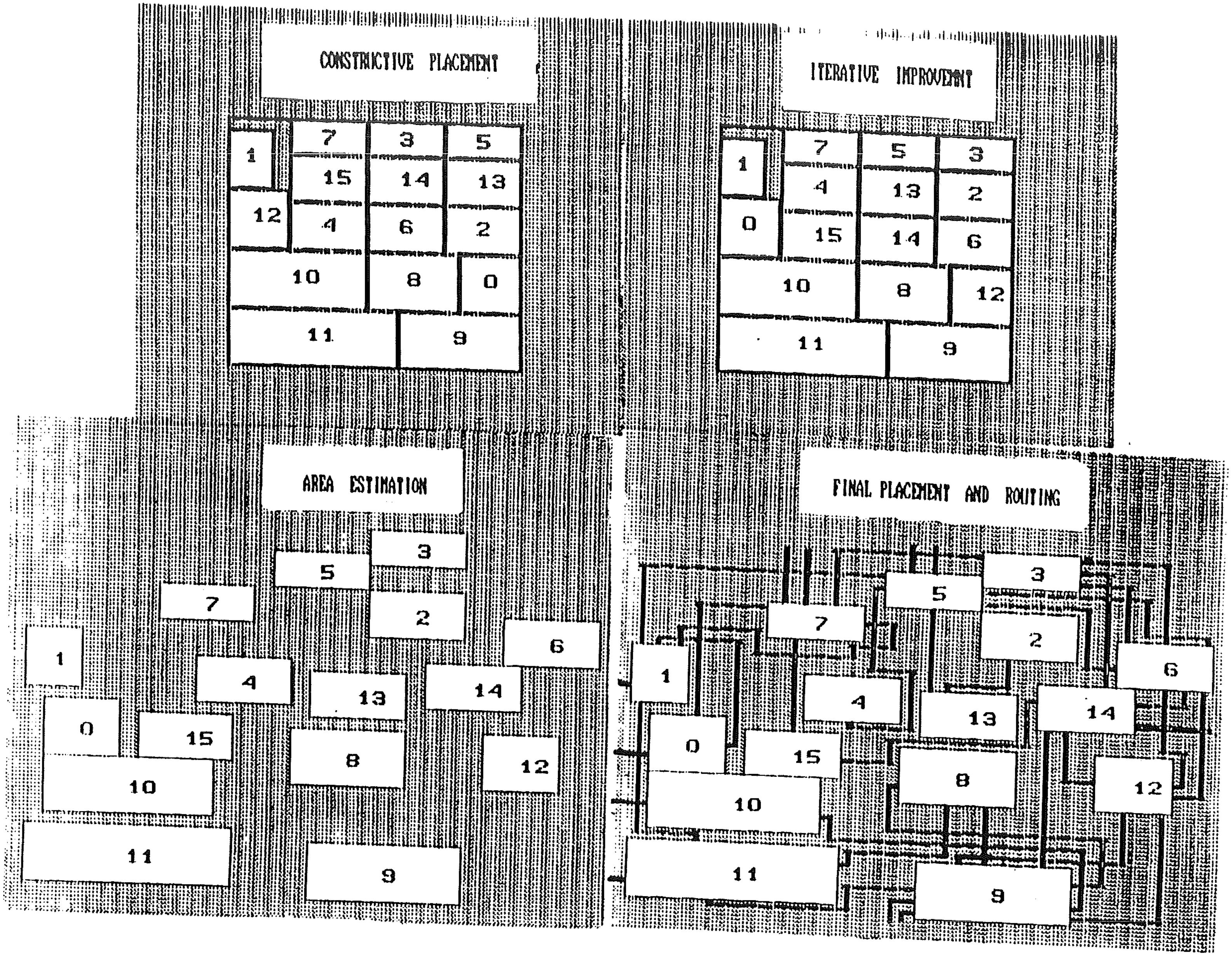
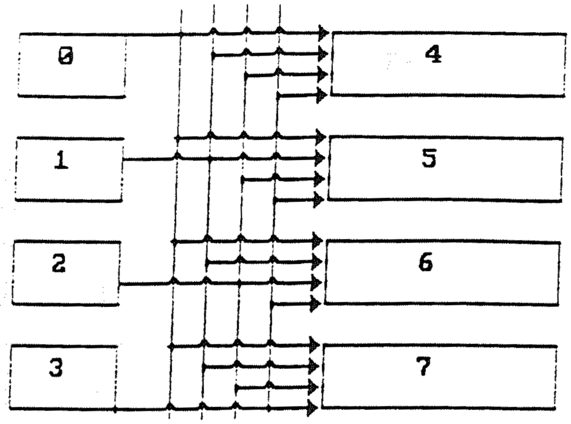


Fig. 4b SAMPLE LAYOUT MODEL



Blocks 0,1,2 and 3 Peform 16 bit 16 term Inner Product function.

Blocks 4,5,6 and 7 Peform 36 bit 16 term Inner Product function.

Fig.5a. Block Level Data Flow for FFT (Radix 4, 16 point, 16 bit)

```
FFT  PARTITION  SIZES  (LENGTH x BREDTH)  :  2 * 6 , 2 * 6 , 2 * 6 ,
                                             4 * 10, 4 * 10, 4 * 10,

# CONNECT  {
             0  r  5   4  1  0
             0  r  5   5  1  0
             0  r  5   6  1  0
             0  r  5   7  1  0
             1  r  5   4  1  1
             1  r  5   5  1  1
             1  r  5   6  1  1
             1  r  5   7  1  1
             2  r  5   4  1  2
             2  r  5   5  1  2
             2  r  5   6  1  2
             2  r  5   7  1  2
             3  r  5   4  1  2
             3  r  5   5  1  2
             3  r  5   6  1  2
             3  r  5   7  1  2
         }
```

Fig. 5b. FFT partition dimensions and inter partition communicati

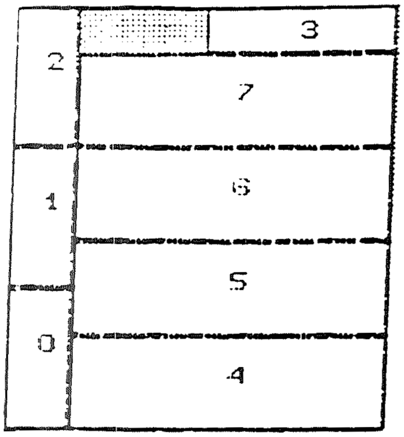


Fig. 5c PLACEMENT OF FFT PARTITIONS (IP BLOCKS)

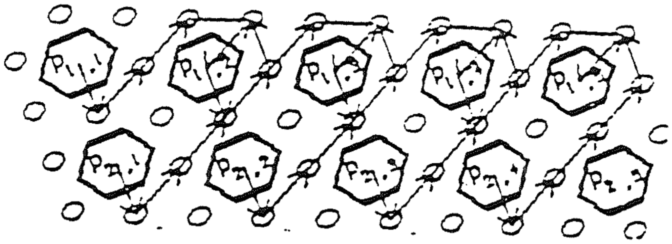


Fig. 5d PROCESSOR LEVEL DATA FLOW FOR BLOCKS(0-3) in Fig. 5

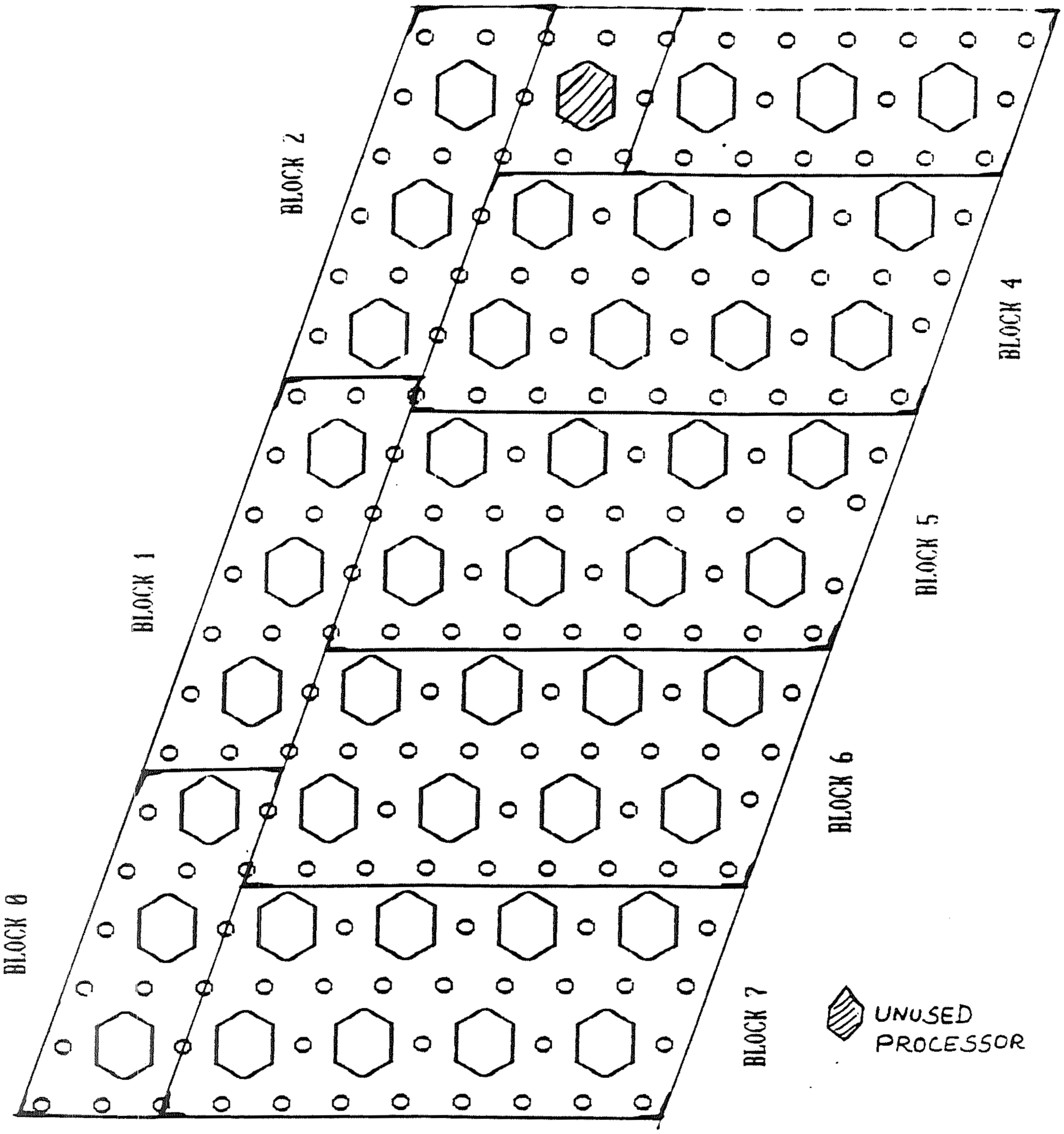
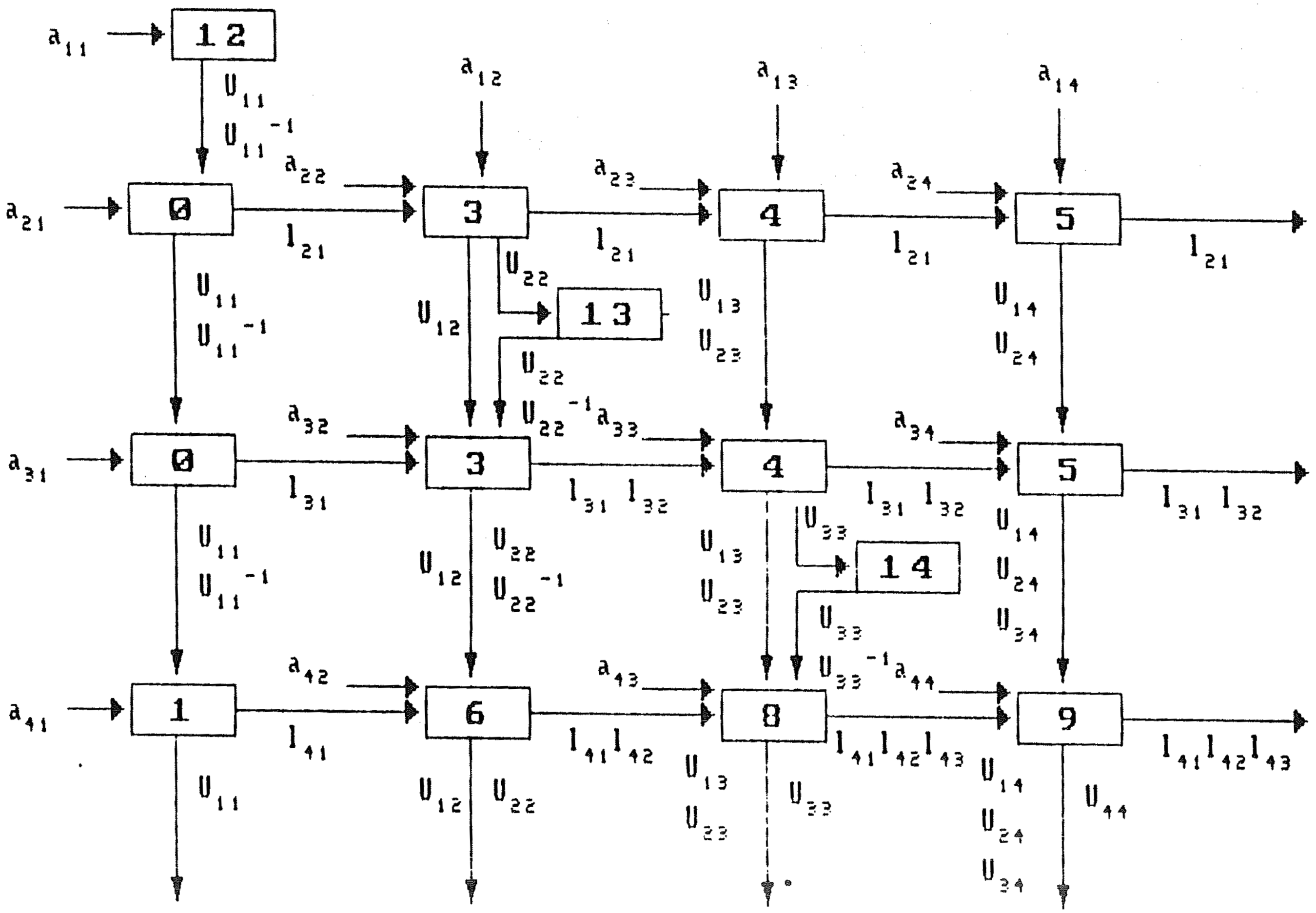


Fig. 5e FINAL PLACEMENT OF FFT PARTITIONS ON THE PROCESSOR ARRAY



Blocks 12, 13 and 14 perform reciprocal operation.

Blocks 0,1 and 2 perform 4bit x 16 bit multiplication.

Blocks 3,4,5,6 and 7 perform 4 x 20 x 16 bit multiplication.

Blocks 8,9 and 10 perform 24 bit 4 term IP

Block 11 perform 36 bit 4 term IP

** IP - Inner Product

Fig. 6a Block level flow for LU decomposition of 4 x 4 matrix.

```
LU DECOMPOSITION  PARTITION  SIZES  : 3 * 3 , 3 * 3 , 3 * 3 , 3 * 11, 3 * 11,
                 (LENGTH * BREDTH)    3 * 11, 3 * 11, 3 * 11, 13 * 13, 13 * 13,
                                      13 * 13, 13 * 13, 19 * 19, 3 * 3, 3 * 3, 3 * 3

# CONNECT  {

 0 b 1   3 t 10
 0 b 1   3 t 8
 0 b 1   3 t 6
 0 b 1   3 t 4
 0 b 1   3 t 2
 0 b 1   4 t 10
 0 b 1   4 t 8
 0 b 1   4 t 6
 0 b 1   4 t 4
 0 b 1   4 t 2
 0 b 1   5 t 10
 0 b 1   5 t 8
 0 b 1   5 t 6
 0 b 1   5 t 4
 0 b 1   5 t 2

 1 b 1   6 t 10
 1 b 1   6 t 8
 1 b 1   6 t 6
 1 b 1   6 t 4
 1 b 1   6 t 2
 1 b 1   9 t 10
 1 b 1   9 t 8
 1 b 1   9 t 6
 1 b 1   9 t 4
 1 b 1   9 t 2
 1 b 1   8 t 10
 1 b 1   8 t 8
 1 b 1   8 t 6
 1 b 1   8 t 4
 1 b 1   8 t 2

 2 b 1   7 t 10
 2 b 1   7 t 8
 2 b 1   7 t 6
 2 b 1   7 t 4
 2 b 1   7 t 2
 2 b 1   10 t 10
 2 b 1   10 t 8
 2 b 1   10 t 6
 2 b 1   10 t 4
 2 b 1   10 t 2
 2 b 1   11 t 10
 2 b 1   11 t 8
 2 b 1   11 t 6
 2 b 1   11 t 4
 2 b 1   11 t 2

 3 b 1   6 t 2
 3 b 1   7 t 2

 4 b 1   8  t 10
 4 b 1   10 t 10
 4 b 9   8  t 10
 4 b 3   8  t 8
 4 b 5   8  t 6
 4 b 7   8  t 4
 4 b 1   8  t 2
 4 b 9   8  t 12
 4 b 9   10 t 10
 4 b 3   10 t 8
 4 b 5   10 t 6
 4 b 7   10 t 4
 4 b 1   10 t 2
 4 b 9   10 t 12

 5 b 9   9  t 10
 5 b 3   9  t 8
 5 b 5   9  t 6
 5 b 7   9  t 4
 5 b 1   9  t 2
 5 b 9   9  t 12
 5 b 9   11 t 10
 5 b 3   11 t 8
 5 b 5   11 t 6
 5 b 7   11 t 4
 5 b 1   11 t 2
 5 b 9   11 t 12

 9 b 10   11 t 2
 9 b 1    11 t 4
 9 b 3    11 t 6
 9 b 5    11 t 8
 9 b 7    11 t 10
 9 b 9    11 t 12
 9 b 11   11 t 14
 9 b 11   11 t 16
 9 b 11   11 t 18

10 b 10   11 t 2
10 b 1    11 t 4
10 b 3    11 t 6
10 b 5    11 t 8
10 b 7    11 t 10
10 b 9    11 t 12
10 b 11   11 t 14
10 b 11   11 t 16
10 b 11   11 t 18

 6 b 1    8  t 2
 6 b 3    8  t 4
 6 b 5    8  t 6
 6 b 7    8  t 8
 6 b 9    8  t 10
 6 b 11   8  t 12
 6 b 1    9  t 2
 6 b 3    9  t 4
 6 b 5    9  t 6
 6 b 7    9  t 8
 6 b 9    9  t 10
 6 b 11   9  t 12

 7 b 1    10 t 2
 7 b 3    10 t 4
 7 b 5    10 t 6
 7 b 7    10 t 8
 7 b 9    10 t 10
 7 b 11   10 t 12
 7 b 1    10 t 2
 7 b 3    10 t 4
 7 b 5    11 t 6
 7 b 7    11 t 8
 7 b 9    11 t 10
 7 b 11   11 t 12

 3 b 1    12 t 2
 3 b 3    12 t 2
 3 b 5    12 t 2
 3 b 7    12 t 2
 3 b 9    12 t 2
 3 b 9    12 t 2

 8 b 1    13 t 2
 8 b 3    13 t 2
 8 b 5    13 t 2
 8 b 7    13 t 2
 8 b 9    13 t 2
 8 b 11   13 t 2
 8 b 11   13 t 2
 8 b 11   13 t 2

14 b 1   0 t 2
14 b 1   0 t 2
14 b 1   0 t 2
14 b 1   0 t 2
14 b 1   1 t 2
14 b 1   1 t 2
14 b 1   1 t 2
14 b 1   1 t 2
14 b 1   2 t 2
14 b 1   2 t 2
14 b 1   2 t 2
14 b 1   2 t 2

12 b 1   6 t 2
12 b 1   6 t 4
12 b 1   6 t 6
12 b 1   6 t 8
12 b 1   7 t 2
12 b 1   7 t 4
12 b 1   7 t 6
12 b 1   7 t 8

13 b 1   10 t 2
13 b 1   10 t 4
13 b 1   10 t 6
13 b 1   10 t 8

}
```

Fig. 6b LU DECOMPOSITION PARTITIONS AND INTER PARTITION COMMUNICATION

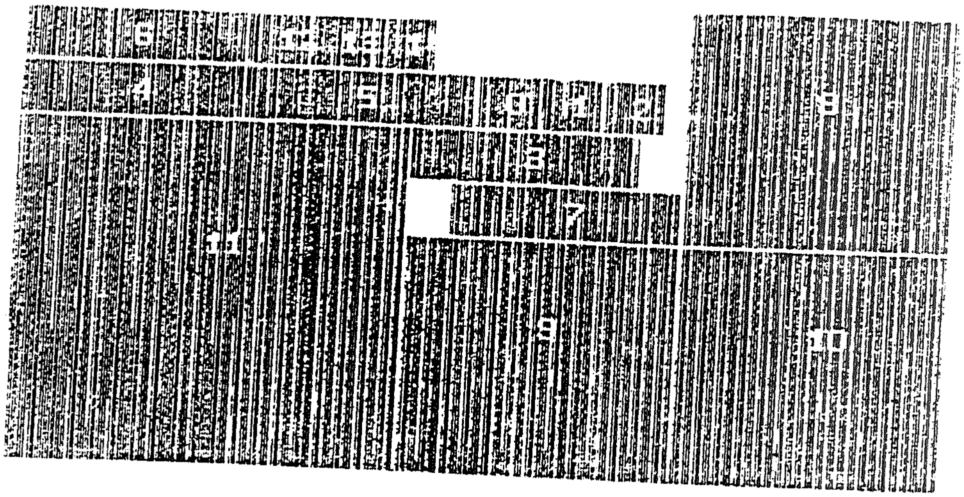


Fig. 6c. INITIAL PLACEMENT OF LU DECOMPOSITION PARTITIONS

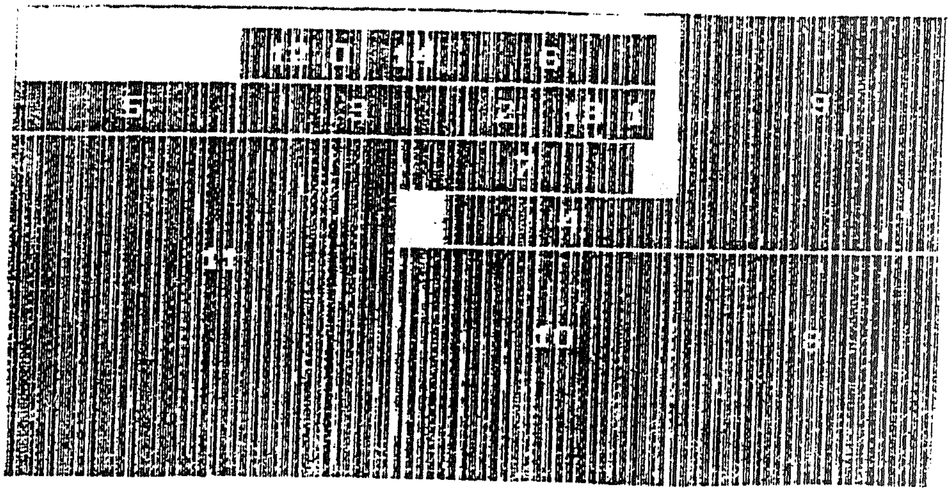


Fig. 6d. FINAL PLACEMENT OF LU DECOMPOSITION PARTITIONS

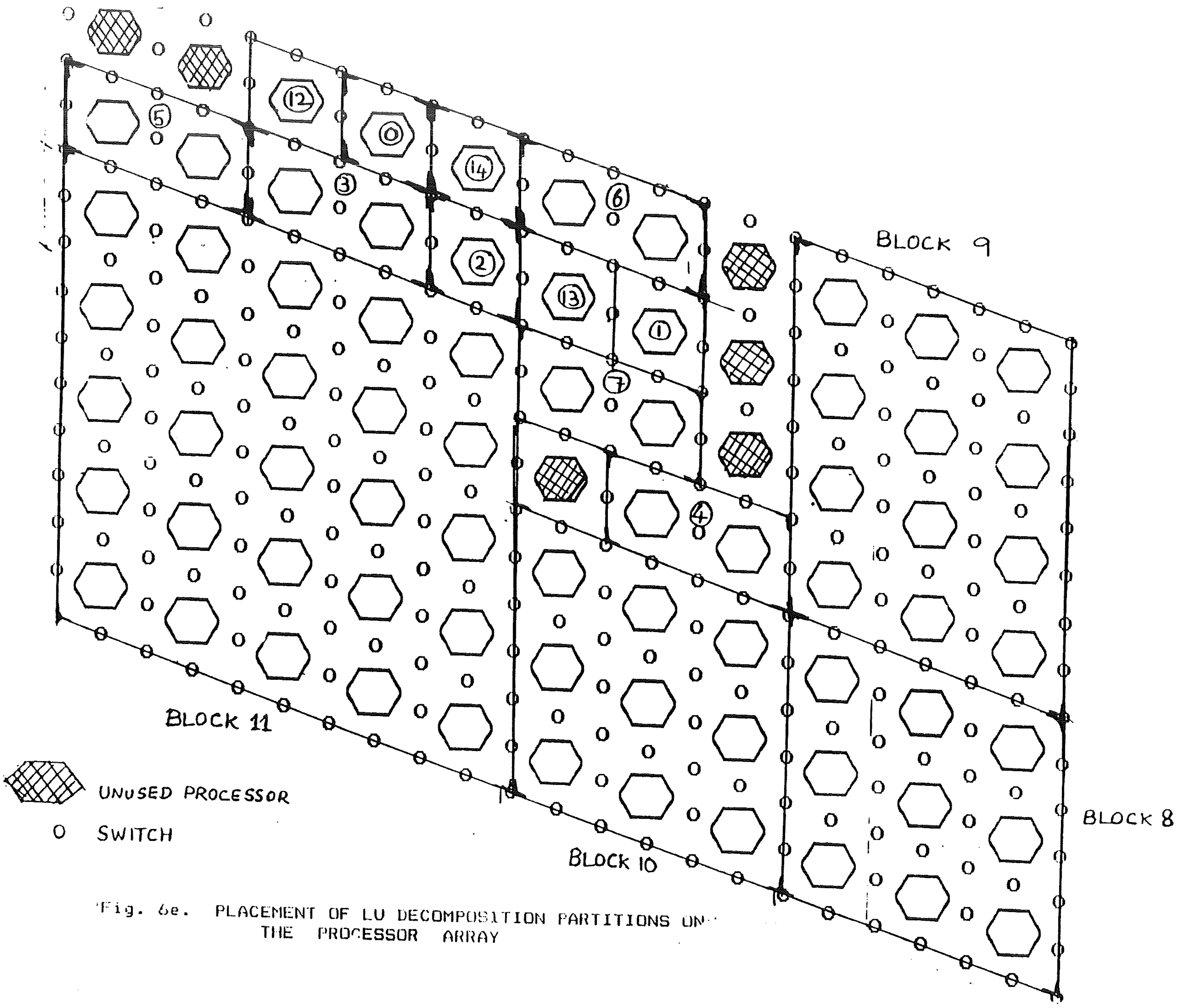


Fig. 6e. PLACEMENT OF LU DECOMPOSITION PARTITIONS ON THE PROCESSOR ARRAY

6 Conclusion

The above process of unifying the layout automation and algorithm mapping will help to integrate the software tools of Special Purpose Silicon Compiler, thereby reducing the turn around time and cost. We are further working towards refining the unified approach. The unified approach for process synchronization in both these problems is being developed.

