

NASA-CR-194590

NAG-W-3293

1990
NASA

II

NASA Space Engineering Research Center
Symposium on VLSI Design

(NASA-CR-194590) THE 2ND 1990 NASA
SERC SYMPOSIUM ON VLSI DESIGN
(Idaho Univ.) 467 p

N94-71098
--THRU--
N94-71131
Unclas

29/33 0191570

**2nd NASA SERC Symposium
on VLSI Design 1990**

**University of Idaho
Moscow, Idaho**

November 6, 1990

Session 1
Featured Presentations

Chairman: Gary Maki

Welcome to the second annual NASA Symposium on VLSI Design. This symposium is organized by the NASA Space Engineering Research Center at the University of Idaho 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 data system handling needs. The symposium provides insights into developments in VLSI which can be used to increase the performance of data systems.

The NASA SERC is proud to offer, at its second symposium on VLSI design, presentations by an outstanding set of individuals from national laboratories and the electronics industry. These featured speakers share insights into next generation advances that will serve as a basis for future VLSI design. Questions of reliability in the space environment along with new directions in CAD and design are addressed by the featured speakers.

Interest in the conference has increased with 36 papers included in this year's proceedings. In addition to the 11 featured papers, 25 others have been accepted in 6 categories for inclusion in the proceedings and for presentation at the symposium. National Laboratories agencies are represented by Langley Research Center, Goddard Space Flight Center and the Jet Propulsion Laboratory. Private industry is represented by Hewlett Packard-CTG, Hewlett Packard-DMD, Bonneville Microelectronics, Gould AMI, Boeing Electronics, Mentor Graphics, The Mathworks Inc., Advanced Hardware Architectures and International Microelectronic Products. Education is represented by Montana State University, Washington State University, Duke, Moscow High School and the University of Idaho.

There are individuals whose assistance was critical to the success of this symposium. Barbara Martin worked long hours to assemble the conference proceedings. Judy Wood and Kelli Grosse did an excellent job of coordinating the many conference activities. Sterling Whitaker played a key role of organizing the symposium digest and many other related activities. The efforts of these professionals were vital and are greatly appreciated.

I am encouraged by the growth we have experienced in this year's symposium and look for suggestions that will allow a better symposium next year. I hope you enjoy your stay in Moscow, Idaho and I extend an invitation to visit MRC research laboratories during the symposium.

Gary K. Maki

Table of Contents

Session 1 – Featured Presentations

Chairman: Gary Maki

Space Qualified Microprocessors H. Benz and P. Hayes	1.1
Reliability in CMOS IC Processing R. Shreeve, S. Ferrier, D. Hall and J. Wang	1.2
A High Speed CCSDS Encoder for Space Applications S. Whitaker and K. Liu	1.3
SEU Hardening of CMOS Memory Circuits S. Whitaker, J. Canaris and K. Liu	1.4
Assessing the SEU Resistance of CMOS Latches Using Alpha-Particle Sensitive Test Circuits M. Buehler, B. Blaes and R. Nixon	1.5
Costs of Becoming a Commodity R. Foote	1.6

Session 2 – Reliable Circuit Design

Chairman: Jon Gibson

Reliable VLSI Sequential Controllers S. Whitaker, G. Maki and M. Shamanna	2.1
Efficient Design of CMOS TSC Checkers A. Biddappa, M. Shamanna, G. Maki and S. Whitaker	2.2
A Fail-Safe CMOS Logic Gate V. Bobin and S. Whitaker	2.3
Supply Current Diagnosis in VLSI J. Frenzel and P. Marinos	2.4

Session 3 – Signal Processing

Chairman: Joseph Feeley

- A VLSI Implementation for Synthetic Aperture Radar Image Processing** 3.1
A. Premkumar and J. Purviance
- VLSI Correlator Chip for Space-Borne mm-wave Radiometer Spectrometers** 3.2
K. Chandra, J. Tarsala, H. Pickett and W. Wilson
- A High Speed CMOS Correlator** 3.3
J. Canaris and S. Whitaker
- Frequency Domain FIR and IIR Adaptive Filters** 3.4
D. Lynn

Session 4 – VLSI Design

Chairman: Kel Winters

- NOVA: A New Multi-Level Logic Simulator** 4.1
L. Miles, P. Prins, K. Cameron and J. Shovic
- Path Programmable Logic: A Structured Design Method for Digital and/or Mixed Analog Integrated Circuits** 4.2
B. Taylor
- Automated Synthesis of Sequence Invariant State Machines** 4.3
D. Buehler, S. Whitaker and J. Canaris
- CMOS Output Buffer Wave Shaper** 4.4
L. Albertson, S. Whitaker and R. Merrell

Session 5 – Featured Presentations

Chairman: Jean Teasdale

Characteristics of a Semicustom Library Development System M. Yancey and R. Cannon	5.1
GaAs VLSI for Aerospace Electronics G. LaRue and P. Chan	5.2
New Directions in CAD Tools S. Peterson	5.3
High Frequency Integrated MOS Filters C. Peterson	5.4
NASA Institute Workshop for Secondary Physics Teachers L. Volkening	5.5

Session 6 – Neural Networks and VLSI Telemetry

Chairman: Gary Maki

Neural Net Diagnostics for VLSI Test T. Lin, H. Tseng, A. Wu, N. Dogan and J. Meador	6.1
Identification of the Connections in Biologically Inspired Neural Networks H. Demuth, K. Leung, M. Beale and J. Hicklin	6.2
Auto-Programmable Impulse Neural Circuits D. Watola and J. Meador	6.3
High Performance VLSI Telemetry Data Systems J. Chesney, N. Special, W. Horner and S. Sabia	6.4

Session 7 – Systems

Chairman: John Purviance

- | | |
|--|-----|
| Data Storage Technology Comparisons
R. Katti | 7.1 |
| Digital Control of Magnetic Bearings in a Cryogenic Cooler
J. Feeley, A. Law and F. Lind | 7.2 |
| A Burst-Correcting Algorithm for Reed Solomon Codes
J. Chen