References

[1] N.Venkateswaran, et.al, 'PAcube VLSI Arrays for SuperComputers', Proceedings of the First International Conference on SuperComputing, conducted by IEEE computer Society and Pentagon, Florida, December 1985.

[2] N. Venkateswaran, et.al, "A Special Purpose Silicon Compiler for designing Supercomputing VLSI Systems", proc. of 3rd NASA SERC Symposium on VLSI Design. University of Idaho, Idaho, Moscow, Oct 1991.

[3] E. S. Kuh, "Recent advances in VLSI layouts", proc. IEEE, vol. 78, No. 2, pp. 237-263. Feb. 1990.

[4] J. Soukup, "Circuit Layout", proc. IEEE, vol. 69, No. 10, 1281-1304, Oct. 1981.

[5] Thomas Lengauer, "Combinatorial Algorithms for Integrated Circuit Layouts", John Wailey & Sons.

[6] S. Y. Kung, "VLSI Array Processor", Prentice Hall.

[7] Leah H.Jamieson. Dennis Gannon and Robert J.Douglas(Eds) 'The Characteristics of Parallel Algorithms', MIT Press, 1978.

[8] N.Venkateswaran et al., 'A Cost Effective Methodology for Designing Massively Parallel VLSI Functional Units', submitted to 5th NASA symposium on VLSI design, Albuquerque, University of New Mexico.

[9] Lawrence R.Rabiner and Bernard Gold, 'Theory and applications of Digital Signal Processing', Prentice Hall.

omit

Session 12
Signal Processing

Joe Feeley

A Fixed/Variable Bit-Rate Data Compression Architecture [1]

Gregary C. Zweigle, Jack Venbrux† and Pen-Shu Yeh‡
Advanced Hardware Architectures
PO Box 9669
Moscow ID 83843
† NASA Space Engineering Research Center for VLSI Design
University of New Mexico
2650 Yale S.E., Suite 101
Albuquerque, NM 87106
jvenbrux @groucho.mrc.unm.edu
‡ NASA Goddard Space Flight Center, Code 738
Greenbelt M.D. 20771
psyeh@psy.gsfc.nasa.gov

Abstract - **A VLSI architecture for an adaptive data compression encoder capable of sustaining fixed or variable bit-rate output has been developed. There are three modes of operation: lossless with variable bit-rate, lossy with fixed bit-rate and lossy with variable bit-rate. For lossless encoding, the implementation is identical to the USES chip designed for Landsat 7. Obtaining a fixed bit-rate is achieved with a lossy DPCM algorithm using adaptive, nonuniform scalar quantization. In lossy mode, variable bit-rate coding uses the lossless sections of the encoder for post-DPCM entropy coding. The encoder shows excellent compression performance in comparison to other current data compression techniques. No external tables or memory are required for operation.**

1 Introduction

Compression of digital data can be attained by eliminating redundancy. In the ideal case, redundancy is eliminated and only random noise (the information) plus the model used for compression are transmitted. Data compression techniques can be divided into two broad categories, lossless and lossy. The lossless approach loses no data in the compression process and therefore the data after decompression is identical to the data prior to compression. This is necessary when information loss cannot be tolerated. Applications include scientific research and medical diagnosis. The lossless technique has two key disadvantages; the amount of compression attainable is limited and the compressed bit-rate is variable.

Lossy data compression can overcome these disadvantages. To increase the compression ratio, "unnecessary" data is eliminated in the compression process. The definition of "unnecessary" depends on the application. For entertainment purposes, data which cannot be perceived by a human visual system is considered "unnecessary" and is subject to elimination. For scientific purposes, loss can sometimes be acceptable under the constraint that the user understand characteristics of the information that is lost.

[1] This work was supported in part by NASA under Space Engineering Research Center grant NAGW-3292.

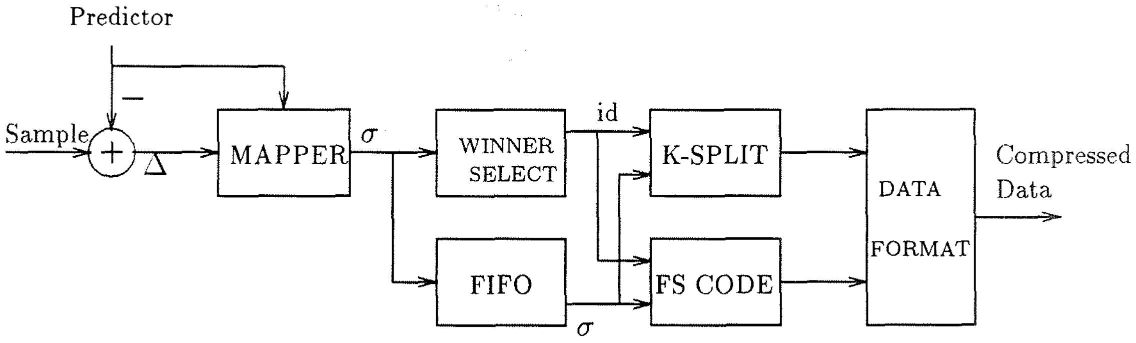


Figure 1: Block diagram of the lossless implementation.

Compression using variable bit-rate techniques result in a bit-rate which cannot be predicted *a-priori* and which can fluctuate wildly. For some applications this is not acceptable. An example is the Landsat 7 satellite which, on occasion, must be able to transmit a known and fixed amount of information. For applications requiring the fixed bit-rate compression, a lossy algorithm can set the compressed bit-rate, independent of the bit-rate of the source by eliminating data which exceeds the required bit-rate.

The chip described in this paper is designed to be user selectable for all three modes. For lossless data compression, input quantization can be between 4 and 15 bits per sample. The output bit-rate is variable, dependent on the entropy of the source. For fixed bit-rate, lossy data compression, the output quantization can be fixed at four or five bits per sample. For variable bit-rate, variable data compression, the output bit-rate is dependent on the quality setting and the entropy of the source. The compressed data format and packetization settings are identical for all compression modes resulting in a uniform data representation.

The paper is organized as follows. Section 2 briefly describes the lossless algorithm and its implementation. For more information on this, the reader is referred to [4] in this publication. Section 3 details the lossy section of the encoder. Section 4 compares compression performance results of the encoder with other compression techniques.

2 Lossless Compression Algorithm

For lossless encoding, the chip implements Rice's algorithm [3]. The entropy coding module in the Rice algorithm has been shown to be equivalent to a set of Huffman codes for a modified laplacian symbol source [8]. The Rice algorithm provides efficient compression for a wide range of source entropy, and requires little overhead.

Rice's algorithm is very amenable to hardware implementation. Figure 1 shows a block diagram of the lossless section of the chip. The predictor removes redundancy from the source. Because of the arithmetic operation used in the predictor, the Δ values which exit the predictor need an extra bit for representation over the number of bits needed for representation of the input `Sample` values. The MAPPER block losslessly removes this extra bit and puts the data into a form suitable for encoding. The mapped Δ values are called σ values. Next, the WINNER SELECT block chooses one of a set of possible codes based on which code can best compress the σ values. The winning code number is represented with a

3 or 4-bit id value. Meanwhile, a FIFO is used to store the σ values while the winning code option is being selected. The σ values are then coded with the winning code and formatted for output in the K-SPLIT, FS CODE and OUTPUT FORMAT blocks. No external RAM is required for this process. Furthermore, only one pass of the data through the chip is required.

The code options include several k-split options, two low entropy options, and one default option. The k-split codes each split off the least significant k bits. The top N-k bits are then coded with a comma code, where N is the size of the input word. A comma code (called fundamental sequence, FS, in [3]) encodes input symbols with a number of zeros equal to the value of the symbol followed by a one. The k least significant bits are packed uncoded with the FS coded N-k bits. The k-split codes provide efficient coding for average source entropies from one to N bits per sample.

At low entropy, the performance of the k-split options is lower bounded by one bit per sample [8]. Because of this, the chip adds two special low entropy coding options. One is an algorithm which groups single samples into pairs and encodes the doublet as a new single sample. This takes advantage of reduced correlation in the higher order statistics. The other option is a run-length coder which encodes from one to 64 block runs of zeros. At the end of the run the number of blocks of zeros is coded with a comma-type code.

The default option is used when the entropy of the σ values does not allow efficient coding. This upper bounds any data expansion that could occur for this case. The worst case data expansion then is equal to the bit-rate at the input plus a small overhead due to id bits.

The coding is performed on a block-by-block basis. This provides adaptivity to changing source statistics. Each block is coded with the coding option that gives the lowest bit-rate. An id value transmitted with each block tells the decoder which option was used for encoding. The id value is 4-bits for input quantization greater than 8-bits per sample and 3-bits for smaller input quantization. An extra "twiddle" bit is added to the id to account for low entropy options when necessary.

3 Lossy Compression Algorithms

The lossy section of the chip implements an adaptive, differential pulse code modulation (ADPCM) scheme. Figure 2 shows the basic operation of a DPCM algorithm, both encoder and decoder. Adaptivity of the DPCM algorithm is accomplished by adjusting quantization tables based on the variance of the source seen by the quantizer for each block. These δ values occur after a prediction step, which is similar to the one used for lossless encoding. Because of this similarity, the lossless section of the chip is used to attain the variance estimate of a block of δ values. The variance is estimated by using a relationship between the code option chosen and the probability density function of the σ values [8]. It is the similarity between a block of σ values and a block of δ values that allow the id, which is output from the WINNER SELECT block, to be used as an estimate of the variance of the block of δ values.

One key difference between a block of σ values and a block of δ values is the fact that the quantizer in the DPCM loop can cause slope overload [1]. Slope overload is a condition inherent to the DPCM algorithm in which edges are represented by ramps. This is caused

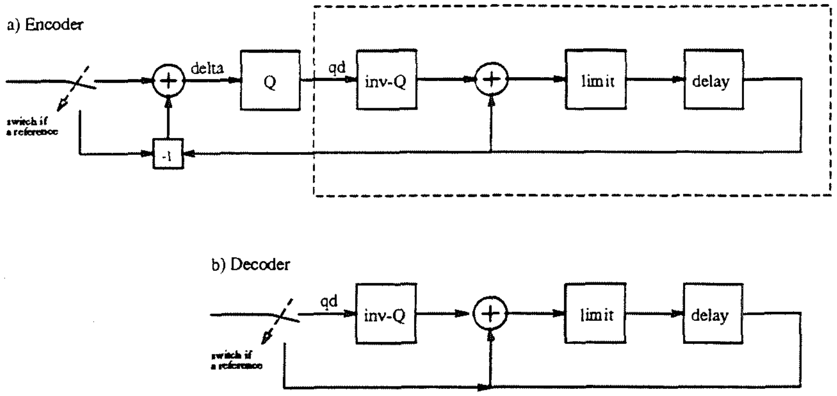


Figure 2: Basic DPCM algorithm.

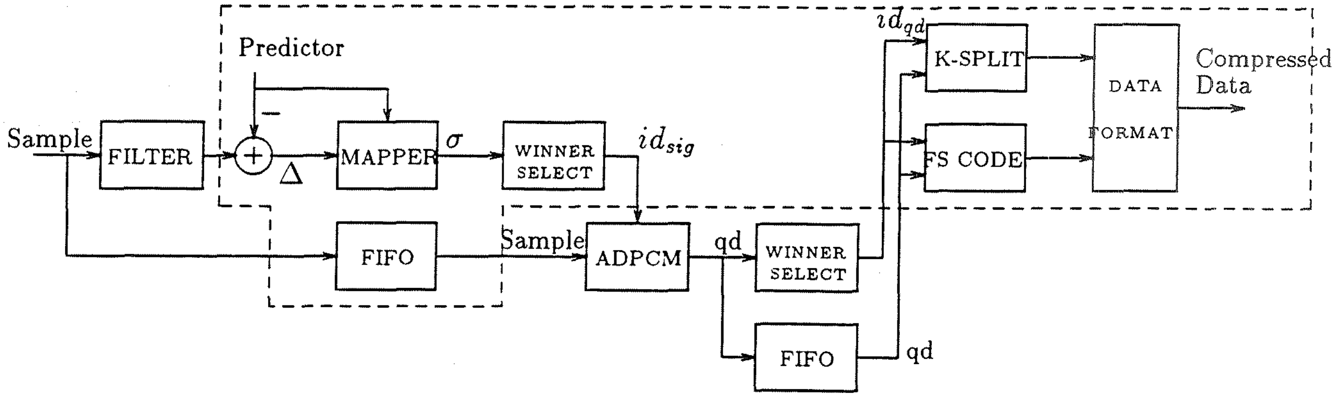


Figure 3: Block diagram of the lossy implementation. The section in the dashed box is the re-used lossless encoder.

by a quantizer limitation when trying to represent values outside the expected dynamic range. Simulations show that using an adaptive algorithm based on σ values to predict the variance of the δ values can exacerbate the slope overload problem. To compensate for this, the solution is to lowpass filter the `Sample` data before prediction. The lowpass filter approximates the slope overload condition and results in σ values which provide a better estimate of the variance of the δ values.

The `id` value used to adjust the quantization tables for a block of δ values is passed to the DPCM section a full block before the data arrives. This allows a recursive computation of the quantization tables which eliminates the need to store all quantization tables on chip. The resulting area savings is obtained at the expense of a minor increase in the latency of the chip.

Figure 3 shows a block diagram of the implementation of the lossy section of the chip. The data first goes through a lowpass filter in the FILTER block. The data then proceeds through the chip as it would in the lossless case. WINNER SELECT outputs an `id` value which is used to choose a nonuniform quantization table. The original input data (unfiltered), meanwhile, arrives after a FIFO delay and is coded with ADPCM.

Following quantization, the quantized delta values (qd) are sent to another WINNER SELECT block. This is the only block of the lossless section that needs to be duplicated for entropy coding of lossy outputs. The K-SPLIT, FS CODE, and OUTPUT FORMAT blocks are multiplexed for use by either lossy or lossless coding. If fixed bit-rate is required, these blocks are set to default winning and send the qd data out of the chip without entrpy coding.

For fixed bit-rate lossy data compression, the `id` value represents the quantization table chosen. For variable bit-rate coding, two `id`s are needed. One for the quantization table, and one for the lossless encoding option chosen. These two `id`s are mapped into a single 6-bit `id` for greater than 8-bits per sample input quantization and a single 5-bit `id` for smaller input quantization. No "twiddle" bit is required.

4 Performance Comparison

In this section 26 USC standard 8-bit images have been compressed and decompressed with the encoder and other popular compression algorithms. All three modes of operation, lossless, lossy variable bit-rate, and lossy fixed bit-rate are considered.

4.1 Lossless Comparisons

Figures 4 and 5 show lossless image compression comparisons. The comparisons are made against arithmetic coding as implemented by [7], UNIX pack, UNIX compress, gzip in best compression mode, and lossless mode JPEG [2, 5] for all 26 standard images. In Fig. 4a) it is seen that the compression performance is nearly identical to JPEG in Huffman lossless mode. However, the JPEG algorithm requires two passes of the data to attain this compression performance. The first pass builds Huffman tables based on the statistics of the input data. The second pass does the actual encoding. The lossless algorithm requires only one pass of the data. For Huffman lossless JPEG, one pass configuration is possible, with an associated loss in performance (increase in bit-rate). The encoder outperforms UNIX pack which is also a Huffman based algorithm.

Figure 4b) shows comparison to arithmetic coding, gzip (which is an LZ-77 based algorithm) and UNIX compress (which is a commonly used LZW-based algorithm [6]). The lossless algorithm outperforms these algorithms in all but two images.

Figure 5 shows comparisons to the compress, gzip, arithmetic, and pack encoders when the predictor and mapper from the lossless encoder front end are used as a preprocessor to these other algorithms. Adding the preprocessor for JPEG is not useful because JPEG already has a predictor. Figure 5a) shows comparisons with the Lempel-Zip based algorithms and Fig. 5b) shows comparisons with arithmetic coding and UNIX pack. Arithmetic coding and UNIX pack improve substantially with the help of the mapping preprocessor. The gzip encoder also is improved.

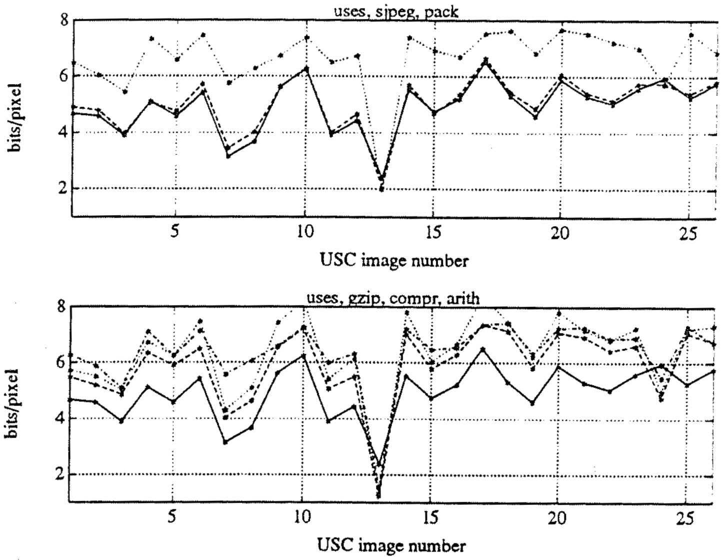


Figure 4: In both figures, performance of lossless encoder (USES) is solid line. In the top figure, JPEG is represented with the dashed line and UNIX pack with the dotted line. In the bottom figure, gzip is represented with the dashed line, UNIX compress with the dotted line, and arithmetic coding with the dashed-dotted line.

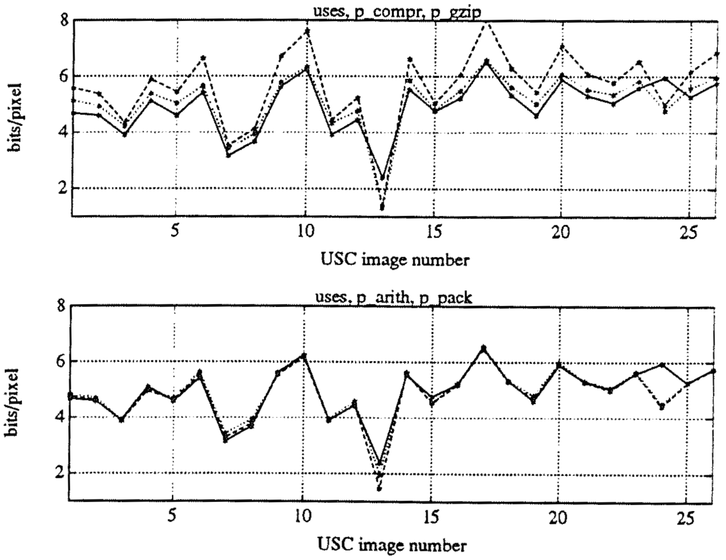


Figure 5: In all figures, performance of lossless encoder (USES) is solid line. In the top figure, UNIX compress is represented with the dashed line and gzip is represented with the dotted line. In the bottom figure, arithmetic coding is represented with the dashed line and UNIX pack is represented with the dotted line.

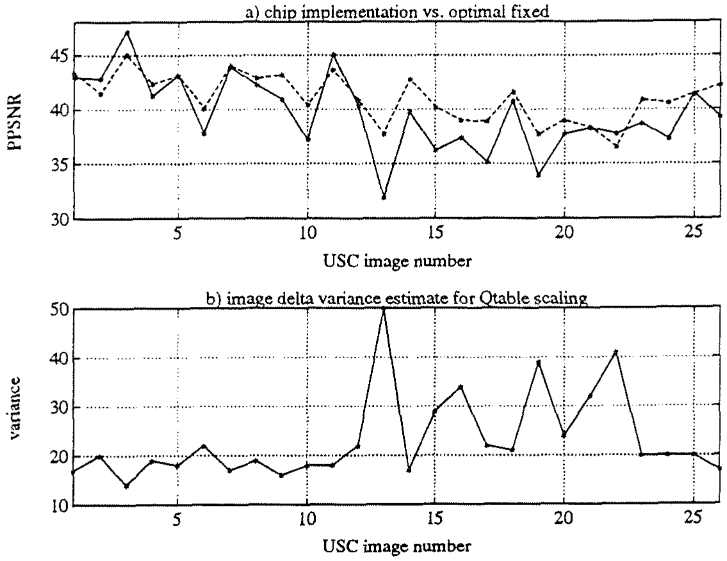


Figure 6: In top figure, fixed bit-rate encoder is solid line. Optimal encoder is dashed line.

4.2 Lossy Comparisons

An optimal DPCM coder is used to demonstrate the performance of the fixed bit-rate coder. The optimal coder implements a fixed (non-adaptive) Max [1] nonuniform quantizer, optimized in the mean-square error sense and for a source assumed to have laplacian distribution. These assumptions are valid for DPCM coded image data. The optimization is performed over the entire image which results in the optimal solution being nearly impossible to implement in VLSI. The entire image data must be passed through the coder multiple times while the algorithm converges to the optimal solution. In Fig. 6a), peak-to-peak signal-to-noise ratio is the performance measure for data that has been compressed from 8-bits per sample to 4-bits per sample. The solid line is the fixed bit-rate coder performance. The dashed line is the optimal encoder performance. It is seen that the fixed bit-rate encoder performs very close to optimal and sometimes even has a higher signal-to-noise ratio. This better performance is due to the advantage of adaptivity. Figure 6b) shows the variance estimate of the entire image δ values used for optimal quantization scaling.

Figure 7 shows a comparison between the lossy fixed bit-rate and lossy variable bit-rate compression data rate. The fixed bit-rate is fixed at just over four bits per sample. The slight overhead is due to id data and and occasional reference which must be transmitted. The amount of side information is fixed when the chip is initialized. Entropy coding of the post-DPCM data allows larger compression. The entropy of the post-DPCM data shows the best that can be achieved. It is seen that the lossy variable bit-rate operation results in a bit-rate which is close to the ideal.

Comparing Figures 4 and 7, an interesting phenomenon is observed. Sometimes the lossless bit-rate is lower than the lossy fixed bit-rate. This is due to the fact that fixed bit-rate operation is fixing the bit-rate at something less than the original bit-rate (in this case, 8-bits per sample go to just over 4-bits per sample). The lossless mode operation is

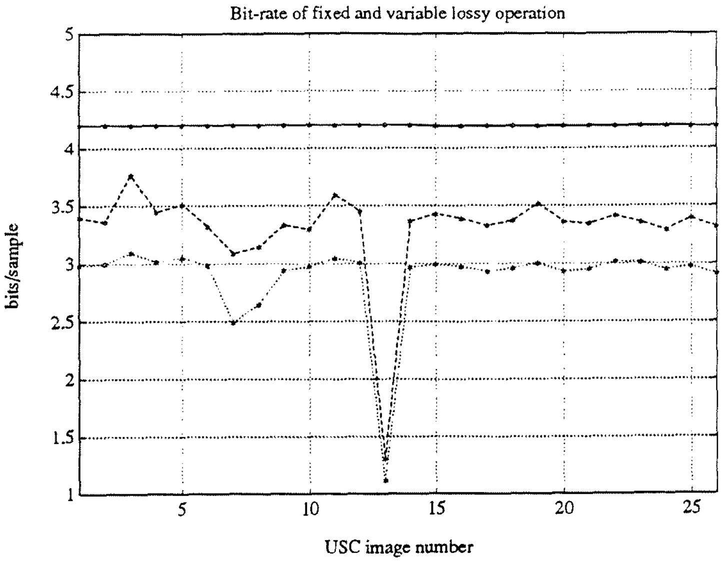


Figure 7: Fixed bit-rate is solid line. Variable bit-rate is dashed line. Dotted line is entropy of *qd* data which shows ideal performance.

attaining the lowest bit-rate possible under the constraint of no data loss. When entropy coding is applied to the fixed bit-rate data then the performance is better than the lossless variable bit-rate data.

5 Conclusion

A multipurpose data compression architecture has been described, capable of a fixed or variable bit-rate. For lossless, variable bit-rate data compression, bit-rate comparisons show that these algorithms perform better than or comparable to other state-of-the-art data compression techniques. For lossy and fixed bit-rate data compression, bit-rate and signal-to-noise comparisons with optimal solutions demonstrate the excellent performance of the algorithms and their implementation.

References

[1] N.S. Jayant and P. Knoll, *Digital Coding of Waveforms*, Prentice Hall, Englewood Cliffs, NJ, 1984.

[2] JPEG Committee Draft ISO/IEC CD 10198-1.

[3] R.F. Rice, P.S. Yeh, and W.H. Miller, "Algorithms for a Very High Speed Universal Noiseless Coding Module," *JPL Publication 91-1*, Jet Propulsion Laboratory, JPL Publication Office, Mail Stop 111-130, Pasadena, CA 91109, Feb. 15, 1991.

[4] J. Venbrux, G. Zweigle, J. Gambles, D. Wiseman, W. Miller, P. Yeh, "An Adaptive, Lossless Data Compression Algorithm and VLSI Implementations," *Fifth NASA Symposium on VLSI Design*, Nov. 1993.

[5] Gregory K. Wallace, "The JPEG Still Picture Compression Standard," *Comm. of the ACM*, vol. 34, pp. 31-44, April 1991.

[6] T.A. Welch, "A Technique for High-Performance Data Compression," *IEEE Computer*, vol. 17, no. 6, pp.8-19, June 1984.

[7] I.H. Witten, R.M. Neal, and J.G. Cleary, "Arithmetic Coding for Data Compression," *Commun. ACM*, vol. 30, no. 6, pp.520-540, June 1987.

[8] P.S. Yeh, R.F. Rice, and W.H. Miller, "On the Optimality of Code Options for a Universal Noiseless Coder" *JPL Publication 91-2*, Jet Propulsion Laboratory, JPL Publication Office, Mail Stop 111-130, Pasadena, CA 91109, Feb. 15, 1991.

A CAM-Based LZ Data Compression IC

K. Winters, R. Bode† and E. Schneider‡
Microelectronics Research Center
University of Idaho
Moscow ID 83843
† Advanced Hardware Architectures, Inc.
P.O. Box 9669
Moscow ID 83843
‡ Video Lottery Consultants, Inc.,
Bozeman, MT

Abstract - **A custom CMOS processor is introduced that implements the Data Compression Lempel-Ziv (DCLZ) standard, a variation of the LZ2 Algorithm. This component presently achieves a sustained compression and decompression rate of 10 megabytes/second by employing an on-chip content-addressable memory for string table storage.**

1 Introduction

The performance of textual substitution data compression methods depends primarily upon the speed at which a dictionary or string table may be searched for the occurrence of previously encountered strings. While the use of an associative or content-addressable memory (CAM) to perform dictionary searches in a single clock cycle was proposed by Welch [Welch84] for when "true associative memories are available in this [sufficient] size," most prior lossless compression integrated circuits store the dictionary in an external SRAM and employ hash coding to accelerate searches. This necessitates the expense of multiple component solutions, requires more complex control logic to manage exception handling, and limits compression performance to 8 or more clock cycles per byte on average.

The symbol matching overhead of large CAM arrays was widely considered too expensive for embedded processor applications for many years. Prior examples of content addressable dictionaries in data compression ICs are limited to modest size [Bunton90, Tran92] in trie partitioned systems. However, the recent introduction of sub-micron processing technologies have changed the economies of integration, warranting a second look at CAM based compression architectures. Moreover, the LZ2 algorithm, and its derivative the DCLZ standard, are particularly well suited to CAM based implementation because all entries to the dictionary are guaranteed to be unique. Therefore, no arbitration of multiple CAM matches is necessary.

This paper describes the architecture of the AHA 3210, a single chip CMOS processor performing DCLZ compatible data compression and decompression at 10MB/s It is implemented in Hewlett-Packard CMOS26B, a three metal layer .8 micron process. The die measures 8.0mm by 9.1mm in size and contains 882,000 transistors.

2 Processor Architecture

A block diagram of the data compression IC is shown in Figure 1. It consists primarily of four modules: the DCLZ Engine, the Ports A and B Interfaces, the Processor Interface, and the Register Control Module. The DCLZ Engine, comprising the majority of the IC die area, was developed by Advanced Hardware Architectures, Inc. and is the primary focus of this paper. The interface and Register Control modules were designed separately by the Hewlett-Packard Co.

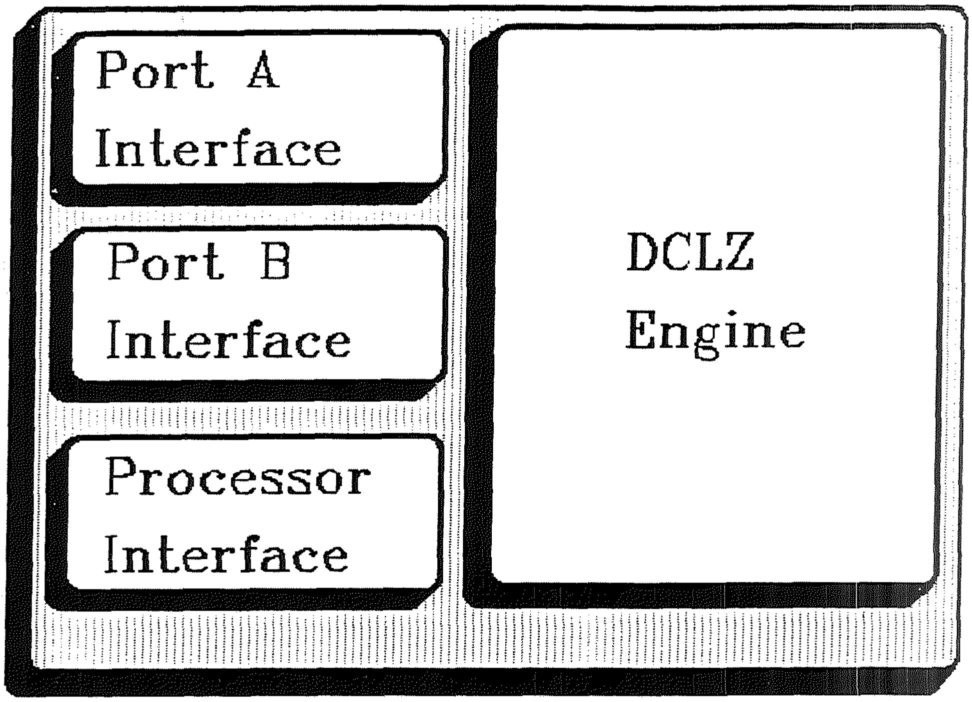


Figure 1: AHA 3210 Block Diagram

The DCLZ Engine, where data compression and decompression occurs, is shown in Figure 2. It is made up of a 4096 word by 20 bit Static CAM module, a Last-In-First-Out RAM module for decompression string reversal, a control logic block called LZControl, and an input/output Formatter module. The CAM and RAM blocks were designed as full custom tiled arrays, while the control logic block was implemented in standard-cells using the U.C. Berkeley OCT synthesis tools.

Using a three metal layer, 0.8 micron CMOS process, the CAM module was implemented in an area of 26 square millimeters for the current processor, operating at a clock rate of 20MHz. The basic CAM cell, shown in figure 3, consists of a static CMOS RAM circuit and a comparator which discharges a precharged WordMatch line when the value of the Column lines differs from that stored in the RAM cell. Match lines for each row and column in the CAM array are derived from the WordMatch signals from which the match address is encoded. To avoid precharge-race conditions, both column lines must be precharged low prior to a search. Data throughput is limited to 10MB/s by the Port Interface Modules. The DCLZ Engine module is capable of considerably greater performance.

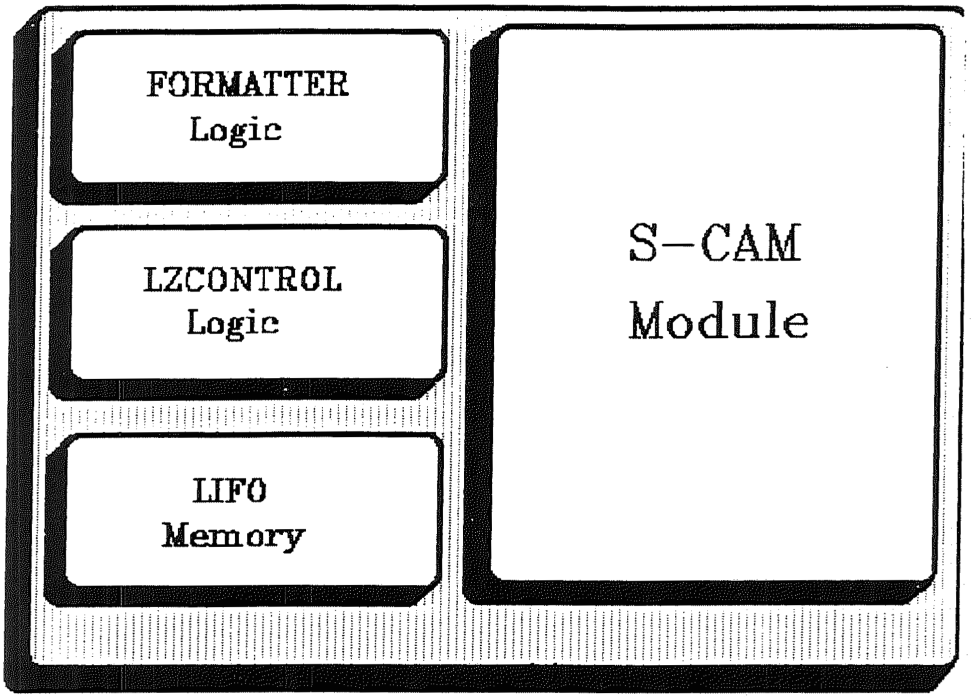


Figure 2: DCLZ Engine Module

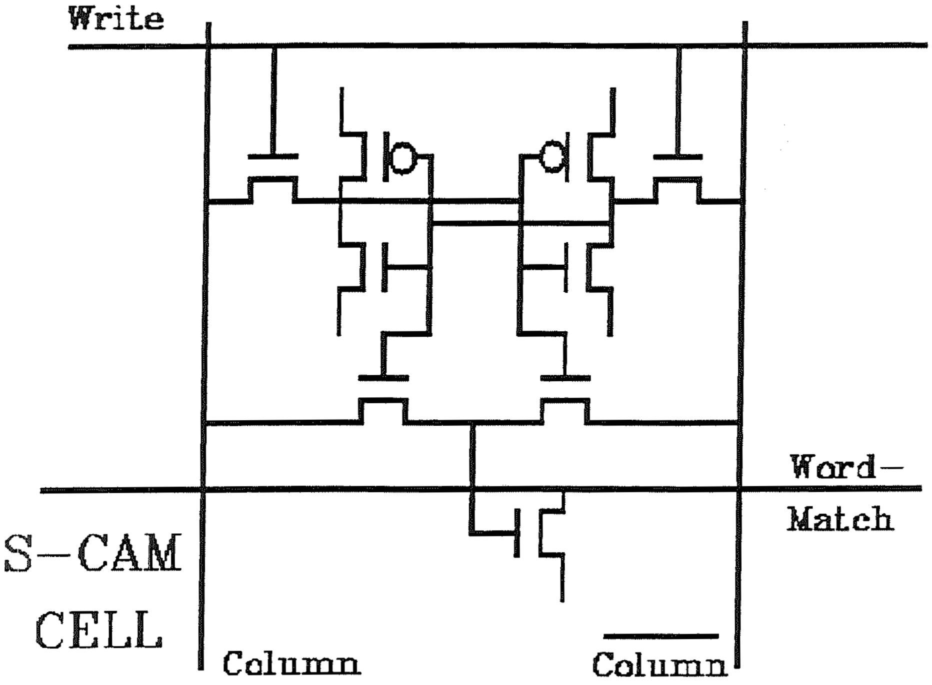


Figure 3: Basic CAM Cell

3 Compression

The LZ2 algorithm in essence represents strings of symbols with codes that are addresses of a string table or dictionary. Each dictionary entry consists of two fields: omega, another code representing the substring containing all but the last symbol, and k, the last string symbol. For a 4K word dictionary, the omega code is 12 bits. The omega register holds the code representing the longest string presently found in the dictionary. Another register, k, holds the next 8-bit symbol from the input data stream.

During a compression operation, the content addressable dictionary, Figure 4, is searched for the 20-bit value of omega-k in a single clock cycle. If it is found, the CAM stores the 12-bit match address as the new value of omega, and the process repeats with the next input symbol, k. If omega-k is not present, it is written to the dictionary at the next available CAM address, omega is output representing the code for the last compressed string, and k is transferred to the omega register for the next search. The CAM write address pointer is provided by the control logic, and is incremented upon each CAM mismatch. Performance is optimized by careful electrical design of the CAM match detection circuits, and by pipelining the omega, k, and address pointer registers to break up critical timing paths.

Next, the 12-bit string codes are truncated to the most significant 9 through 12 bits according to the DCLZ standard and are packed into an 8-bit parallel output port. Power management circuits are employed to suppress row and column line cycling when data is not actively being compressed or decompressed. The CAM module consumes 400mW at a data rate of 10MB/s.

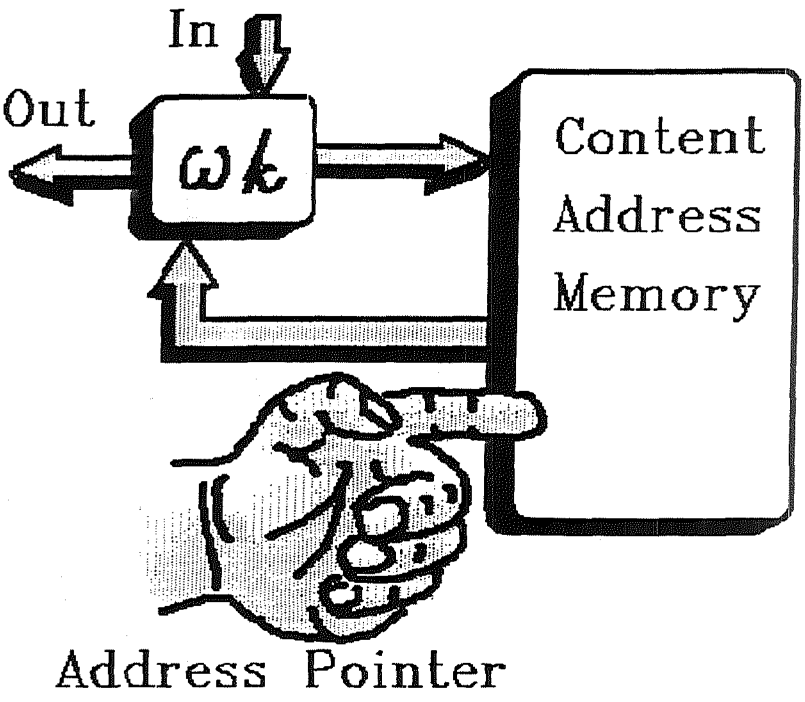


Figure 4: Compression Operation

4 Decompression

The decompression function is a straight-forward implementation of the LZ2 algorithm. The CAM is again used to store the string table. However, in decompression mode its operation is modified somewhat to function as a simple RAM, providing single clock cycle read and write access. The CAM must be switched between compression and decompression mode, since the performance of associative searches requires that the column lines be precharged low, while standard memory reads require that they be pre-charged high.

Since the CAM is capable of providing single clock cycle reads and writes, the decompressor is able to produce one output character per clock cycle. However, a challenge of the LZ2 decompression algorithm is the fact that it decompresses input codes into symbol strings in reverse order. If a simple LIFO or stack is used to reverse each string, it would take two clock cycles per character to produce the output to the interface FIFOs. Moreover, the decompressor function would be stalled whenever the stack data is popped. In other words the processing of short strings following a long one would be stalled while the long string is being reversed by the stack.

The use of multiple stacks alleviates this situation. The string reversal mechanism concurrently pushes characters on one stack and pop characters off a previously used stack. A circular queue structure in a separate two-port RAM module is used to provide a sufficient number of stacks to insure that string reversal matches the rate of symbol decompression in the CAM module.

5 Design Methodology

The DCLZ Engine was designed using a combination of CAD tools from the University of California at Berkeley, Hewlett-Packard Co., and the University of Idaho Microelectronics Research Center. The LZControl and Formatter modules were synthesized in standard-cells from hardware description language (HDL) models using the Berkeley OCT tool suite. This was the first known commercial use of the OCT tools. The full custom SCAM module was designed using proprietary artwork and schematic capture software from Hewlett-Packard. All modules were simulated using the NOVA simulator from the UI-MRC. It was necessary to construct a number of translation utilities to integrate three tool groups, as illustrated in Figure 5.

6 Summary

This new data compression processor demonstrates the practicality of a large content-addressable dictionary where high performance is desired. It is conservatively designed to encode or decode at a sustained rate of 10 MBytes per second and beyond. The component cost, board area, and timing overhead of external memory components are eliminated, as well as the control complexity and exception handling associated with hash coding schemes.

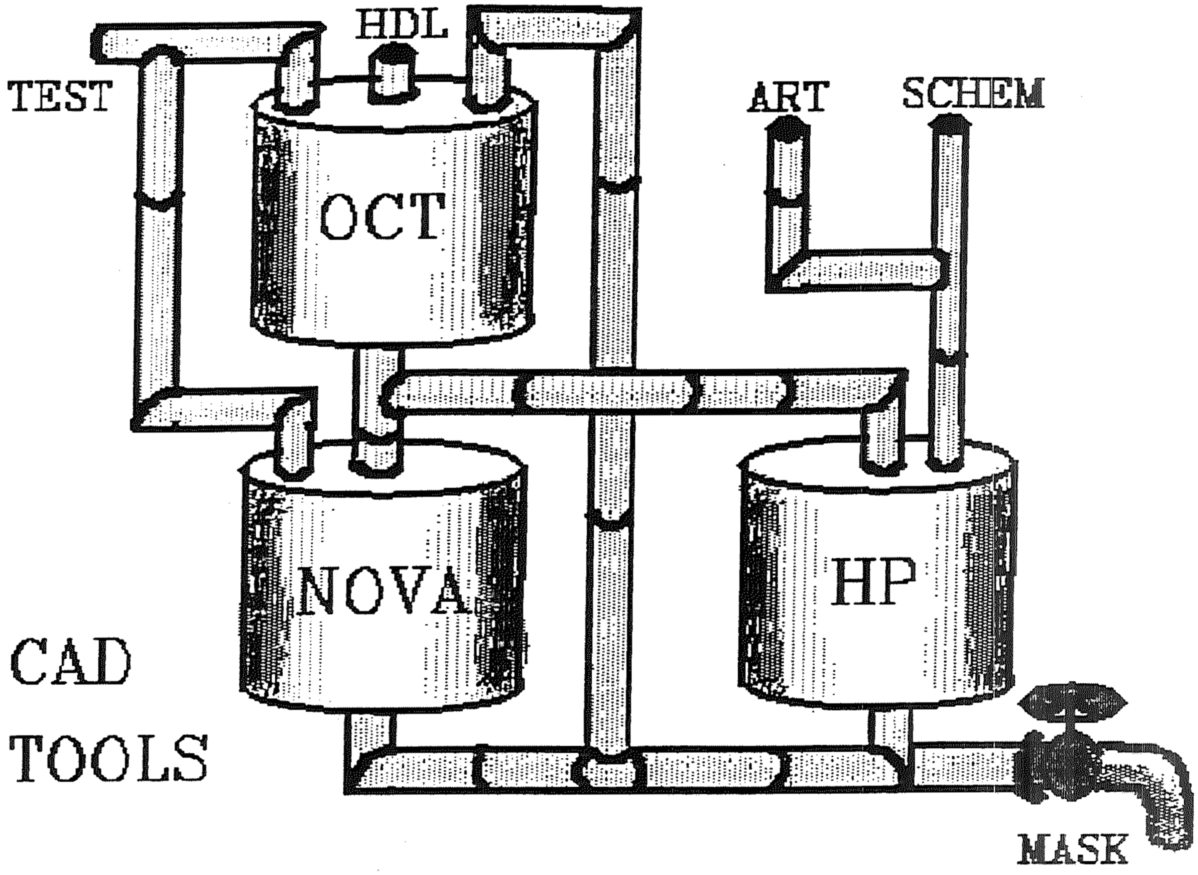


Figure 5: CAD Tool Path

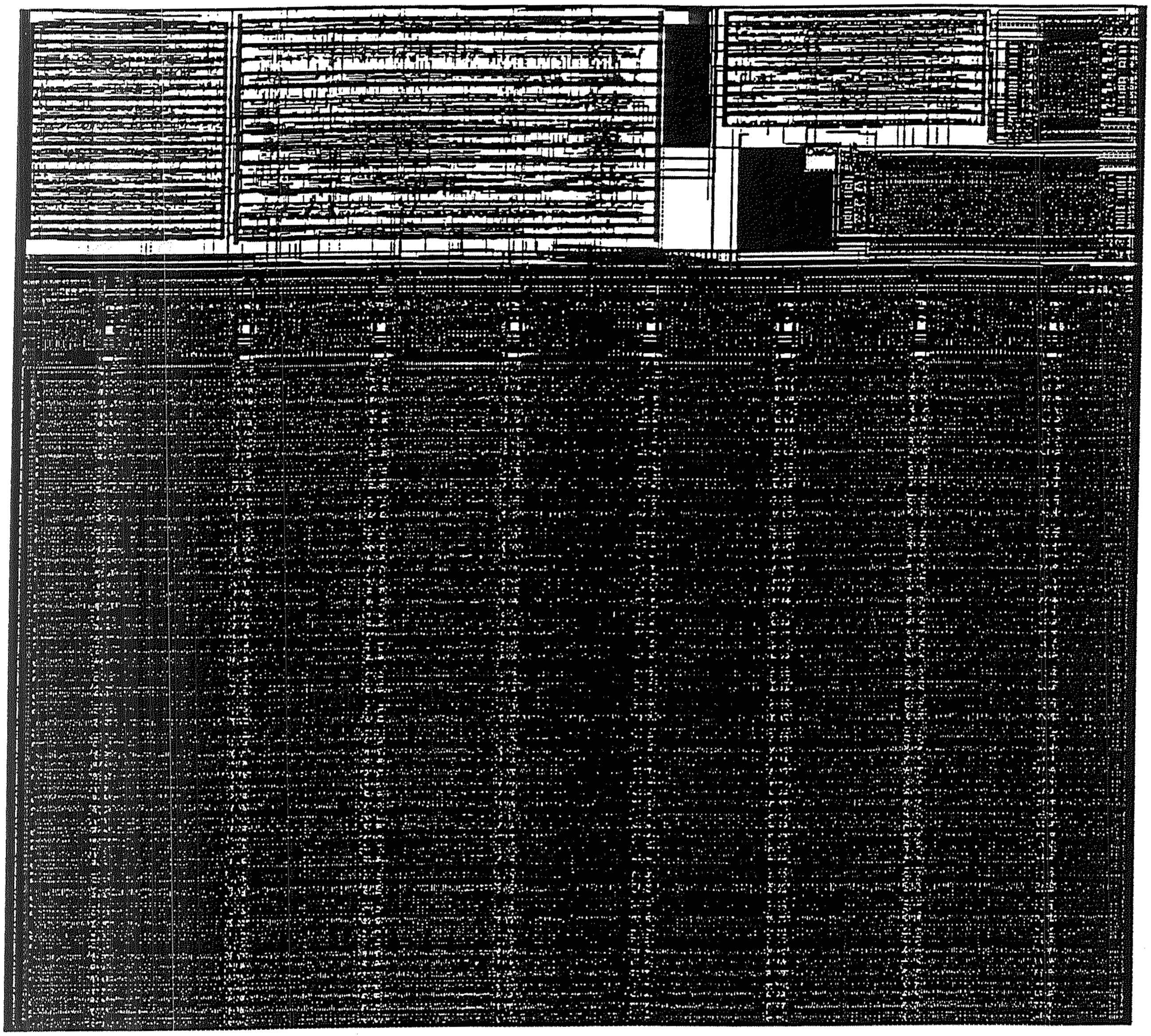


Figure 6: DCLZ Engine Artwork

References

[Bunton90] . Bunton and G. Borreillo, "Practical Dictionary Management for Hardware Data Compression," Technical Correspondence ID# 02-90-09, Washington Research Foundation, Seattle WA, 1990.

[QIC91] IC Development Standard, DCLZ Data Compression Format, QIC-130, Rev A, Quarter-Inch Cartridge Drive Standards, Inc., Santa Barbara, CA, April 1991.

[Tran92] . Tran, B. Wei, M. Desai, "A Single-Chip Data Compression/Decompression Processor," Proceedings Data Compression Conference, Snowbird UT, March, 1992, p. 433.

[Welch84] . Welch, "A Technique for High-Performance Data Compression," IEEE Computer, June 1984.

[Ziv78] . Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," IEEE Trans on Inf Theory, VOL IT-24, No. 5, Sept 1978.

High-Speed Viterbi Decoding with Overlapping Code Sequences

Michael D. Ross and William P. Osborne
Manuel Lujan Jr. Center for Space Telemetering and Telecommunications Systems
New Mexico State University
Las Cruces NM 88001

Abstract - **The Viterbi Algorithm for decoding convolutional codes and Trellis Coded Modulation is suited to VLSI implementation but contains a feedback loop which limits the speed of pipelined architecture. The feedback loop is circumvented by decoding independent sequences simultaneously, resulting in a 5-9 fold speed-up with a two-fold hardware expansion.**

1 Introduction

Since 1971, when the Viterbi Algorithm [1] was introduced as the optimal method of decoding convolutional codes, improvements in circuit technology have steadily increased its speed and practicality. Recently, there has been an interest in developing new architectures to increase the speed at which the algorithm can be performed. As a brute force approach, one might build systems in parallel, increasing the throughput while allowing a proportional increase in hardware volume; however, it would be preferable to improve on parallelism, obtaining an increase in throughput with a less than proportional increase in hardware volume. This paper presents an architecture which accomplishes this goal.

Pipelining and parallelism are well-known techniques for increasing the throughput of arithmetical operations at the logic gate level, especially in VLSI implementation. The Viterbi algorithm consists of three major parts: branch metric calculation, add-compare-select (ACS), and survivor path memory. Of these three, branch metric calculation and survivor path memory are highly adaptable to pipelining, but the add-compare-select function introduces feedback which precludes the pipelining of the entire Viterbi algorithm. Fettweis and Meier [2] approach this problem by combining multiple stages of the Viterbi algorithm trellis, and introducing massive parallelism into their decoder.

This paper presents an alternative approach, the construction of a single decoder to decode multiple independent code sequences simultaneously. Given Ω pipeline stages between the input and output of the add-compare-select unit, it is possible to design a decoder which processes Ω independent code sequences, using less hardware than would be required to build Ω complete decoders.

While the independent sequence architecture could be applied to virtually any convolutional code, it has in fact been used in the design of the complete logic for a decoder which decodes the 16-state rate 2/3 8-PSK Ungerboeck code. Ungerboeck codes are described in references [2] and [3]. All of the logic for this 8-PSK Decoder has been verified by computer simulation.

2 The Structure of the Viterbi Algorithm

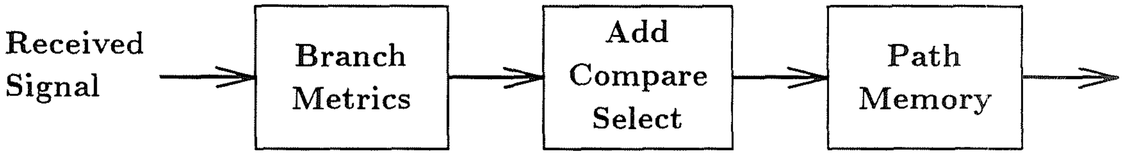


Figure 1: Top level diagram of a Viterbi Decoder

A top level diagram for a Viterbi decoder, consisting of three major parts, is shown in Figure 1. The branch metric calculator calculates a metric for each symbol in the symbol set, relative to the received symbol. The received signal could be hard or soft-decision binary, or a signal vector from a QAM signal set. The Add-Compare-Select unit maintains a state metric to each path, and revises the state metrics by adding in the branch metrics, and selecting the minimum metric at each state. The path memory unit maintains a running history of the most likely path to each state, accepting and rejecting paths according to the decisions made by the add-compare-select unit.

The three major parts of the Viterbi decoder are interrelated in the sense that architectural decisions made for one unit will dictate constraints in the architecture of the other units. The efficiency of the branch metric calculator may be maximized by pipelining, however, the add-compare-select unit requires that the results of the previous iteration (the previous state metrics) be available before the next iteration can begin. This paper shows how a pipelined add-compare-select unit can process multiple independent code sequences simultaneously, although it will not speed up the rate at which any individual sequence can be processed.

3 Add-Compare-Select Logic Design

In Section 2, the operation of the Viterbi algorithm was illustrated using a four-state trellis having two branches converging into each node. The rate 2/3, 16-state Ungerboeck code requires that 4 branches converge to each node, which greatly increases the complexity of the ACS unit, as there are for quantities to be compared, rather than only two. Of course, a four way comparison could be implemented by two initial two-way comparisons, followed by a final two-way comparison, however, the number of pipelined stages in the add-compare-select operation critically affects the overall design of the Viterbi decoder, therefore, it was considered preferable to implement the four-way comparison as six parallel two-way comparisons. The cell illustrated in Figure 2 performs a four-way add-compare-select operation, and stores the state metric for one node of the trellis. To decode a 16-state code, sixteen such units would be required. The cell accepts 7-bit branch metrics and generates 10-bit state metrics. Using no more than three gates (NAND, NOR, or the equivalent) in any pipelined stage, the complete operation can be completed in 9 stages, ie: the factor Ω, mentioned in Section 1, is 9. The method of Hekstra [5] uses the concept of metric roll-over, as opposed to metric overflow. Knowing the maximum possible difference between any two state metrics,

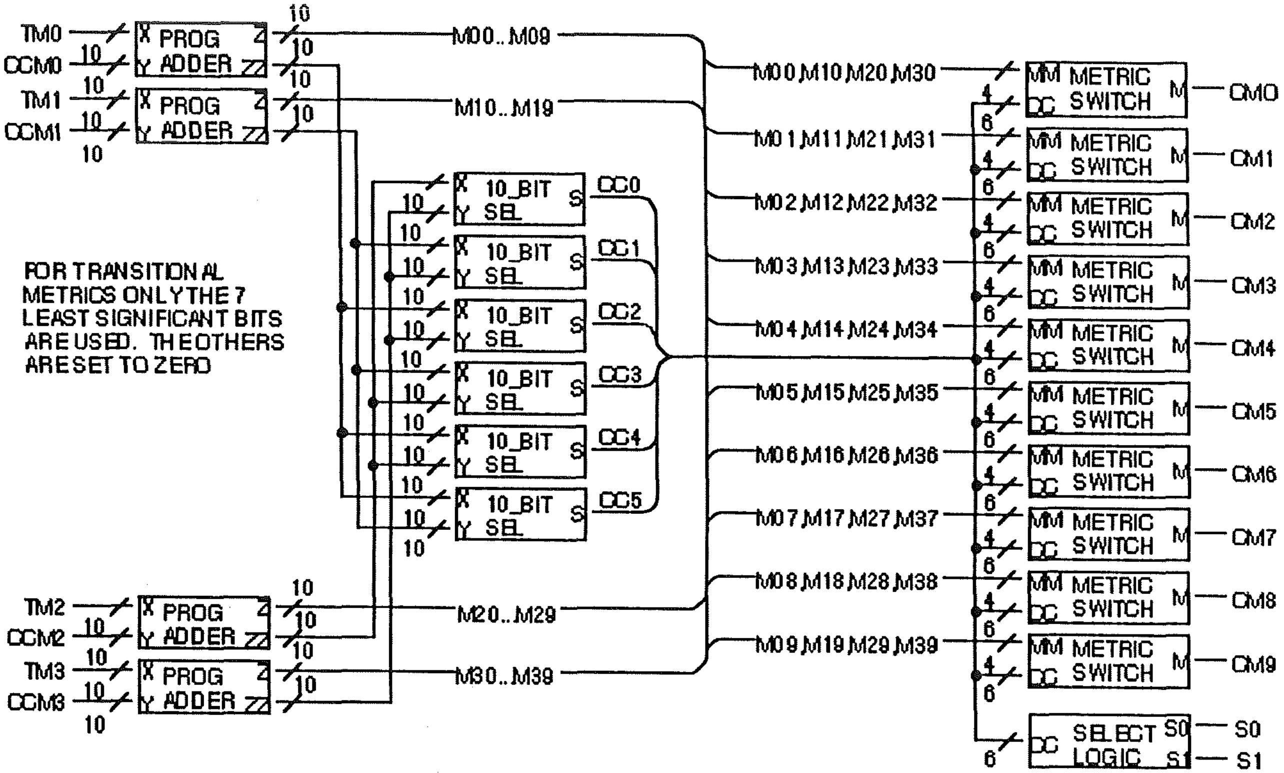


Figure 4.23. ACS Cell.

Figure 2: Circuit to perform a four-way Add-Compare Select Operation

it is possible to correctly compare the rolled over metrics, and it is never necessary to reset the metric registers.

The progressive adder (PROG ADDER) cell (we are still referring to Figure 2) adds a transitional metric (TM) to a previous cumulative metric (CCM), producing a new path metric. The progressive adder will add two 10-bit numbers, although only a 7-bit quantity is required for the transitional metric. The addition occurs in 5-stages of two bits each. On the output Z, 10-bits associated with a new metric are presented in series, whereas on the output ZZ, each pair of bits in the sum is presented to the comparator/selector (10_BIT CELL) as soon as it is available. The comparator is designed to take advantage of this, which reduces the total number of stages required to complete the add-compare select operation.

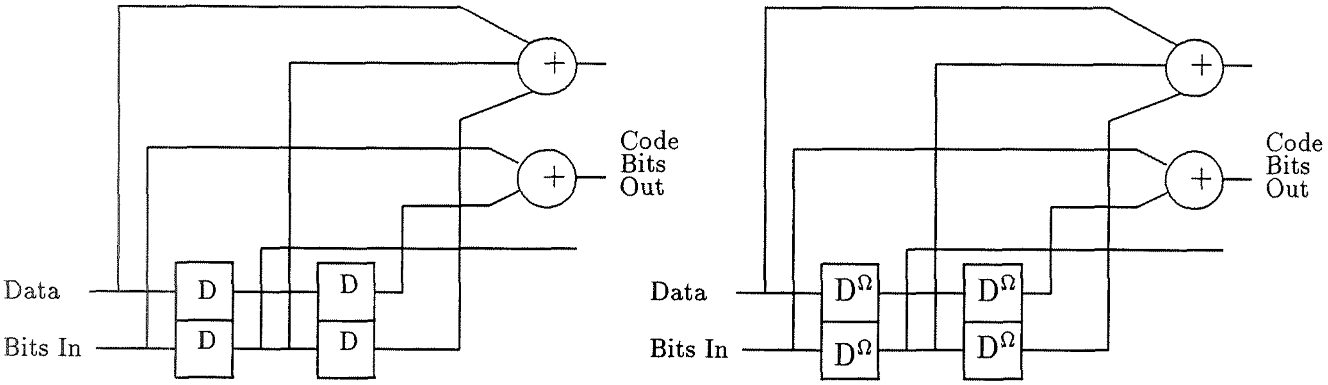
Four progressive adders generate metrics for four contending paths. Each of six comparators compares two metrics and outputs the result of the comparison (CC). The metric switch is a specialized multiplexer, which will select the appropriate metric on the basis of the results of the six comparisons. Each switch handles one bit of the final 10-bit metric. therefore 10 such switches are required. The final result of all this is a 10-bit new cumulative metric (CM) and two bits, S0 and S1, indicating which of the four trellis branches is selected at the given node.

4 The ACS Feedback Loop and the Super-Trellis Solution

The add-compare-select loop introduces feedback into the Viterbi algorithm and has therefore been seen to limit the extent to which the process can be sped up by pipelining [2]. The metric calculation, which feeds the add-compare-select unit depends only on current data, therefore, the metric calculator may be pipelined and symbol metrics can be produced at the same rate at which received symbols are clocked into the decoder. By contrast, the cumulative metric, by definition, depends on the result of the previous calculation. A subsequent symbol from a given convolutionally encoded sequence cannot be processed until the calculation of the previous symbol is complete. This implies that the rate at which symbols can be processed is limited by the speed at which the ACS operation can be completed.

An approach to Viterbi decoder architecture which has received some attention in recent literature, is the combined trellis stage, or "super-trellis" approach of Fettweis and Meyr [2]. This approach is termed as a linear scale solution because it offers an M-fold increase in speed in return for an M-fold increase in the volume of hardware in the ACS unit. It is explained later that adopting the super-trellis architecture in place of the simple trellis architecture initially multiplies the volume of the ACS unit by the number of states in the trellis code, then the linear scale solution is obtained thereafter. Fettweis and Meyr have applied their architecture to a 4-state binary code. In this paper, a different approach is presented for the 16-state Ungerboeck decoder, because the super-trellis approach introduces considerable complexity, which is compounded for codes of greater number of states.

The combined trellis stage approach consists of forming a super trellis stage, which shows branches for all of the state transitions which the encoder can make in M steps of operation, unlike a standard trellis stage, which shows only the transitions which the encoder can make



a) Convolutional encoder for sixteen-state Ungerboeck code.

b) Convolutional encoder modified to generate independent, overlapping code sequences.

Figure 3:

in one step. The authors of the combined trellis architecture use the terminology, 1-step trellis to apply to the standard trellis and M-step trellis to apply to the super trellis. Presumably, if larger hardware can be built to perform the ACS operation for the super trellis stage, the data rate could be increased, since an M-step trellis represents an M-fold increase in data, while the ACS operation for the super trellis should require only slightly longer than the ACS operation for the simple trellis. To apply this approach, metrics must be calculated for the branches of the combined trellis stage, each of which now consists of M symbols. Also, the operation of combining the trellis stages increases the number of branches which c onnect into each node, and leads to the formation of parallel branches, multiple branches which connect the same pair of states. If the parallel branches are eliminated prior to the super trellis ACS operation, the number of branches converging into a single node is limited to the number of states. Therefore, the difficulty of applying the super trellis approach grows substantially with the number of states of the code.

5 The Overlapping Code Sequence Approach

The speed limitation introduced by the ACS loop may also be circumvented by making a small modification to the coding standard, which works as follows. Although the add-compare-select unit cannot process a symbol from a convolutional code sequence until the processing of the previous symbol from the same sequence is complete, the pipelined hardware can begin the processing of a symbol from another independent code sequences on the immediately following clock cycles. Thus if there are Ω pipeline stages in the ACS calculator, the decoder will process Ω independent code sequences concurrently. Each pipeline stage of the ACS unit will hold a calculation in progress associated with a symbol from a different sequence.

The parameter Ω will be referred to as the overlap factor. Figure 3a shows the standard convolutional encoder for the 16-state Ungerboeck code, while Figure 3b shows the modified convolutional encoder. The only difference is that the modification replaces the single delay units (D) with multiple delay units (D^{Ω}), which delay the input by Ω clock cycles, as opposed to only 1 clock cycle. The effect of this is that the modified encoder is actually encoding the data onto Ω independent code sequences. The independent sequences overlap each other in rotation, while consecutive symbols from the same sequence are separated by Ω clock cycles. Metrics from the metric calculation unit arrive at the ACS unit according to the same pattern, which is exactly what is needed to make the decoder function properly. The metrics associated with a symbol arrive at the ACS unit just as the ACS calculation associated with the previous symbol of the same sequence is complete. Meanwhile, the same hardware is being used to process the other independent sequences.

The path memory unit must also be modified to accommodate the overlapping sequence architecture. Note that the basic cell of a generic path memory consists of a multiplexer followed by a latch as shown in Figure 4a. The modification required is exactly the same as the modification introduced to the convolutional encoder. The single latch is replaced by an Ω stage delay as shown in Figure 4b.

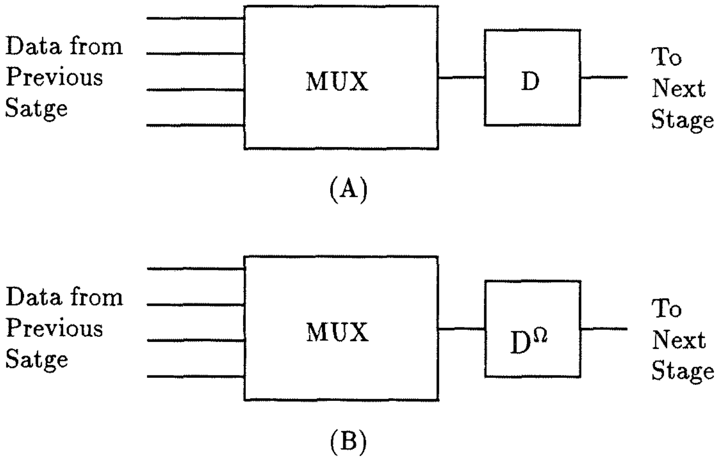


Figure 4: A) path memory cell. B) modified path memory cell

The use of multiple independent coding sequences allows a speedup in operation with a less than proportional expansion in hardware. By allowing only a single code sequence, pipelining the ACS operation does not change the fact that symbols can only be processed at the rate at which the ACS operation can be performed. Although the exact speeds involved depend on the technology employed and the specific structure of the ACS circuitry, it stands to reason that if latches can be installed at the approximate half-way points in all of the critical paths of the ACS circuitry ($\Omega = 2$), the data rate of the overall system can be approximately doubled with only a slight increase in the hardware of the ACS unit. Certainly, a twofold increase in speed has been obtained without a twofold increase in ACS hardware. The effect of this strategy on the memory hardware is that where there was formerly a latch and a MUX, there are now two latches and a MUX. Since a latch consists of two logic gates, and a (two-way) MUX consis ts of three, the hardware in the path memory expands by approximately 7/5, while increasing the speed of the system by a factor of two. In the case of a trellis with four branches expanding into a node, the benefits of this design approach are comparable. For Ω other than 2, it is a matter of simple arithmetic that the expansion in hardware is less than the increase in speed; that is, up to the point where the ACS unit is maximally pipelined. It is, however, desirable to minimize the length of the ACS path, since this ultimately drives the size of the memory. In the codec presented here, with the rule of no more than three logic gates between any pair of latches, the ACS operation was found to require 9 clock cycles, therefore an overlap factor of Ω=9 was employed.

6 Architectural Impact

The impact of the overlapped independent sequence (OIS) approach on the architecture of the decoder depends on the particular convolutional code to be implemented. In comparing the independent sequence architecture to conventional architecture, it should be kept in mind that both require the propagation delay from input to output of the add-compare-select circuit to be minimized. The structure of this circuit will be approximately the same; however, the ACS unit in the independent sequence architecture will be pipelined to a greater degree. The greater complexity of a 4-way ACS over a 2-way ACS is a factor which has roughly the same effect in both architectures.

The metric calculation circuitry, which has been given very little attention in this paper, has essentially the same structure in both architectures, except that again, the circuit for the independent sequence architecture will have more pipelining, due to the fact that the OIS approach gives the ACS unit has a means of using the greater throughput.

Most of the expansion resulting from the application of the overlapping sequence architecture occurs in the path memory. Every Viterbi decoder must record the maximum likelihood path to every state, for some length, referred to as the decoder depth, whose recommended length depends on the code. The required number of memory cells of the kind shown in Figure 4, is equal to the number of states, times the decoder depth, times the number of information bits per branch. The number of information per branch is 1 for a code with two converging branches per node, 2 for a code with four converging branches. Comparison of Figures 4a and 4b shows the impact of the OIS architecture on this part of the decoder.

7 Conclusion

While pipelining can increase the throughput of logical and arithmetic operations, it cannot increase the throughput of the Viterbi decoding of a single convolutionally encoded sequence, due to the feed-back required in the add-compare-select operation. An increase in data throughput can be obtained by modifying the convolutional encoder to generate Ω independent code sequences, as shown in Figure 3, where Ω is the number of pipeline stages in the add-compare-select unit. The path memory of the Viterbi decoder must also be modified to handle the multiple sequences.

The overlapping sequence architecture was applied to the design of the complete logic of a Viterbi decoder for the rate 2/3, 8-PSK 16-state Ungerboeck code, with a decoder depth (or path history length) of 40. Using a computer simulation package it was verified that the bit error rates for this machine are as should be expected for the employed code. Using the considerations discussed in section 6, the amount of logic required for the overlapping

	LOGICAL OPERATIONS	
	OVERLAP	CONVENTIONAL
METRIC CALCULATION	7,026	3,386
ADD-COMPARE-SELECT	37,504	22,400
PATH MEMORY	29,440	8,960

VOLUME INCREASE: 2.13 SPEED INCREASE: 5-9

Table 1: Camparison of of Overlapped-Independent-Sequence Architecture with conventional architecture

sequence architecture, as compared to conventional architecture was determined, and is shown in Table 1. Here, a "logical operation" refers to 1 NAND gate, "NOR" gate or inverter. In principle, the OIS could increase the throughput of the decoder by a factor of Ω, but allowing for non-idealities in timing, a more realistic estimate could be as low as 5, for $\Omega = 9$. However, from Table 1 we can see that the amount of required logic increased by a factor of 2.13, showing that a better than linear scale architecture has been obtained.

References

[1] Viterbi, A.J., "Convolutional Codes and their performance in Communication Systems," IEEE Transactions on Communication Technology, vol. CT-19, pp. 751-771, October 1971.

[2] G. Fettweis and H. Meyr, "Parallel Viterbi Algorithm Implementation: Breaking the ACS-Bottleneck, "IEEE Transactions on Communication," vol. COM-37, pp. 785-790, Aug 1989.

[3] Ungerboeck, Gottfried, "Trellis Coded Modulation with Redundant Signal Sets, Part I: Introduction," IEEE Communications Magazine, Vol. 25, No. 2, pp. 5-11, February 1987.

[4] Ungerboeck, Gottfried, "Trellis Coded Modulation with Redundant Signal Sets, Part II: State of the Art," IEEE Communications Magazine, Vol. 25, No. 2, pp. 12-21, February 1987.

[5] Hekstra, A.P. "An Alternative to Metric Rescaling in Viterbi Decoders," IEEE Transactions on Communications, vol. 37, pp. 1220-1222, Nov. 1989.

omit

Included Papers

Arithmetic in Large GF(2^n)

Kelly Cameron

NASA Space Engineering Research Center
for VLSI System Design
University of New Mexico

Abstract– **The decoding of Reed Solomon (BCH) codes usually requires large numbers of calculations using GF(2^n) arithmetic. Though efficient algorithms and corresponding circuits for performing basic Galois field arithmetic are known, many of these techniques either become very slow or else require an inordinate amount of circuitry to implement when the size of the Galois field becomes much larger than GF(2^8). Consequently, most currently available Reed-Solomon decoders are built using small fields, such as GF(2^8) or GF(2^{10}), even though significant coding efficiencies could often be obtained if larger symbol sizes, such as GF(2^{16}) or GF(2^{32}), were used. Algorithms for performing the basic arithmetic required to decode Reed-Solomon codes have been developed explicitly for use in these large fields. They are discussed in detail.**

1 Introduction

The use of Reed-Solomon (BCH) codes in various error control applications is becoming widespread. Uses range from such relatively low performance applications as Compact Disks and CD ROMS to high performance tape drives. Reed-Solomon codes are used to ensure data integrity in magnetic and optical disk drive system. Several proposed standards for HDTV (High Definition Television) [19, 11, 21, 20, 29] also call for the use of Reed-Solomon codes to improve performance under poor signal conditions. High performance (15Mbyte/s) Reed-Solomon codecs are commercially available [1] and the world-wide standard for near earth space telemetry transmissions (CCSDS) [5] also employs such codes.

Reed-Solomon codes are algebraic block codes. They are defined in terms of finite field arithmetic, [2, 3, 6, 10]. Both the information and the redundancy portions of such codes are viewed as consisting of elements taken from some particular Galois field. A Galois field is commonly identified by the number of elements which it contains. Since the number of elements contained in a Galois field is always equal to a prime number raise to a positive integer power, the notation, GF(q^m), where q is a prime number and m is an integer is commonly used to refer to the finite field containing q^m elements. Though Reed-Solomon codes may be defined over any Galois field, a common choice is the use of GF(2^8). This is usually a convenient choice since each symbol in this field may be viewed as an eight bit byte.

Though efficient algorithms and corresponding circuits for performing basic Galois field arithmetic are known, many of these techniques either become very slow or else require an inordinate amount of circuitry to implement when the size of the Galois field becomes much larger than GF(2^8). Consequently, most available Reed-Solomon decoders are built using be obtained if larger symbol sizes, such as GF(2^{16}) or GF(2^{32}), were used. Algorithms for performing the basic arithmetic required to decode Reed-Solomon codes have been developed explicitly for use in these large fields.

2 Traditional Implementations

Addition in GF(2^n) is implemented quite easily, regardless of the size of the field. It corresponds to bitwise xor of the elements to be added [7, 4] i.e. addition modulo 2.

2.1 Multiplication

Multiplication is not so simple. The multiplicative structure of a finite field can be determined from the primitive [4] polynomial used to create the normal representation [15] of the field. Utilizing a primitive polynomial, it is a simple matter to generate a logarithm table, which can be used to perform multiplication. The manner in which such log tables are created is well known and is widely described in the literature [14].

Given such a log table, any two non-zero elements may be multiplied by taking:

$$C = AB = \log^{-1}[\log(A) + \log(B) \pmod{2^n - 1}] \tag{1}$$

where $\log^{-1}$ is the inverse of the log function. Though this approach to multiplication works quite well in software, as long as the size of the field isn't too large, it tends to be bulky when implemented in hardware, even for fields as small as GF(2^8). The inherent problem is that the amount of data stored in the tables grows faster than exponentially with the number of bits in the field. $2n2^n$ bits are required for the log and $\log^{-1}$ tables of GF(2^n).

Consequently, it is desirable to construct circuits or algorithms which actually calculate the product of two numbers without the aid of lookup tables. Several successful approaches have been pursued in this regard. The first circuit devised for multiplying two general field elements is attributed to Elwyn Berlekamp [18]. It is a bit serial circuit which requires n clock cycles to perform a multiplication.

The first parallel implementation of a finite field multiplier known to the author was developed by Yeh, Massey and Omura [28, 16, 25]. Though these implementations manage to make the circuitry for calculating each bit of the product identical, with simply a permutation of the inputs to the circuit, the implementation of the actual multiplier circuit is quite irregular and not necessarily easily implemented in VLSI. Due to its irregularity, the area required to implement this type of adder as a function of n is not easily ascertained, nor are the propagation delay characteristics of the circuit easy to determine.

Another implementation of a parallel finite field multiplier is described in [17]. This implementation is very closely related to a traditional integer multiplier, with the terms corresponding to powers of α greater than or equal to n being equated to sums of lower powers of α, in accordance with the defining primitive polynomial. Such a circuit is shown in figure 1 for the field defined by $p(x) = x^4 + x + 1$.

This implementation, being regular in structure, is easily analyzed, both in terms of circuit area required to implement, as well as speed performance. The area required to

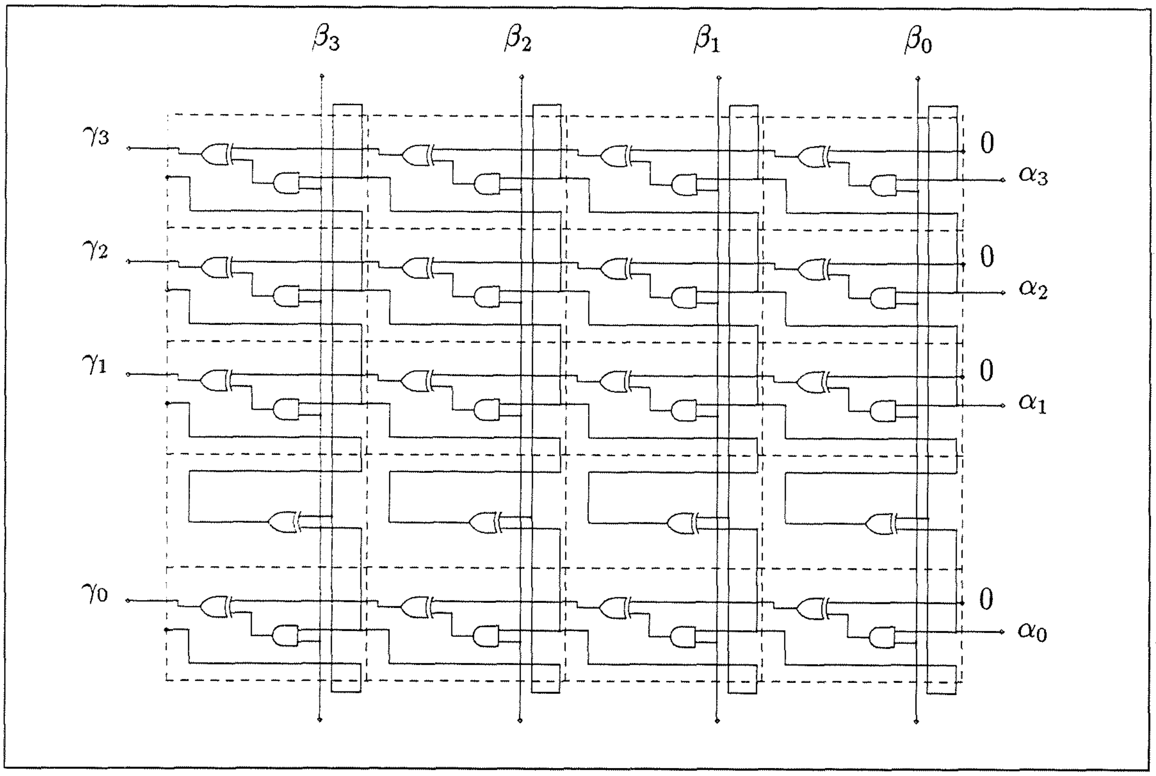


Figure 1: Multiplication Circuit for GF(2^4)

implement this design is proportional to n^2. This is the best implementation, area-wise, known to the author. Propagation delay grows proportional to n.

It should be mentioned that multiplying a finite field element by a constant is much simpler than general multiplication of two arbitrary field elements [7]. It corresponds to a linear operation– a GF(2) n by n matrix multiplication by an n bit vector. Such operations are easily and efficiently performed [17]. Though not as obvious, both the square and the square root operations are linear operators in any field of characteristic 2 [25]. They can be computed with similar matrix operations.

2.2 Division

The task of dividing to numbers in a finite field is not straightforward. The usual algorithm for dividing integers does not generalize to a finite field. Division is often accomplished with log and $\log^{-1}$ tables or inverse tables. These methods permit inverses to be found quite rapidly, but suffer from the same problem that log table multiplication has, namely the amount of information required to implement these schemes grows proportional to $n2^n$. Be that as it may, inverse table lookup may very well be the method of choice for smaller fields– as of this writing 8 to 10 bit fields for hardware implementations.

A well known method for calculating inverses in a finite field exists [9]. It follows directly from the cyclic structure of such a field that the inverse of a field element can be obtained directly from exponentiation. To be more precise:

$$a^{-1} = a^{2^n-2} \tag{2}$$

A simple analysis shows that this operation can be accomplished with $2n-3$ multiplications.

3 Constituent Subfields

It may be observed that a basic characteristic shared by all of these "traditional" procedures for performing arithmetic is that they operate by combining operations defined in GF(2) arithmetic. As will be shown, this may not always be an optimal approach.

Though the multiplicative structure of a finite field is typically defined in terms of a primitive polynomial, there are actually several alternate, though closely related, ways of doing so [14] (p.62). The approach investigated here involves yet another method for determining the multiplicative structure of the field. The multiplicative structure of the field will be defined in terms of multiple primitive polynomials. The method may only be applied to finite fields where n is a composite number, however. The procedure is as follows:

Let n, the number of bits in the field, be a composite number such that $\prod p_i = n$, where p_i any set of factors of n. The p_i values may be repeated, if necessary. The multiplicative structure of the field may be determined by i primitive polynomials, $\mathcal{F}_i$, where $\deg(\mathcal{F}_i) = p_i$. Though one typically thinks of GF(2^n) as the n^{th} extension field of GF(2), it may also be viewed it as the p_i^{th} extension field of some subfield of GF(2^n).

In particular, let $\mathcal{F}_1$, a primitive polynomial over GF(2), define a representation of the p_1^{th} extension field of GF(2). Now let $\mathcal{F}_2$, a primitive polynomial in GF(2^{p_1}), define a p_2^{th} extension field of GF(2^{p_1}). This procedure may be continued with all factors, p_i, of n, until the desired finite field is constructed.

For example, let $n = 16$, $p_1 = 8$, and $p_2 = 2$. A representation of GF(2^{16}) may be defined in terms of the primitive polynomials, $\mathcal{F}_1 = x^8 + x^5 + x^3 + x + 1$, a primitive polynomial over GF(2), and $\mathcal{F}_2 = x^2 + x + 149$, a primitive polynomial over GF(2^8).

The advantage of constructing the finite field from one or more of its subfields lies in the fact that the structure of these subfields remains accessible to direct manipulation when this type of construction used. Furthermore, operations defined in GF(2^n) may be directly calculated in terms of operations performed in one or more of the constituent subfields.

Though the above discussion refers only to fields of characteristic 2, the same principles may be applied to finite fields of any characteristic.

4 Basic Operations

For the remainder of this discussion, attention will be focused on the implications of representing a finite field, GF(2^n), as a quadratic extension of a smaller field. This, of course limits the discussion to finite fields with n even. It will be demonstrated that the operations of multiplication, inverse, square, square root, and conjugation may be performed in GF(2^n), utilizing only operations from the subfield, GF(2^m).

4.1 Multiplication

Let GF(2^m) be the subfield, and GF(2^n) be a quadratic extension field constructed from GF(2^m), with the primitive polynomial, $\mathcal{F}$,

$$x^2 + x + \beta = 0 \tag{3}$$

where β is an element from GF(2^m). Though the above choice for the form of $\mathcal{F}$ is admittedly arbitrary, it is in no way restrictive. An arbitrary quadratic primitive polynomial, $Ay^2 + By + C = 0$, may be converted to the form specified for $\mathcal{F}$ by the linear transformation $y = Bx/A$.

Now, choosing α to be a primitive root of $\mathcal{F}$, we have the following relationship, which will be used to define the multiplicative structure of the quadratic extension field, GF(2^n).

$$\alpha^2 = \alpha + \beta \tag{4}$$

Now, let α^0 and α^1 be the basis vectors for constructing the quadratic extension field, GF(2^n). Furthermore, let $A + \alpha B$ and $C + \alpha D$ be arbitrary elements from GF(2^n). Performing symbolic multiplication on these quantities yields:

$$(A + \alpha B)(C + \alpha D) = AC + (AD + CB)\alpha + \alpha^2 \tag{5}$$

Applying the identity defined in equation 4, one can re-write the right hand side of equation 5 as follows:

$$(A + \alpha B)(C + \alpha D) = (AC + BD\beta) + (AD + CB + BD)\alpha \tag{6}$$

This forms the definition of multiplication in GF(2^n) in terms of operations taken solely from GF(2^m). This rule is very similar to the manner in which multiplication in the complex plain is defined in terms of real number multiplication and addition. Indeed, from a comparative standpoint, α serves the same purpose in GF(2^n) as $i = \sqrt{-1}$ serves in the complex plain.

It should be noted that for the special case when $B = 0$ and $D = 0$, equation 6 simplifies to: $(A + 0\alpha)(C + 0\alpha) = AC + 0\alpha$. It follows directly from this observation that the subfield GF(2^m) shows up explicitly in this representation of GF(2^n). It corresponds to those elements of GF(2^n) where the α term is equal to zero. It should also be noted that the multiplication of an element from GF(2^n) by an element of GF(2^m) is also easily accomplished:

$$(A + \alpha B)C = AC + \alpha BC \tag{7}$$

4.2 Division and Inverses

An expression for the results of the division of one element by another in GF(2^n) can be developed from the defining equation for multiplication. If we let $(A + \alpha B)(X + \alpha Y) = (C + \alpha D)$, apply equation 6 and solve for X and Y, the following expression can be obtained:

$$\frac{C + \alpha D}{A + \alpha B} = \frac{(AC + BC + BD\beta) + \alpha(AD + BC)}{A^2 + AB + \beta B^2} \tag{8}$$

If one substitutes unity for the numerator of the left hand side of this equation, an expression for the multiplicative inverse of an element in GF(2^n) is obtained which utilizes only operations from the subfield, GF(2^m).

$$\frac{1}{A+\alpha B} = \frac{(A+B)+\alpha B}{A^2+AB+\beta B^2} \tag{9}$$

4.3 Squares and Square Roots

Applying the basis expression for multiplication, equation 6, the square of a number can be seen to be:

$$(A+\alpha B)^2 = (A^2+\beta B^2)+\alpha B^2 \tag{10}$$

Solving for the inverse relationship, yields an expression for the square root of a field element of GF(2^n) in terms of only subfield operations:

$$\sqrt{A+\alpha B} = \sqrt{A+\beta B}+\alpha\sqrt{B} \tag{11}$$

4.4 Conjugates

The calculation of the conjugate Z^* of an arbitrary element in GF(2^n), Z, with respect to GF(2^m) follows directly from equation 10, since $Z^* = Z^{2^m}$ [14]. In particular, if $Z = A+\alpha B$, then m repeated application of equation 10 times yields:

$$Z^* = (A+\alpha B)^{2^m} = \alpha B^{2^m} + A^{2^m} + B^{2^m}\sum_{i=0}^{m-1}\beta^{2^i} \tag{12}$$

This expression can be further simplified if one observes that $A^{2^m} = A$ and $B^{2^m} = B$. This results from the fact that these operations are performed in GF(2^m). Furthermore, the summation of powers of β is equal to the trace of β in GF(2^m) [14]. Now, it is known that the trace of β must equal unity, since equation 3 has no roots in GF(2^m) [18]. Consequently, the expression for the conjugate of A may be written:

$$Z^* = (A+\alpha B)^{2^m} = (A+B)+\alpha B \tag{13}$$

It should be clear that this is a bi-directional mapping and holds for any element in GF(2^n).

4.5 Comments

Though the above discussion was only applied to quadratic extension fields, the results obtained may be recursively applied to smaller and smaller subfields, up to the number of factors of 2 contained in n. It should also be noted that similar equations may be developed for other extension fields than quadratic, though the final relationships will, in general, be more complicated.

5 Computational Efficiency

Computational efficiency is evaluated on the basis of number and type of operation that need to be performed as well as the area (very roughly) required to implement the corresponding circuits.

5.1 Multiplication

The operations represented by the multiplication equation, 6, can be re-arranged so as to require only 3 actual general subfield multiplications, a number of additions, and a couple of constant multiplications. In particular, if: $X = (A + \beta B)$, $Y = (C + D)A$, and $Z = [(1 + \beta)D + C]B$ then:

$$(A + \alpha B)(C + \alpha D) = (X + Y) + \alpha(X + Z) \tag{14}$$

If only the number of general GF(2^m) multiplications required were taken into account when calculating the circuit area to implement such a circuit, one would conclude that the area of the circuit grows proportional to $n^{1.585}$, which is considerably better than the n^2 figure for the "traditional" implementation described in section 2. This, however, is an overly simplistic approach, which does not account for the decrease in regularity– both of circuit formation and data flow, and the not insignificant number of "simple" operations that must also be performed. When these items are taken into account, it is anticipated that the actual area required to implement such a multiplier is approximately the same as required for the "traditional" multiplier.

The propagation delay characteristics of such a multiplier is considerably better than the "traditional" multiplier, however. Recursively appling equation 6 to construct the multiplier, a propagation delay proportional to $\log n$ is attainable. One may also select the defining primitive polynomials in such a manner as to simplify the constant multipliers required. The author utilized an exhaustive computer search to arrive at the choice of primitive polynomials listed in section 3.

5.2 Inverse

A straightforward analysis of equation 9 indicates that this expression for the inverse of a field element can be performed using $2n - 3$ GF(2^n) multiplications. Wang [25] proposes a solution whereby $n - 1$ of these general multiplications may be replaced by a simple linear transform.

The expression for the calculation of the inverse of a field element based upon the structure of the quadratic extension field, equation 9 can be calculated using only order $\log n$ operations, all of which are taken, not from GF(2^n), but rather from a subfield. It should be noted, however, that the number of operations at each recursive step is larger than required for the traditional, linear approach. For sufficiently large n, the order $\log n$ algorithm will always be faster, but at what point it becomes faster is a function of the relative cost of the different constituent operations.

5.3 Square and Square Root

Using the "traditional" matrix approach described in section 2, both the square and square root operations can be performed with an nxn by n matrix multiplication. This is, of course, an n^2 process. A typical hardware implementation [17], however requires n^2 area and operates in linear time. The same may be said of the general multiplier, which can, alternatively, be used to implement the square function.

The asymptotic complexity of the quadratic extension field equation is also n^2, when viewed as consisting of GF(2) operations. The time complexity to implement a purely recursive solution to these equations is also linear with respect to n. Significant execution speed advantages, at least for the calculation of the square root function, is possible in some instances, however, if equation 11 is combined with table lookup, in some smaller sub-field.

6 Cube Root

The computation of cube roots in GF(2^n) utilizing only operations from the quadratic subfield is much more involved. A procedure will be briefly described, mostly for completeness sake. First, an expression for the cube of a number is written:

$$(x\alpha + y)^3 = [(\beta + 1)x^3 + xy^2 + x^2y]\alpha + (\beta x^2 y + y^3 + \beta x^3) = (c\alpha + d) \tag{15}$$

Separating the unity and α terms into separate equations, solving the equations for x, and then making the substitutions: $u = d/c$ and $z = x^3/c$, yields the following equation, which can be solved for z by any of a number of different methods [14, 12].

$$z^3 + z^2 + (u^2 + u + \beta + 1)z + 1 = 0 \tag{16}$$

Given values for x, corresponding values of y can be found by solving the following equation, which is constructed by equating the α terms of equation 15.

$$xy^2 + x^2y + (\beta + 1)x^3 + c = 0 \tag{17}$$

These equations typically generate spurious solutions. The correct solutions may be determined by substituting back into equation 15.

7 Logarithms

This discussion will be concluded with a brief discussion on the subject of the computation of discrete logarithms. Being able to perform such calculations quickly and efficiently would be of great advantage in performing finite field arithmetic. Unfortunately, this is a task which, in general, is not readily accomplished, in spite of significant advances in recent years [14]. A partial solution can be obtained quite efficiently, however, in terms of the quadratic subfield.

It should be clear the raising any element of GF(2^n) to the $2^m + 1$ power will map that element into the quadratic subfield. Due, to the representation of GF(2^n), however, this is guaranteed to be an m bit quantity:

$$(\alpha x_1 + x_0)^{2^m+1} = 0\alpha + x_1x_0 + x_0^2 + \beta x_1^2 \qquad (18)$$

The logarithm of this value can be looked up in a GF(2^m) logarithm table, and when divided by $2^m + 1$, it yields the logarithm of the original number, modulo $2^m - 1$.

8 Conclusion

Significant computational advantages can be obtained by representing a finite field as a quadratic extension field of one or more subfields. This type of field representation allows embedded subfields, as well the primary extension field to be simultaneously represented in normal form. Consequently, it becomes possible to rewrite the basic arithmetic operations for the extension field solely in terms of operations performed in one or more subfields.

References

[1] *Advanced Hardware Architectures: AHA4510 High Speed Reed Solomon Encoder/Decoder*, Data Sheet, 1989.

[2] E. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, 1968.

[3] R. Bose, D. Ray-Chaudhuri, "On a Class of Error Correcting Binary Group Codes," Inf. and Control, 3, pp. 68-79.

[4] R. Blahut, *Theory and Practice of Error Control Codes*, ISBN 0-201-10102-5, Addison-Wesley, Reading, Massachusetts, 1984.

[5] *CCSCS Recommendation for Telemetry Channel Coding,* CCSCS Secretariat, Communications and Data Systems Division, Code-TS, NASA, 1984.

[6] R. Chien, "Cyclic Decoding Procedures for the Bose–Chaudhuri–Hocquenghem Codes," IEEE Trans. Inf. Theory, IT-10, pp. 357-363, October 1964.

[7] G. Clarke, J. Cain, *Error-Correction Coding of Digital Communications*, ISBN 0-306-40615-2, Plenum Press, New York, 1982.

[8] N. Demassieux, F. Jutand, M. Muller, "A 10Mhz (255,223) Reed-Solomon Decoder," IEEE 1988 Custom Integrated Circuits Conference, pp. 17.6.1-17.6.4.

[9] G. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," IEEE Trans on Computers, vol 38, no 10, pp. 1383-1386, Oct 1989.

[10] G. Forney, *Concatenated Codes*, MIT Press, Cambridge, Massachusetts, 1966.

[11] T. Fujio, "A Study of High-Definition TV System in the Future," IEEE Trans. Broadcast., vol BC-24, no. 4, pp. 92-100, Dec 1978.

[12] A. Hillman, G. Alexanderson, *A first Graduate Course in Abstract Algebra*, Wadsworth Publishing, pp. 349-350, 1983.

[13] "A Cellular-Array Multiplier for GF(2^m)," IEEE Transactions on Computers, vol C-20, pp. 1573-1578, December 1971.

[14] R. Lidl, H. Niederreiter, *Introduction to Finite Fields and their applications*, Cambridge University Press, Cambridge, 1986.

[15] F. MacWilliams, N. Sloane, *The Theory of Error-Correcting Codes,* New York: North-Holland, 1977.

[16] J. Massey, J. Omura, "Computational Method and Apparatus for Finite Field Arithmetic," U.S. Patent, 1981.

[17] G. Maki, et al, "High-Speed Real-time Reed-Solomon Decoder," U.S. Patent No. 4,873,688, Issued: Oct. 10, 1989.

[18] R. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Publishers, Norwell, Massachusetts, 1987.

[19] G. Meeker, "High Definition and High Frame Rate Compatible N.T.S.C. Broadcast Television System," IEEE Trans. Broadcast., vol. 34, no. 3, pp. 313-322, Sept 1988.

[20] B. Mortimer, M. Moore, M. Sablatash, "Performance of a Powerful Error-Correcting and Detecting Coding Scheme for the North American Telex System (NABTS) for Random Independent Errors: Methods, Equations, Calculations and Results," IEEE Trans. Broadcast., vol 36, no. 2, pp. 113-131, June 1990.

[21] K. Niwa, T. Araseki, T. Nishitani, "Digital Signal Processing for Video," IEEE Circuits and Devices Magazine, vol 6, no 1, pp. 27-33, Jan 1990.

[22] I. Reed, G. Solomon, "Polynomial Codes over Certain Finite Fields," J. Soc. Industrial Applied Mathematics, 8, pp. 300-304.

[23] H. Shao, I. Reed, "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays," IEEE Transactions on Computers, vol. 37, no. 10, October 1988.

[24] Y. Sugiyama, M. Kasahara, S. Hirasawa, T. Namekawa, "A Method for Solving Key Equation for Decoding Goppa Codes," Inf. Control, 27, pp. 87-99, 1975.

[25] C. Wang, T. Truong, H. Shao, L. Deutsch, J. Omura, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)," IEEE Transactions on Computers, vol. C-34, no. 8, pp. 709-717, August 1985.

[26] S. Whitaker, K. Cameron, G. Maki, J. Canaris, P. Owsley, "Custom CMOS Reed Solomon Coder for the Hubble Space Telescope", Proceedings of the IEEE Military Communications Conference, Monterey, CA, October 1990, pp. 4.3.1-4.3.5.

[27] S. Whitaker, J. Canaris. K. Cameron, "Reed Solomon VLSI Codec for Advanced Television", IEEE Transactions on Circuits and Systems for Video Technology, June 1991, pp. 230-236.

[28] C. Yeh, I. Reed, T. Truong, "Systolic Multipliers for Finite Fields GF(2^m)," IEEE Transactions on Computers, vol C-33, pp. 357-360, April 1984.

[29] *Digital Spectrum Compatible*, Tech. Description, Zenith Electronic Corporation and AT&T, Feb. 22, 1991.

S53-62

191547 N94-21132 p. 16

A Cost-Effective Methodology for the Design of Massively-Parallel VLSI Functional Units

N.Venkateswaran, G. Sriram, J. Desouza
R. Srinivasan and G. Suresh
Dept. of Computer Science and Engineering
Sri Venkateswara College of Engineering
Pennalur, Sriperumbudur - 602 105
Tamilnadu, INDIA

Abstract - **In this paper we propose a generalized methodology for the design of cost-effective massively-parallel VLSI Functional Units. This methodology is based on a technique of generating and reducing a massive bit-array on the mask-programmable PAcube VLSI array. This methodology unifies (maintains identical data flow and control) the execution of complex arithmetic functions on PAcube arrays. It is highly regular, expandable and uniform with respect to problem-size and wordlength, thereby reducing the communication complexity. The memory-functional unit interface is regular and expandable. Using this technique functional units of dedicated processors can be mask- programmed on the naked PAcube arrays, reducing the turn-around time. The production cost of such dedicated processors can be drastically reduced since the naked PAcube arrays can be mass-produced. Analysis of the the performance of functional units designed by our method yields promising results.**

INDEX TERMS - Array Adder, Programmable Array of Array Adders (PAAA or PAcube), macrocell, Generalised Inner Product and Outer Product (GIPOP), Inner Product (IP), Chain Multiplication (CM), Bit-Product, Partition, Hacksaw, Domain, Partial Bit-product Array (PBA), Basic Sieve-Unit (BSU), Column Sieve, Macrocell Array, Carry-Row.

1 Description of the Generalized Methodology

A close analysis of the computational structures of several numerical algorithms like LU Decomposition, FFT, QR Decomposition, Convolution, Numeric methods of solving partial differential equations, and algorithms occurring in image processing, signal processing, pattern recognition etc., reveal that their computational structures can be expressed in terms of some complex combination of the following two equations which we call the Generalized Inner Product and Outer Product (GIPOP) equations [1] [2].

$$GIP = \sum_{i=1}^{N}(A_i * B_i)/C_i \tag{1}$$

$$GOP = \Pi_{i=1}^{N}(A_i + B_i)/C_i \tag{2}$$

GIP and GOP can be realized using Inner Product (IP) and Chain Multiplication (CM) functions, since the reciprocal/ division operation can also be brought under CM [3].

$$CM = \Pi_{i=1}^{N} A_i \tag{3}$$

$$IP = \sum_{i=1}^{N}(A_i * B_i) \tag{4}$$

where A_i, B_i, C_i are called OPERANDS, and their combinations above are called TERMS. 'N' is the number of terms (Problem Size).

IP and CM functions can be mapped on PAcube arrays using the generalized mapping technique presented in this paper. This mapping is based on the generation and reduction of a massive bit-product array on the PAcube VLSI array. Special purpose macrocell-based mask- programmable PACUBE (PA^3 - Programmable Array of Array Adders, pronounced pee-ay-cube) VLSI arrays have been proposed in [1] and [2]. The fundamental computing element of the PAcube macrocell is the ARRAY- ADDER. It can operate in any of the following modes

1. adding four 4-bit numbers
2. adding two 13-bit numbers
3. comparing two 13-bit numbers
4. XOR mode (16-input,8-output)

Based on the above methodology numerical algorithms need only be converted to GIPOP form for their parallel VLSI implementation. In addition, any fine-grain VLSI processor can be implemented by expressing its ALU or functional unit (FU) in terms of GIPOP functions. The control of such processors can also be easily mapped onto the PAcube VLSI array. [4]

2 The Generalized Mapping Technique for IP and CM

The task is to obtain an optimized mapping of IP and CM (based on Unsigned Binary number system) onto the PAcube VLSI array keeping in mind certain technological constraints. We have also generalized these mappings for the TWO'S COMPLEMENT FIXED POINT NUMBER SYSTEM. However it is not presented in this paper due to space constraints.

2.1 Massive Bit-Array Generation, Partitioning and Reduction

The bit-wise expansion of an IP or CM function results in a massive array of BIT-PRODUCTS. (A bit-product refers to the product of several bits, e.g. each a_ib_i in Fig.1.) Each row of bit-products is called a PARTIAL BIT-PRODUCT. The massive bit-product array is also called a PARTIAL BIT-PRODUCT ARRAY (PBA, shown in Fig.1). The PBA needs to be summed up to solve the IP or CM function. This is done on the PAcube Macrocell array in three steps: bit-array generation, partitioning and reduction.

Bit-Array Generation: Mathematically, BIT-ARRAY GENERATION refers to the algebraic bit-wise expansion of the function to be mapped. Physically, bit-array generation refers to the injection of the operands of the IP or CM problem into the macrocell array so as to form the PBA.

Bit-Array Partitioning: BIT-ARRAY PARTITIONING refers to the partitioning of the PBA into PARTITIONS and DOMAINS. Each partition is a 4 × 4 bit-product matrix which contains a group of sixteen bit- products, e.g. each box in Fig.1. The WEIGHT of a partition is given by the positional weight of its Least Significant Bit-product (LSB) in the PBA. A DOMAIN is defined to be a vertical column of partitions of the same weight. Thus the PBA is partitioned into several domains of weights increasing regularly in steps of 4.

Bit-Array Reduction: There are four levels of BIT-ARRAY REDUCTION. In BIT-PRODUCT REDUCTION the bits of a bit-product are multiplied (using the AND-plane of a macrocell), generating a single bit. Hence the 16 bit-products in a partition are reduced to 16 bits or four 4- bit numbers. In PARTITION REDUCTION these four 4-bit numbers of a partition are summed up by a macrocell operating in mode 1, to produce a 6-bit PARTIAL RESULT. In DOMAIN REDUCTION the 6-bit partial results of the partitions of a domain are added up by macrocells operating in mode 2. The result of summing up a domain is to produce 4-bits of final output and some carry bits which need to be passed to the next domain of higher weight. Finally, in MASSIVE BIT-ARRAY REDUCTION this CARRY-PROPAGATION among domains is performed, resulting in the final output.

This process of generating and summing up the PBA in the macrocell array (in three steps as described above) to obtain the solution for an IP or CM function is called MASSIVE ARRAY REDUCTION (MAR). We have identified the MAR1-4 and MAR4-4 processes for mapping IP and CM respectively on the PAcube array. These MAR processes have similar generation and reduction steps but differ only in their partitioning step as described in the following sections.

2.2 MAR1-4 IP

In the MAR1-4 process, to generate the massive array, partial bit- products of the same weight are clubbed together; and then this rearranged PBA is CROSS-CUT into partitions, as shown in Fig.1. The PBA is further partitioned into HACKSAWS and DOMAINS as described below.

In a MAR1-H process, a HACKSAW of size 'H' or a H-HACKSAW contains 'H' consecutive partitions (each shifted by one bit) along a STEPPED- COLUMN. Fig.2 (Block diagram of Fig.1) shows two stepped-columns, each containing two 4-hacksaws. (Each 4-hacksaw is made up of 4 consecutive partitions along a stepped-column.) The WEIGHT of a hacksaw is the lowest weight of its constituent partitions. A DOMAIN is a vertical column of h-hacksaws of the same weight. Fig.3 shows the block diagram of the PBA of Fig.2, partitioned into 4-hacksaws and domains. Note that (See Domain 2 in Fig.3) partitions 9-12 and 5-8 are two 4-hacksaws of the same weight, which therefore fall in one domain. Partitions 9-12 and 13-16 are two 4-hacksaws with different weights, which fall in subsequent domains although they belong to the same stepped-column.

The complete reduction of the PBA in the MAR1-H process is as described in Section II.A, with an additional level called HACKSAW REDUCTION, described below. Each of the four partitions of a 4- hacksaw are shifted by a bit. Hence the macrocells adding the four 6- bit partial results of these partitions must perform bit-shifted addition, which is achieved by suitably mask-programming the inter- macrocell communication lines.

From Figs.2 and 3, we see for the given problem that along a stepped- column, 16 bits are shared by the partitions/4-hacksaws of that stepped-column and along a row of partitions, 4 bits are shared by the partitions of that row. In other words 16 bits are shared row-wise by each row of 4-hacksaws, as shown in Fig.3.

3 MAR4-4 FOR 3-OPERAND CM

Similar to the MAR1-4 IP PBA generation and partitioning, we generate and partition the MAR4-4 CM PBA into domains and 4-hacksaws, as shown in Fig.8a (block diagram). By comparing Fig.8a with Fig.3, it can be seen that the MAR1-4 IP and MAR4-4 CM partitioned PBAs are similar. The reduction process for the MAR4-4 CM PBA is similar to the MAR1-4 IP reduction process explained in the previous section.

4 Generalized Mapping of IP and CM

The mapping methodology presented is a general technique of interconnecting basic processing elements (PEs) to design FUs of dedicated processors. In this section we give a methodology for mapping IP and CM on PAcube arrays, where the PE is a macrocell. While mapping, certain constraints like the communication complexity, levels of metalization, channel width, macrocell arithmetic overflow and level of parallelism.

The various levels of the PBA reduction process described in Section II.A are performed in the following mapping units :

LEVEL NO.	BIT-ARRAY REDUCTION LEVEL	PERFORMED BY
1	Bit-Product reduction	AND-gate
2	Partition reduction	Macrocell
3	Hacksaw reduction	Basic Sieve-Unit (BSU)
4	Domain reduction	Column Sieve
5	PBA reduction	Macrocell Array

The reduction at the first two levels has already been described in Sections II.A and II.B. We now describe the remaining three levels.

4.1 Basic Sieve-Unit (BSU)

Fig.4 shows a BASIC SIEVE-UNIT capable of reducing a 4-hacksaw. As described in Section II.B, in a 4-hacksaw 16 bits are common to the partitions stepped-column- wise and also row-wise. The injection of these bits into macrocells 1-4, is shown in Fig.4. In the initial loading phase (Cycle 0) the 16 bits common stepped-column-wise are loaded into the input register of macrocell 2, and the 16 bits common row-wise are loaded into the input register of macrocell 4. Each of macrocells 1-4 now has the bits needed to reduce one partition. In this way, all 4 partitions of the 4-hacksaw are injected simultaneously into macrocells 1-4 of the BSU. In the partition reduction phase (Cycle 1) macrocells 1-4 produce 6-bit partial results, which are added up in macrocells 5, 6, and 7, in the hacksaw reduction phase (Cycles 2 and 3). The BSU therefore reduces a 4- hacksaw in four cycles with 100of pipelining. Note that macrocells 1-4 are in mode 1 and 5-8 are in mode 2. Macrocell 8 is reserved for use in the column sieve.

4.2 Column Sieve

A COLUMN SIEVE (or just a SIEVE) is used to reduce a domain by summing up the hacksaws of the domain. A column sieve for a 4-term IP is shown in Fig.5. It is constructed by replicating (and placing one below the other) several of the 4-term BSUs shown in Fig.4. In each BSU there are one or more unused macrocell(s), e.g. Macrocell 8 in Fig.4. Since a sieve is made up of several BSUs placed one below the other, these unused macrocells fall along the same vertical column of macrocells in the sieve. These unused macrocells are used as INTERMEDIATE ACCUMULATORS (IAs), and the vertical column of macrocells containing these IAs is called the ACCUMULATIVE PIPELINE of the sieve (ACC-PIPE, See macrocell Column 4 in Fig.5).

We now explain the operation of a column sieve with reference to Fig.5. Each BSU of the sieve corresponds to a 4-hacksaw of the domain being reduced in the sieve. The hacksaws of a domain are fed into the subsequent BSU's of the sieve in subsequent cycles. Each IA along the acc-pipe is used to add the result of its corresponding BSU with the result of the IA above it. Note that this upper IA contains the accumulated result of all the BSU's above the current BSU. Thus the pipelined summation of the hacksaws of a domain is performed by the IAs in the acc-pipe. The output of a domain is obtained from the bottom-most IA of the acc-pipe in the last cycle. Note that the top- most IA does not have two results to add and therefore simply freeflows the result from the top-most BSU.

Different domains of the same PBA can be 100sieve and their results suitably shifted and accumulated in an external accumulator. This pipelining in the sieve is useful in the serial-parallel approach described later. Note that (See Fig.5), the first two vertical column of macrocells are always in mode 1 and the next two mode 2. The mode of operation of the macrocells of the sieve can hence be mask-programmed, thereby simplifying the control enormously.

The features of a column sieve are summarized as follows :

1. It used to sum up all the hacksaws of a domain in a 100% pipelined fashion.

2. It is constructed by replicating BSUs. The HEIGHT of a sieve is equal to the number of BSUs in the sieve which is equal to the number of hacksaws in the domain being reduced.

3. Its inputs have to be injected in a staggered fashion, i.e. each subsequent BSU in the sieve receives input in a subsequent cycle.

4. The results of the BSUs get accumulated down the accumulative pipeline of the sieve.

4.3 Macrocell Array Formation and Data Flow

As described in the previous section each domain of the partitioned PBA is reduced by a column sieve. The macrocell array is constructed by placing the corresponding column sieves (of different heights) side by side. By performing the MAR1-4 process (described in Section II.B) for a 16-bit 4-term IP, we obtain a PBA which has seven domains containing 1,2.3.4,3,2 and 1 4-hacksaws respectively. For this PBA, a block diagram of the distribution of data through the macrocell array is shown in Fig.6, where each box represents a BSU, and each vertical column of boxes represents a column sieve.

Peripheral Input Injection : Injection of input into the macrocell array is performed by using the peripheral BSU's. The 16 bits common to each row of hacksaws are injected into the rightmost BSU of each corresponding row of BSU's in the macrocell array, in alternate cycles as shown in Fig.6. Similarly, the 16 bits common to each stepped- column of partitions/hacksaws in the PBA are injected into the topmost BSU of each stepped-column of BSU's in the macrocell array in subsequent cycles, as shown in Fig.6.

Internal Hopping : Distribution of the input data through the macrocell array is shown in Fig.6. Bits injected row-wise hop along the row from one BSU to the next in every cycle. Bits injected stepped-column-wise hop along the stepped-column from one BSU to the next in every alternate cycle.

Data Flow Synchronization : As can be seen from Fig.6, each BSU receives its row-wise/ stepped-column-wise input in the same cycle, enabling it to reduce the corresponding 4-hacksaw. Note that each column sieve (a vertical column of boxes) is able to sum up the results of its BSUs because each subsequent BSU receives its input in the subsequent cycle.

Staggered Output : The result of summing up a domain is to produce 4- bits of output and some carry bits which need to be passed to the next domain of higher weight. This carry propagation between the column sieves is performed to obtain the final output. Note that (See Fig.6), the final output appears every alternate cycle along the rightmost stepped-column of BSUs, and every subsequent cycle along the bottom-most row of BSUs. The final output appears 4-bits at a time starting from the right-most sieve in the fifth cycle.

4.4 Detailed Mapping for MAR1-4 8-BIT 4-TERM IP

A block diagram of the distribution of the input data through the macrocell array for a 8-bit 4-term IP is shown in Fig.7a. The actual macrocell array mapping is shown in Fig.7b. The injection of the row- wise and stepped-column-wise inputs is actually peripheral. However some redundant macrocells have been shown along the periphery in Fig.7b, in order to obtain a regular geometry for the macrocell array.

These redundant macrocells may be used for two purposes :

1. Inculcation of redundancy for fault-tolerance during manufacture.

2. To give free flows in order to make the inputs and outputs (along the right-most stepped-column of BSUs) appear in subsequent cycles instead of alternate cycles.

Intermediate Carry Propagation : For a 4-term MAR1-4 IP sixteen 4- hacksaws of a domain can be added without exceeding the capacity (14 bits) of the array-adder. Overflow occurs in the IA after feeding 16 rows of BSUs of a sieve. An intermediate carry-row of macrocells is inserted after the 16th row of BSUs (32nd row of macrocells) so that only the lower 4 bits of the 14-bit output are passed down along the sieve and the remaining (upper) 10-bits are passed sideways to a sieve of higher weight.

4.5 Detailed Mapping for Fully Parallel 3-Operand CM

The MAR4-4 process applied to the 8-bit 3-operand CM results in the partitioned PBA shown in Fig.8a (Shown as a block diagram.) The number of domains in the PBA is 5 and the number of 4-hacksaws in the domains are 1, 3, 4, 3, and 1 respectively. From Figs.3 and 8a, it is seen that the respective partitioned PBAs for MAR1-4 IP and MAR4-4 CM are similar. The macrocell array formation (using sieves of heights 1, 3, 4, 3, and 1 respectively) and the reduction of domains in the sieves for CM are similar to those of IP, which have been explained above. The data flow for 8-bit 3-operand CM is shown in Fig.8b and the mapping is shown in Fig.8c. The row-wise and stepped-column-wise peripheral injection and the computation flow can be seen from Fig.8c. The data flow, computation and the control for IP and CM are similar but for the following two differences :

1. The number of input bits common row-wise and stepped-column-wise.
2. The height of the column sieves.

For CM, injection along a stepped-column requires only 7 bits and along a row requires 4 + (t − 2) bits, where t is the number of terms. The cycle-wise execution of CM is very similar to that of IP described above, and is not presented here due to space constraints.

4.6 Generalized T-Term IP and CM Mapping

We can generalize the above mapping for IP to any number of terms by two approaches:

- Fully-Parallel Approach for IP : The BSU described in Section III.A can be used to map 4-Term IP of any wordlength in a Fully Parallel fashion because of the fact that it reduces one 4-hacksaw. But a different BSU is required for IP of different problem-size. This is because a larger IP (of problem size $N = 4*k$) can be considered to be composed of one or more (i.e. k) 4-term IPs. If we wish to reduce this larger IP in Fully-Parallel fashion, the BSU must be able to reduce k different 4-hacksaws simultaneously. We have so far designed basic sieve-units for up to 32 term IP problems. The 32-term BSU permits simultaneous injection of eight 4-hacksaws, each 4-hacksaw corresponding to a sub-IP (of 4 terms) of the 32-term IP with each sub-IP needing its own 16 row-wise and stepped-column-wise bits (as stated in Section II.B). Therefore, we design a BSU for the particular IP problem size (say t-term IP), construct column sieves and then map the t-term IP for any wordlength.

- Serial-Parallel Approach : If the number of terms (operands) is too many the IP (CM) problem can directly be broken into the sum (product) of several IPs (CMs). each of fewer terms but of increased wordlength. Each of these sub-problems can be reduced serially (pipelined fashion) in separate modules and the results accumulated (multiplied). Thus the serial-parallel approach in dealing with problem size is to construct several small macrocell arrays on a chip, rather than just one large monolithic array. The small arrays or Islands of macrocells are separated by a channel width larger than the intra-island width, for the purpose of communication. The level of parallelism can be tuned to suit user requirements. Using WSI we can fabricate several islands on a wafer thereby enhancing the power of the wafer while maintaining modularity.

4.7 Mapping of GIPOP and Other Numeric Functions

There are two ways of mapping combined operations like IP + CM, IP * CM, IP / CM etc. One is by forming the corresponding massive array for the entire combination and reducing it, and the other is by reducing IP and CM in independent macrocell array modules and then combining the outputs in another macrocell array module. This inter- module mapping can be done without affecting the pipelining rate of the individual modules. Operations like massive array addition and scalar-vector multiplication forms special cases of GIPOP functions. They have similar data flow as that of IP and CM operation and hence can be easily mapped using the same strategy. It is also suited to implementation on Wafer Scale Integration (WSI) by fabricating individual modules of macrocell arrays.

5 Evaluation

Control Unit: In designing high performance computing processors the control should be kept as simple as possible. Simple control helps in achieving better performance. In executing a combination of the GIPOP functions the basic control operation to be performed is the latching of the I/O registers of the macrocell. Hence the design of the control unit for controlling the operation of any functional unit mapped on the PAcube is extremely simple. This is because the execution of the basic GIPOP functions requires only a single global clock. Memory-Functional Unit Interface : Stepped-Column-wise input is injected staggered by 16 bits (4 bits each of 4 operands from the LSB to the MSB) in alternate cycles. Row-wise input is injected staggered by 16 bits (4 bits each of 4 operands from the LSB to the MSB) in subsequent cycles. Therefore data flow from the memory is regular. Hence Memory-Functional unit communication is very simple, regular and expandable.

Functional-Unit Performance Evaluation: The advantages are :

- Naked PAcube arrays can be mass-produced and complex functional- units can be mask-programmed on them in two layers of metallization thereby reducing cost and turnaround time drastically. Dedicated processors can therefore be easily manufactured based on PAcube arrays.

- The degree of parallelism can be easily varied on PAcube arrays as the mapping of functional units described is expandable and regular without increasing control complexity.

- Serial-parallel IP and CM can be mapped for any wordlength and problem-size. Fully-parallel IP and CM can be mapped for any wordlength but limited problem-size. This implies any numerical function can be mapped on PAcube arrays using the methodology.

- The pipelining degree is 100% and after an initial latency we obtain a complete output to a problem in every clock cycle. The pipelining rate is approximately equal to a macrocell delay which is of the order of 10 nanoseconds for CMOS 1.2 micron technology simulated on SPICE. A table of the throughput in Million Operations Per Second, for various macrocell array sizes and IP problem sizes is given below.

INNER PRODUCT Macrocell Array size	Number of Terms → Operand bit length	4	8	16	32
10 × 6	8 bits	100	50	25	12.5
17 × 10	16 bits	100	50	25	12.5
28 × 18	32 bits	100	50	25	12.5

- If we fix the macrocell array size per chip as approximately 1000 (M=1000), then the IP problems that can be solved on such an array using our mapping are given below.

n-bits	4	8	12	16	20	24	28	32	64
t-terms	512	128	64	32	20	14	11	8	2

- Other VLSI design approaches (such as custom design or any other VLSI array based approach) can use the methodology presented in this paper to design functional units of Massively-Parallel- Processors.

- The methodology is eminently suited to WSI design.

- This methodology permits inculcation of redundancy for fault-tolerance up to any desired percentage by introducing redundant rows anywhere in the massive VLSI array.

The limitations are:

1. Currently fully-parallel mapping for only 3-operand CM has been designed.
2. The PAcube VLSI array size (i.e. number of macrocells) per chip is limited by technological constraints and this limits the size of the problem we can map on one chip/wafer.
3. Any VLSI array based design is inefficient in speed and area-utilization when compared to the custom approach.

6 Conclusion

We have presented a generalized mapping methodology for the design of GIPOP-based functional units of massively parallel processors implemented on PAcube arrays. The mapping is presented for IP and CM only since all other numeric functions can be realised using them. This mapping is easily expandable, and has excellent I/O qualities, very simple control, 100and high throughput. The mapping is presented using PAcube arrays due the advantages of reduced cost and turnaround time of manufacturing. However, any other VLSI design approach can use the methodology presented in this paper to design functional units of Massively-Parallel Processors. The application of this paper is mainly in the design of dedicated ultra-high performance fine-grain massively parallel processors.

References

[1] Venkateswaran, .N, "PAcube VLSI Arrays for SuperComputers". Proceedings of the First International Conference on SuperComputing, conducted by IEEE computer Society and Pentagon, Florida, December 1985.

[2] Venkateswaran, .N, et al., "A Special Purpose Silicon Compiler for Designing SuperComputing VLSI systems". Proceedings of the Third NASA Symposium on VLSI Design, held at NASA Space Engineering Research Center (SERC), University of Idaho, Idaho, USA, October 1991.

[3] Cavanagh, Joseph, "Digital Computer Arithmetic". McGraw Hill, 1986.

[4] Venkateswaran, N., et al., "A Novel VLSI Processor Architecture for Supercomputing Arrays', proceedings of the Fifth NASA Symposium on VLSI Design, held at the University of New Mexico, Albuquerque, New Mexico, 1993.

Figure 1. BLOCK DIAGRAM OF THE MACROCELL

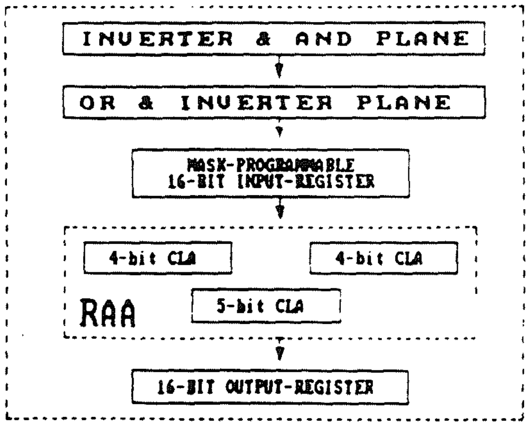


FIGURE 2. MAR1-4 8-BIT 4-TERM IP PBA

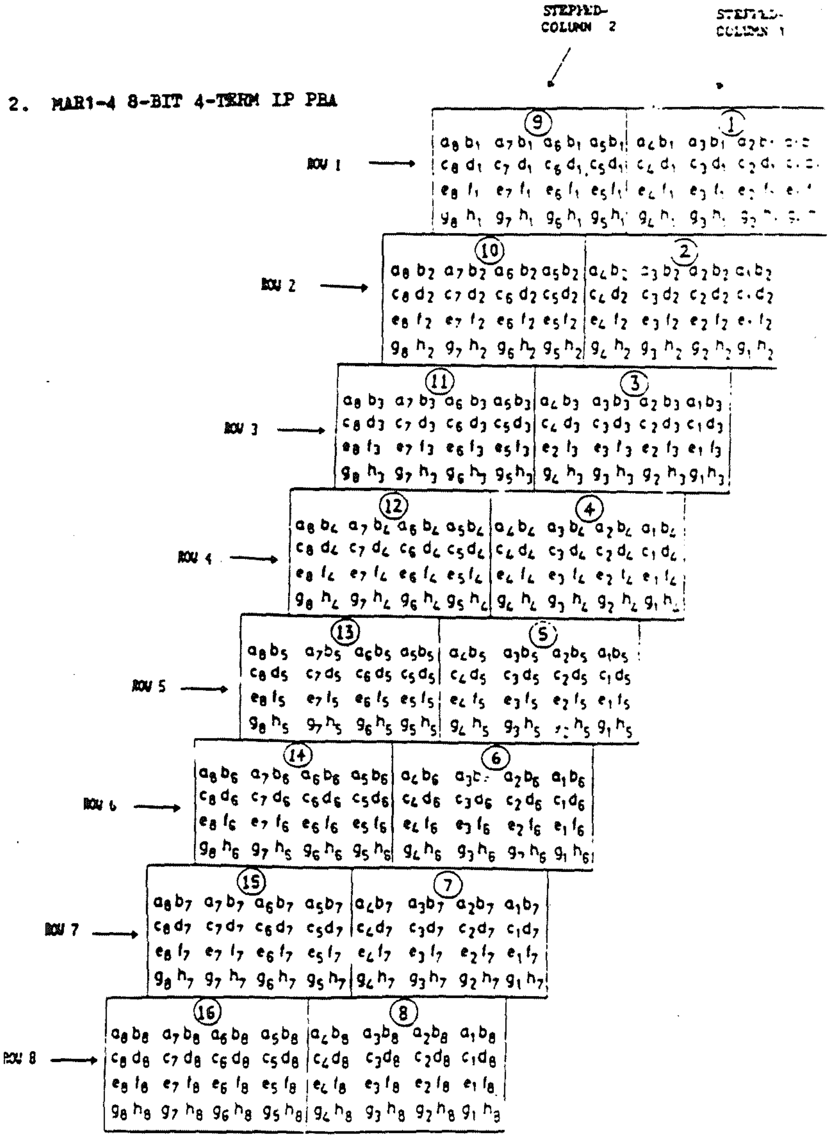
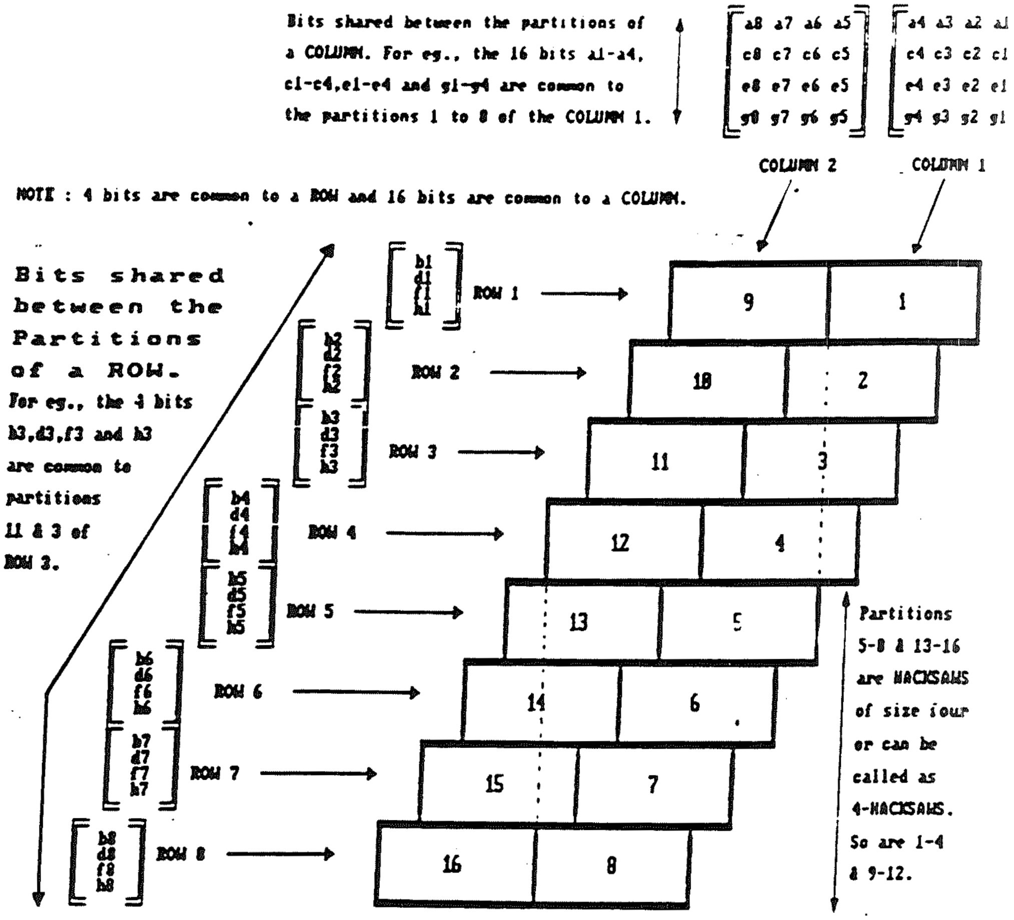

STEPPED-COLUMN 2
STEPPED-COLUMN 1
ROW 1
9
1
a8 b1 a7 b1 a6 b1 a5 b1 a4 b1 a3 b1 a2 b1 a1 b1
c8 d1 c7 d1 c6 d1 c5 d1 c4 d1 c3 d1 c2 d1 c1 d1
e8 f1 e7 f1 e6 f1 e5 f1 e4 f1 e3 f1 e2 f1 e1 f1
g8 h1 g7 h1 g6 h1 g5 h1 g4 h1 g3 h1 g2 h1 g1 h1
ROW 2
10
2
a8 b2 a7 b2 a6 b2 a5 b2 a4 b2 a3 b2 a2 b2 a1 b2
c8 d2 c7 d2 c6 d2 c5 d2 c4 d2 c3 d2 c2 d2 c1 d2
e8 f2 e7 f2 e6 f2 e5 f2 e4 f2 e3 f2 e2 f2 e1 f2
g8 h2 g7 h2 g6 h2 g5 h2 g4 h2 g3 h2 g2 h2 g1 h2
ROW 3
11
3
a8 b3 a7 b3 a6 b3 a5 b3 a4 b3 a3 b3 a2 b3 a1 b3
c8 d3 c7 d3 c6 d3 c5 d3 c4 d3 c3 d3 c2 d3 c1 d3
e8 f3 e7 f3 e6 f3 e5 f3 e2 f3 e3 f3 e2 f3 e1 f3
g8 h3 g7 h3 g6 h3 g5 h3 g4 h3 g3 h3 g2 h3 g1 h3
ROW 4
12
4
a8 b4 a7 b4 a6 b4 a5 b4 a4 b4 a3 b4 a2 b4 a1 b4
c8 d4 c7 d4 c6 d4 c5 d4 c4 d4 c3 d4 c2 d4 c1 d4
e8 f4 e7 f4 e6 f4 e5 f4 e4 f4 e3 f4 e2 f4 e1 f4
g8 h4 g7 h4 g6 h4 g5 h4 g4 h4 g3 h4 g2 h4 g1 h4
ROW 5
13
5
a8 b5 a7 b5 a6 b5 a5 b5 a4 b5 a3 b5 a2 b5 a1 b5
c8 d5 c7 d5 c6 d5 c5 d5 c4 d5 c3 d5 c2 d5 c1 d5
e8 f5 e7 f5 e6 f5 e5 f5 e4 f5 e3 f5 e2 f5 e1 f5
g8 h5 g7 h5 g6 h5 g5 h5 g4 h5 g3 h5 g2 h5 g1 h5
ROW 6
14
6
a8 b6 a7 b6 a6 b6 a5 b6 a4 b6 a3 b6 a2 b6 a1 b6
c8 d6 c7 d6 c6 d6 c5 d6 c4 d6 c3 d6 c2 d6 c1 d6
e8 f6 e7 f6 e6 f6 e5 f6 e4 f6 e3 f6 e2 f6 e1 f6
g8 h6 g7 h6 g6 h6 g5 h6 g4 h6 g3 h6 g2 h6 g1 h6
ROW 7
15
7
a8 b7 a7 b7 a6 b7 a5 b7 a4 b7 a3 b7 a2 b7 a1 b7
c8 d7 c7 d7 c6 d7 c5 d7 c4 d7 c3 d7 c2 d7 c1 d7
e8 f7 e7 f7 e6 f7 e5 f7 e4 f7 e3 f7 e2 f7 e1 f7
g8 h7 g7 h7 g6 h7 g5 h7 g4 h7 g3 h7 g2 h7 g1 h7
ROW 8
16
8
a8 b8 a7 b8 a6 b8 a5 b8 a4 b8 a3 b8 a2 b8 a1 b8
c8 d8 c7 d8 c6 d8 c5 d8 c4 d8 c3 d8 c2 d8 c1 d8
e8 f8 e7 f8 e6 f8 e5 f8 e4 f8 e3 f8 e2 f8 e1 f8
g8 h8 g7 h8 g6 h8 g5 h8 g4 h8 g3 h8 g2 h8 g1 h3

FIGURE 3. BLOCK DIAGRAM OF 8 BIT 4 TERM IP



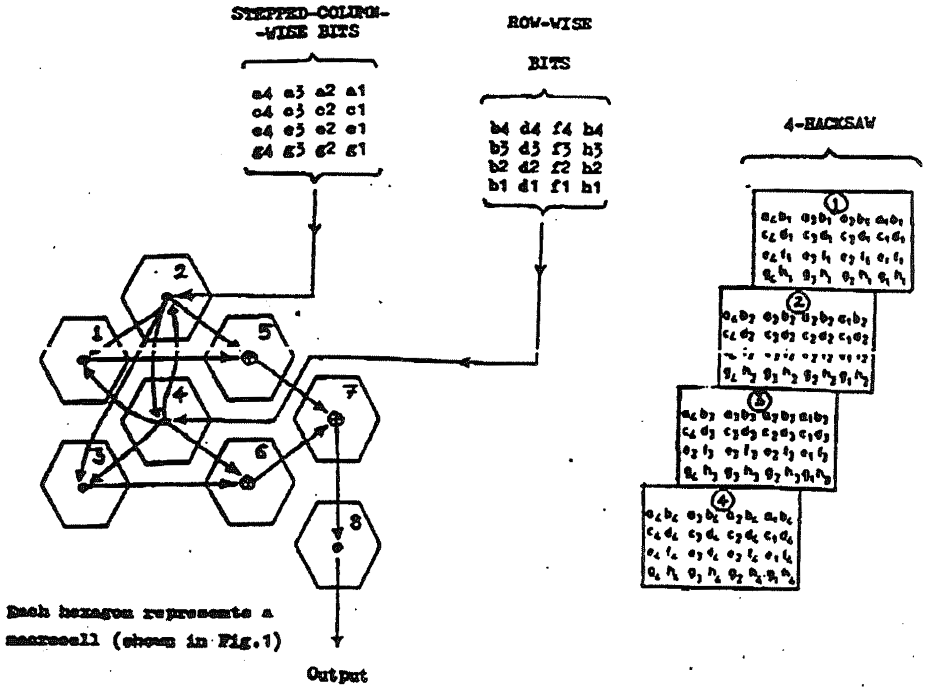
Bits shared between the partitions of a COLUMN. For eg., the 16 bits a1-a4, c1-c4,e1-e4 and g1-g4 are common to the partitions 1 to 8 of the COLUMN 1.

NOTE : 4 bits are common to a ROW and 16 bits are common to a COLUMN.

Bits shared between the Partitions of a ROW. For eg., the 4 bits b3,d3,f3 and h3 are common to partitions 11 & 3 of ROW 3.

Partitions 5-8 & 13-16 are HACKSAWS of size four or can be called as 4-HACKSAWS. So are 1-4 & 9-12.

FIGURE 4. 4 TERM BASIC SIEVE UNIT



Each hexagon represents a macrocell (shown in Fig.1)

FIGURE 5. COLUMN SIEVE

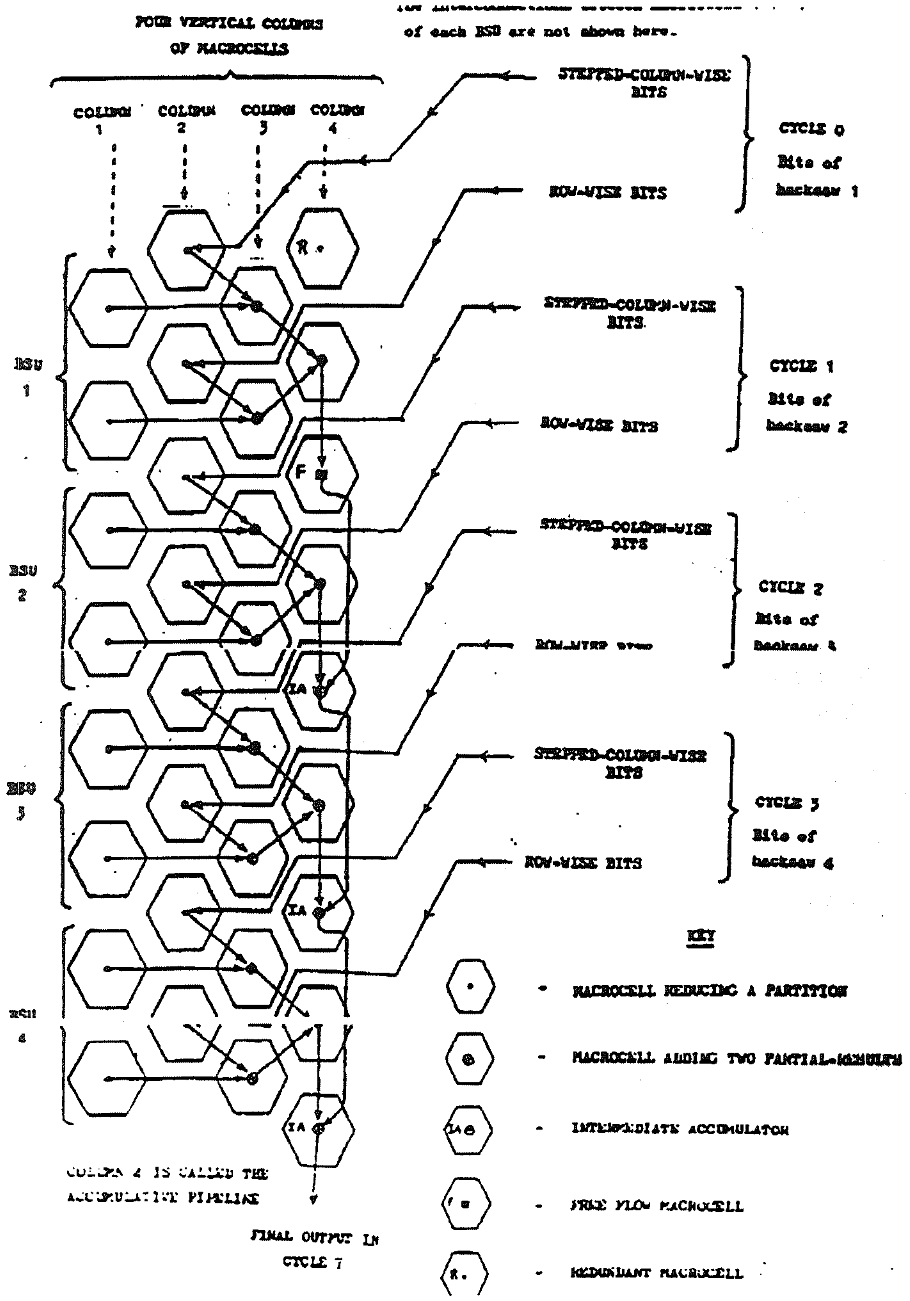


FIGURE 6. DATA FLOW DIAGRAM (16-BIT 4-TERM IP)

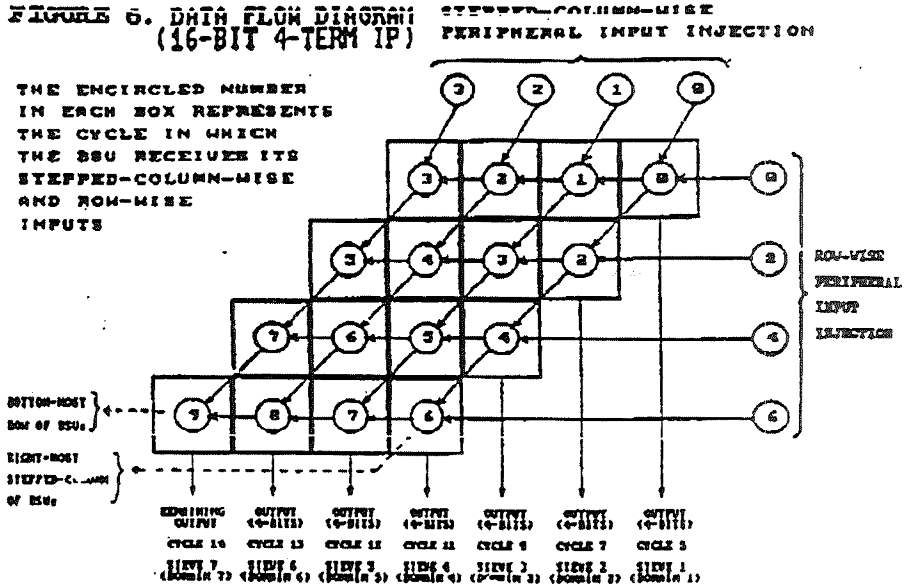

STEPPED-COLUMN-WISE PERIPHERAL INPUT INJECTION
THE ENCIRCLED NUMBER IN EACH BOX REPRESENTS THE CYCLE IN WHICH THE BSU RECEIVES ITS STEPPED-COLUMN-WISE AND ROW-WISE INPUTS
3
2
1
0
3
2
1
0
0
3
4
3
2
2
7
6
5
4
4
9
8
7
6
6
ROW-WISE PERIPHERAL INPUT INJECTION
BOTTOM-MOST ROW OF BSUs
RIGHT-MOST STEPPED-COLUMN OF BSUs
REMAINING OUTPUT CYCLE 14 SIEVE 7 (DOMAIN 7)
OUTPUT (4-BITS) CYCLE 13 SIEVE 6 (DOMAIN 6)
OUTPUT (4-BITS) CYCLE 12 SIEVE 5 (DOMAIN 5)
OUTPUT (4-BITS) CYCLE 11 SIEVE 4 (DOMAIN 4)
OUTPUT (4-BITS) CYCLE 9 SIEVE 3 (DOMAIN 3)
OUTPUT (4-BITS) CYCLE 7 SIEVE 2 (DOMAIN 2)
OUTPUT (4-BITS) CYCLE 5 SIEVE 1 (DOMAIN 1)

FIGURE 7a. DATA FLOW DIAGRAM (8-BIT 4-TERM IP)

Each of the encircled number represents a cycle number.

FIGURE 7b. COMPUTATION FLOW DIAGRAM

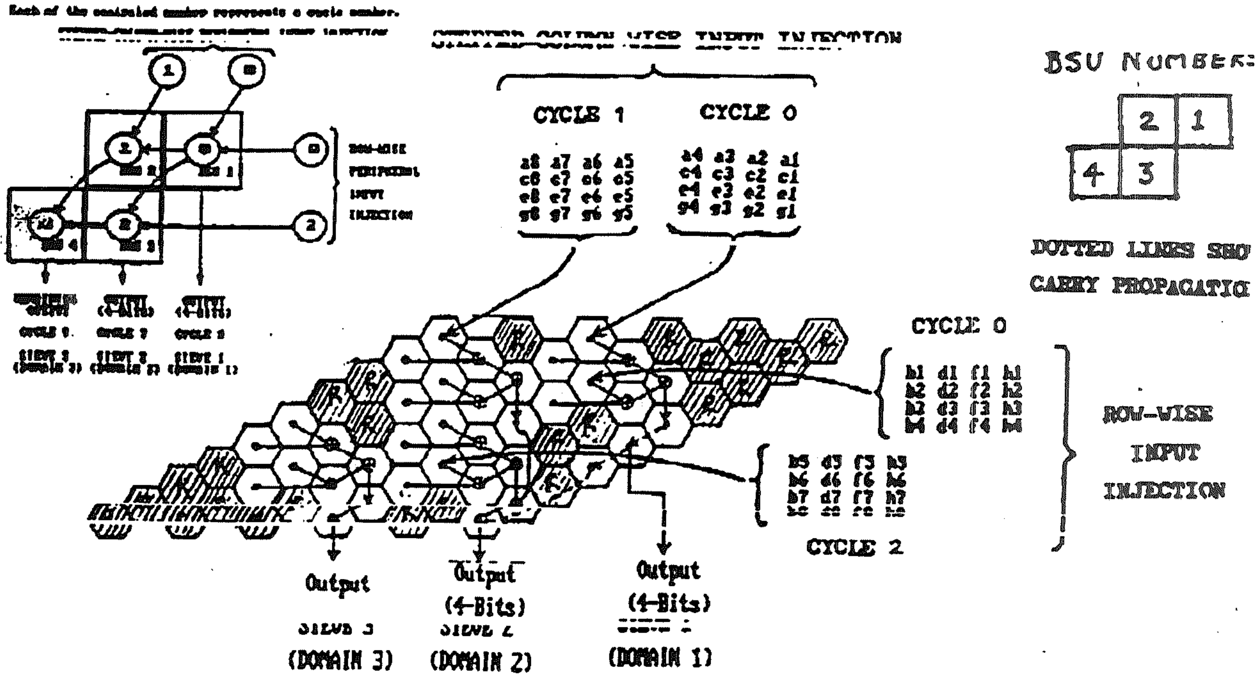

STEPPED-COLUMN-WISE PERIPHERAL INPUT INJECTION
1
0
2
BSU 2
0
BSU 1
0
2
BSU 4
2
BSU 3
2
ROW-WISE PERIPHERAL INPUT INJECTION
REMAINING OUTPUT CYCLE 9 SIEVE 3 (DOMAIN 3)
OUTPUT (4-BITS) CYCLE 7 SIEVE 2 (DOMAIN 2)
OUTPUT (4-BITS) CYCLE 5 SIEVE 1 (DOMAIN 1)
STEPPED-COLUMN-WISE INPUT INJECTION
CYCLE 1
CYCLE 0
a8 a7 a6 a5
c8 c7 c6 c5
e8 e7 e6 e5
g8 g7 g6 g5
a4 a3 a2 a1
c4 c3 c2 c1
e4 e3 e2 e1
g4 g3 g2 g1
BSU NUMBER:
2
1
4
3
DOTTED LINES SHO
CARRY PROPAGATIO
CYCLE 0
b1 d1 f1 h1
b2 d2 f2 h2
b3 d3 f3 h3
b4 d4 f4 h4
b5 d5 f5 h5
b6 d6 f6 h6
b7 d7 f7 h7
b8 d8 f8 h8
CYCLE 2
ROW-WISE INPUT INJECTION
Output SIEVE 3 (DOMAIN 3)
Output (4-Bits) SIEVE 2 (DOMAIN 2)
Output (4-Bits) SIEVE 1 (DOMAIN 1)

FIGURE 8a. DOMAINS AND 4-HACKSAWS FOR 8-BIT 5-OPERAND CM

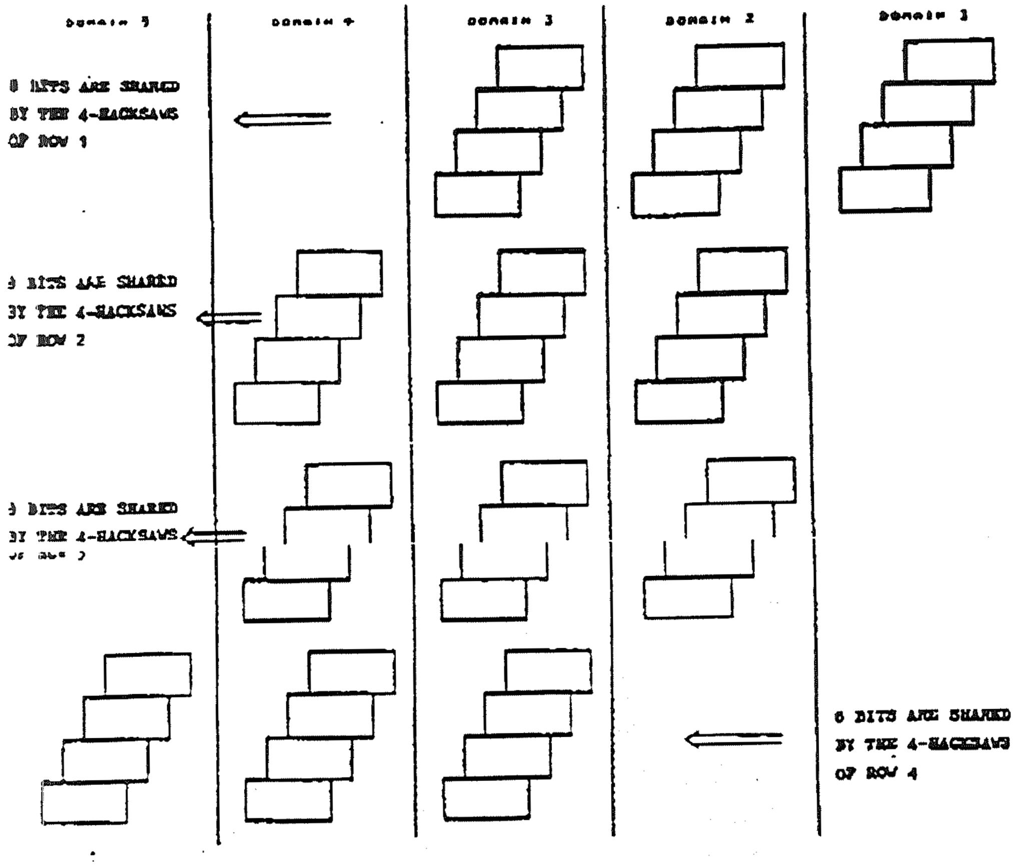
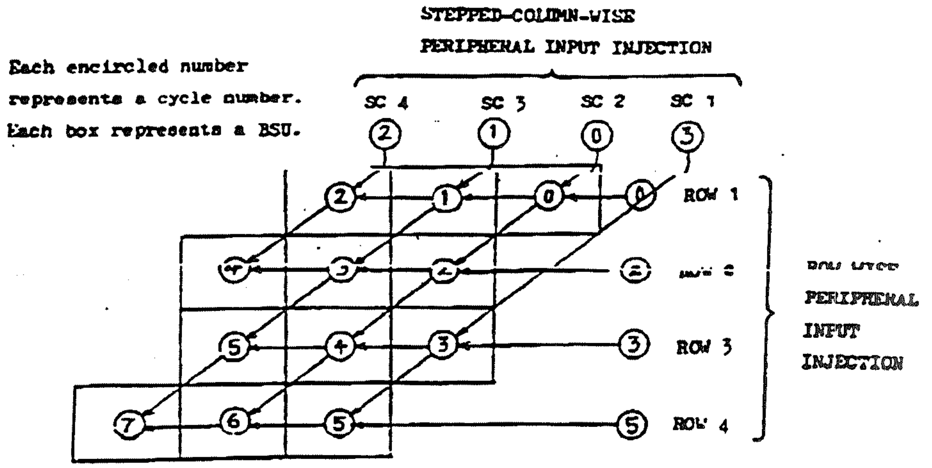


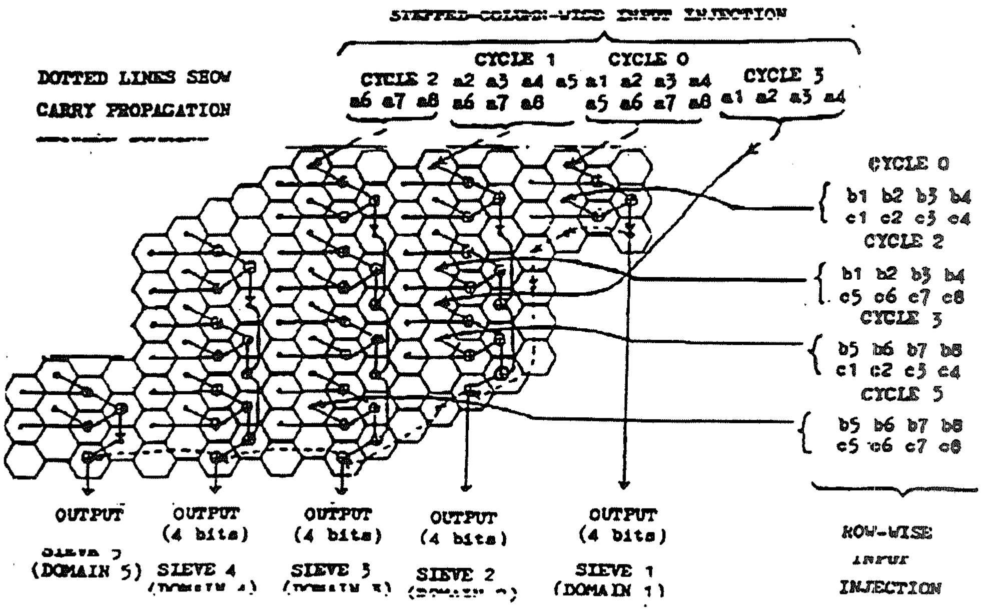
FIGURE 8b. DATA FLOW DIAGRAM FOR 8 BIT 3 OPERAND CM TYPE (1) MAPPING



Each encircled number represents a cycle number. Each box represents a BSU.

Each of the injected ROWS corresponds to a row of 4-hacksaws in the CM FBA.

FIGURE 8c. MAR 4-4 (8 BIT 3 OPERAND) CM TYPE (1) MAPPING



DOTTED LINES SHOW CARRY PROPAGATION

Behavioral Model Simulation Studies of an Image Compressor

Navid Madani and Sterling Whitaker
Department of Electrical Engineering
University of Idaho, Moscow, ID 83843

Abstract - **The high initial cost of prototype Very Large Scaled Integrated (VLSI) parts requires that sufficient verifications be made to eliminate design errors before actually producing the prototypes. Logic simulation is the means by which the VLSI engineer can ensure that the design will function properly. The computer time required for logic simulation can be reduced through the use of behavioral models. Behavioral models, however, require time to write and verify and they do not always produce a dramatic speed up in logic simulation time. This paper presents a study of behavioral modeling aimed at discovering which circuit types benefit most from the use of behavioral models for logic simulation.**

1 Introduction

Logic simulation is an important part of the design process in developing Very Large Scale Integrated (VLSI) circuits. The success of a VLSI project depends on thorough simulation before fabrication. The speed, efficiency and accuracy of a logic simulation are vital to a designer. This paper evaluates the effective use of behavioral modeling in logic simulation.

The candidate VLSI chip selected for making simulation comparisons was the Universal Source Encoder (USE) which was designed and fabricated for aerospace image compression purposes. The USE chip is of a relatively moderate size and has a variety of subcircuits. Moreover, the transistor level simulation data was already available for comparison. This study compares simulation time using behavioral models with full transistor models for various circuits. Resulting in guidelines for the selection of the type of circuits which provide the best return for the investment of coding behavioral models.

The multi-level simulator used is NOVA. NOVA has the capability to simulate from the switch level through the functional model level. It also uses the required 21 states to completely describe a CMOS circuit [1]. Because of the software data structures used NOVA is a fast simulator. NOVA simulates large circuits, 1 million transistors, at 50,000 Events/Second on an HP9000/380 or 100,000 Events/Second on an HP9000/730.

The NOVA functional modeling language is a super-set of 'C'. It is based on a set of 'C' language function calls. Moreover, the user is free to do anything that the 'C' language is capable of doing. There are two major parts to the functional modeling in NOVA. First the 'BOLT' language model description. Second, the 'C' language functional description.

BOLT, block oriented language translator, is the circuit description language that provides the design engineer with a means to describe a schematic diagram to the computer in a textual format. For more information refer to [7]. Further details on NOVA can be obtained from [8].

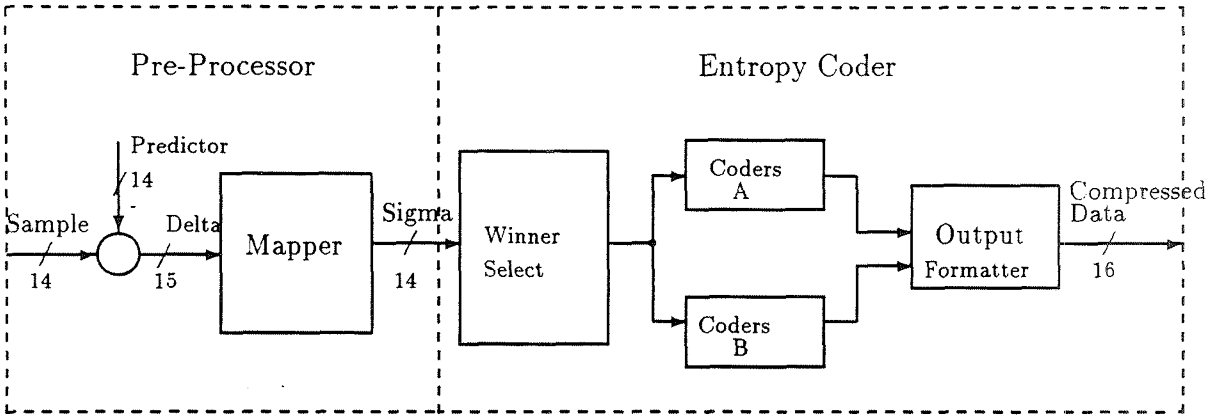


Figure 1: Block Diagram of Encoder (USE) Architecture

2 General Description of USE

A simplified explanation of the encoder is presented in this paper. For a more detailed description refer to [4, 5]. The block diagram of the USE architecture is shown in Figure 1. The USE accepts image data that is quantized for N={4..14} bits [2] which is sent to the Mapper section of the chip. The Mapper calculates the difference between a predicted pixel and the present pixel (Delta Value). Later the differences are mapped to a positive integer called Sigma Value. Sigma Values from the Mapper are sent to the Entropy Coder to calculate the winning option.

The Entropy Coder performs a twelve option Rice coding set that covers an entropy range from 1.5 bits per pixel to about 14 bits per pixel [2]. The winning option is selected on the basis of the exact number of bits which provide maximum compression. The twelve options calculated in parallel.

The mapper block performs a differencing and mapping of incoming samples. CNTBLK is the largest block in the chip and selects the winning code option. FBLK and KBLK are the coding blocks that perform the 12 different coding operations. OUTBLK does the output formatting. INBLK is a control section. SIGFIFO stores the sigma values until the winner is selected. EVAL2 does some calculations needed by the coding blocks.

The coding option with the least number of bits will win. These options include the default option, the Fundamental Sequence (FS) and the sample splitting for k={1..10}. The clk signal for the USE has a maximum frequency of 20 MHz. The input sample rate is 20 megasamples per second and the output sample rate is 20 megawords per second [2]. All calculations and coding are based on a fixed block size of 16 pixels. A reference pixel must be included at the start of the data block.

3 Simulation of Subsections of the USE

Different sections of the USE chip were selected for testing by behavioral modeling simulation. The type of blocks which have been modeled and simulated are examples of a wide variety of circuit types. Simplified description of the subsections of the USE chip has been presented in this paper for more detailed information refer to [9].

3.1 MAPPER

Mapper is an example of a circuit with a relatively complex mathematical function. It has numerous registers, muxes, and a large adder. The MAPPER block performs subtraction and mapping operations on incoming samples. In the pre-processor part, Figure 1, the reference sample is initially saved at the start of every data block such as a scanline. This reference is used to predict the next sample. The MAPPER takes the difference between the two n bit values, the present sample and the predictor, resulting in n+1 bits called delta value. Later, the MAPPER reduces them to n bits, called Sigma values. Sigma values are non-negative integers that the mapper outputs to the entropy coder.

3.2 SIGFIFO

SIGFIFO serves as a memory element by storing the Sigma values until the winning option is calculated in the CNTBLK section of the USE. SIGFIFO is a fully static 36 word x 14 bit RAM configured as a FIFO [3]. FIFOs are commonly used element in the design of VLSI circuits. SIGFIFO is an example of a large block of circuitry with a relatively simple overall operation. A block diagram of the SIGFIFO is shown in Figure 2. The inputs are C, D0...D13, RDLN, WRLN and RN. The outputs are Q0...Q13.

3.3 CBITXX

The CBITXX is an example of a component within a complex section. CBITXX is a bitslice and is repeated many times in the USE architecture. It is composed of a D Flip Flop, a 3 to 1 mux, a DLD register and a tristate cell called TRI. Refer to Figure 3. The input pins include C, EN, S0, S1, D and LD. The output pins include CNT which is the output of the TRI, and CNTIN which is the output of the 3 to 1 MUX.

3.4 ADDX

The ADDX is an example of a simple basic arithmetic function which is commonly used in the design of VLSI circuits. ADD8 and ADD9 are basically two adders used in the CNTBLK section of the chip. ADD8, adds two 8 bit numbers, A0...A7 with B0...B7, and sends out an eight bit number,Q0...Q7. ADD9 is almost the same except it adds two 9 bit numbers and also has a a carry out, COUTN.

3.5 DLYREG

DLYREG is an example of a simple circuit, which when arrayed, results in a large block of circuitry. DLYREG is made up of 6 rows of 16 bit Flip Flops. Refer to Figure 4. The only function of this subcircuit is to delay the output data and BLKRDY signal 6 clock cycles so that the delay would be a power of 2. The pipelined delay of the entire chip is now set at 64 clock cycles rather than 58 [3].

13.2.4

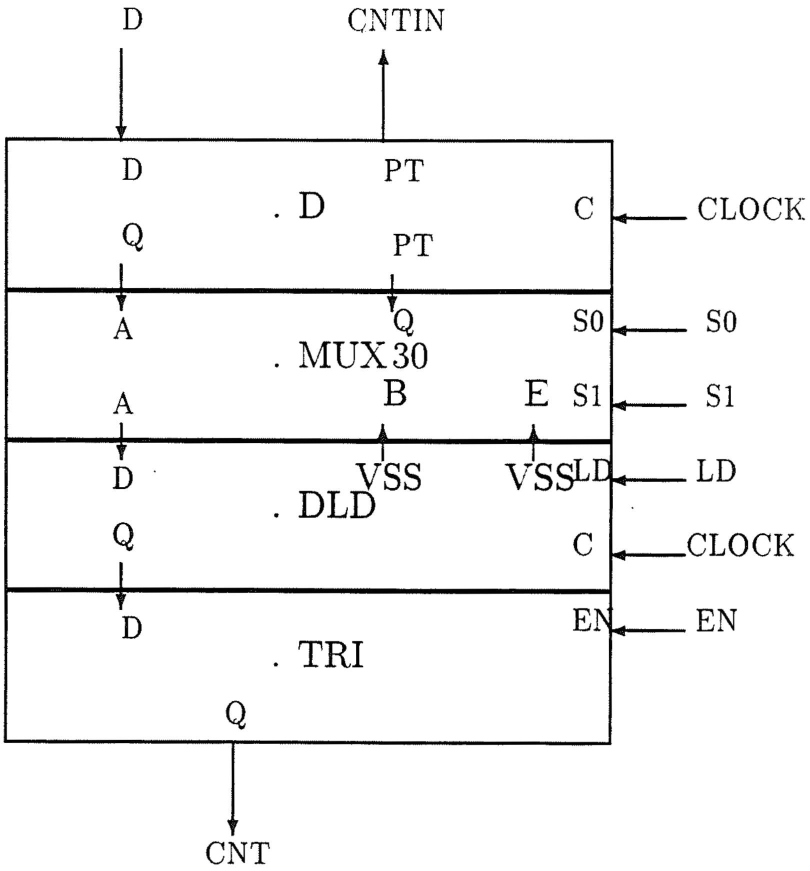


Figure 2: Block Diagram of SIGFIFO

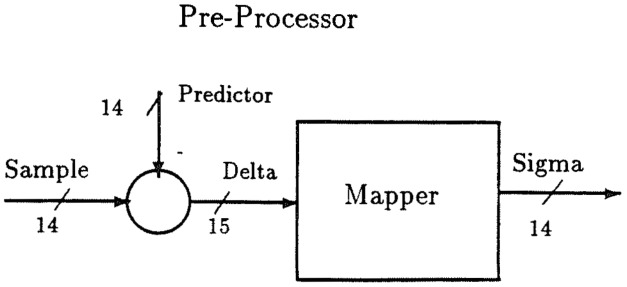


Figure 3: Block Diagram of CBITXX

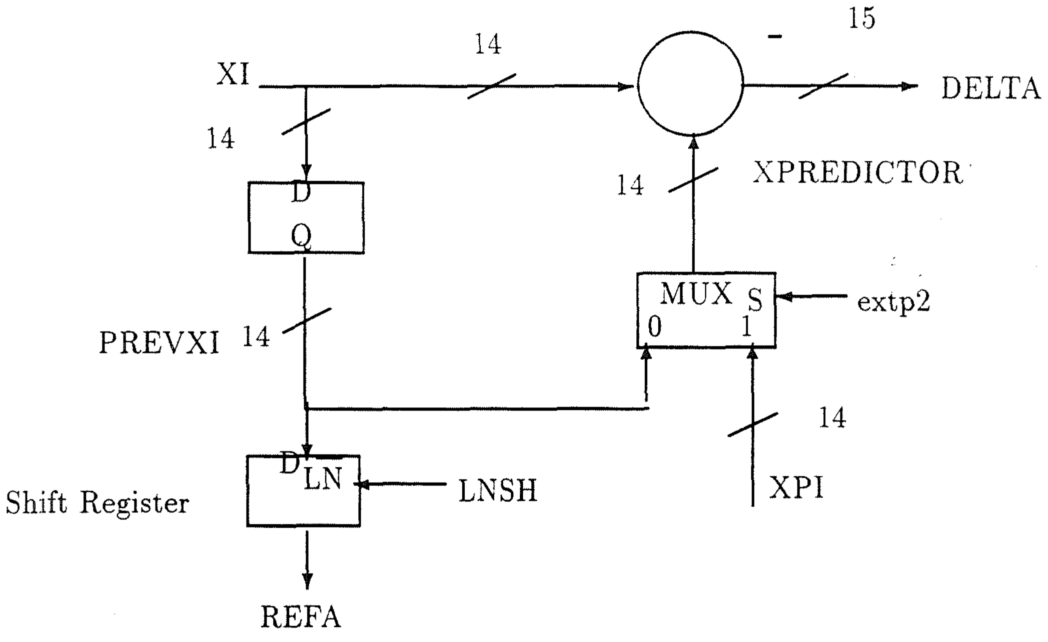


Figure 4: Block Diagram of DLYREG

4 Results and Comparison

The subcircuits were tested individually by NOVA command (COM) files. The COM files were previously written to test the chip at the transistor level. In each test case, the identical COM file was used to test both the behavioral model and the transistor model in order to obtain accurate as well as reliable results. Refer to [9].

The output results were logically identical for both the behavioral model and the transistor model. However, the timing results when the functional model was used did differ from the timing results of transistor model. Table 5 compares the engine stats which is the amount of CPU time to execute the simulate command for the two modeling method. Table 6 compares the total lines of code for the two modeling method for the subsections of the USE.

The behavioral model of the Mapper circuit simulated faster than the previous transistor level simulation and the code was shorter. Thus, a circuit of complex mathematical behavior would be more suited for a behavioral model than a transistor level. The results obtained from SIGFIFO simulation indicates that, for a relatively complex circuit of simple overall function, behavioral model simulation, simulates much faster and the code is shorter. Surprisingly, the results of simulation for CBITXX bitslice indicated that, for simulating a subsection of a complex section, it would be best to use transistor level simulation. ADDX, a simple arithmetic circuit, simulated slightly faster when behavioral model simulation was used. Finally, DLYREG, an example of an array of registers simulated faster by behavioral model simulation.

Table 1: The results of behavioral modeling versus transistor modeling for Subsections of USE.

Engine Stats	Behavioral Model	Transistor Model
MAPPER	0.4 sec	1.7 sec
SIGFIFO	0.1 sec	6.3 sec
CBITXX	17.7 sec	15.1 sec
ADD8	14.0 sec	15.1 sec
ADD9	12.7 sec	15.1 sec
DLEYREG	0.3 sec	1.9 sec

Table 2: Code comparisons between behavioral modeling versus transistor modeling for Subsections of USE.

Subsection	Behavioral Model	Transistor Model
MAPPER	211 lines	540 lines
SIGFIFO	154 lines	761 lines
CBITXX	499 lines	268 lines
ADD8	351 lines	268 lines
ADD9	382 lines	268 lines
DLEYREG	110 lines	82 lines

Table 3: The comparison of behavioral model simulation time to the transistor model simulation.

	slowest				fastest
Circuit	CBITXX	ADDX	MAPPER	DLYREG	SIGFIFO
Relative Speed	1.17 times slower	1.07 times faster	4 times faster	6 times faster	63 times faster

Table 4: The comparison of behavioral model code to the transistor model code.

	Longest				Shortest
Circuit	CBITXX	ADDX	DLYREG	MAPPER	SIGFIFO
Relative length	1.8 times longer	1.4 times longer	1.3 times longer	2.5 times shorter	5 times shorter

Summary

In this paper several subcircuits of the Universal Source Encoder (USE) were tested using behavioral models and the results were compared with those of transistor models. The subcircuits were chosen such that they would represent a wide variety of circuit types, allowing the associated speed up of each circuit type to be studied. The software package used in this experiment is a versatile simulator, NOVA, developed at the NASA Space Engineering Research Center.

In general, a design engineer can benefit from behavioral models because they are faster simulation models, easy to program and easy to use. Moreover, these models would relieve the designer from doing detailed logic design for every piece of circuitry, thereby increasing the flexibility of the design and reducing the design cycle time. Surprisingly, however, the behavioral models for certain circuit types actually simulate slower than the transistor level models. The results obtained from this experiment indicate that design time can, therefore, be reduced only when behavioral models are used to simulate certain types of circuits.

The experiments conducted, illustrated that behavioral models produced the best results for a large block of circuitry, circuits of complex mathematical function and circuits consisting of arrays of identical simple functions. The basic arithmetic circuit investigated, demonstrated only a marginal advantage from behavioral modeling and finally, behavioral models produced the worst results for a bit sliced subcircuit within a complex circuit.

Tables 7 and 8 illustrate a comparison between the results of behavioral model and transistor model simulations. It summarizes the circuit types for which behavioral models provide the best return on the engineering investment.

5 Acknowledgements

Microelectronic Research Center, at the University of Idaho, supported this thesis with either grants or equipment donations. The authors would like to express their appreciation to Jack Venbrux and Lowell Miles who helped extensively in developing the algorithms.

References

[1] L. Miles, P. Prins and K. Cameron, "NOVA: A New Multi-level Logic Simulator", NASA Space Engineering Research Center Symposium on VLSI Design, Moscow, ID, University of Idaho, 1990, pp 4.1.1-4.1.13.

[2] J. Venbrux, Universal Source Encoder-USE, Moscow, ID, University of Idaho, July, 1991.

[3] J. Venbrux, Engineering Documentation notes on the USE, Moscow, Idaho, University of Idaho, 1990.

[4] J. Venbrux, N. Liu, P. Vincent and R. Merrell, "A Very High Speed Lossless Compression Decompression Chip Set", JPL Publication 91-13, Jet Propulsion Laboratory, JPL Publication office, Mail stop 11-130, Pasadena, California 91109, July 15, 1991.

[5] J. Venbrux, P. Yeh, N. Liu, "A VLSI Chip Set for High Speed Lossless Data Compression", *IEEE Transaction on Circuits and Systems for Video Technology*, vol 2, no. 1, March, 1992.

[6] Mentor Graphics. "An Introduction to Digital Simulation", Part No.20948, 1989.

[7] NASA Space Research Center for VLSI Systems Design, "BOLT SHIVA Generator Users Manual", MRC, University of Idaho, Moscow, Idaho.

[8] NASA Space Research Center for VLSI Systems Design, University of Idaho, Version 0.17, February, 1991.

[9] N. Madani, " Behavioral Model Simulation Studies For the Universal Source Encoder", University of Idaho, Moscow, ID, May 1993.

This research was supported in part by NASA under Space Engineering Research Center Grant NAGW-3293.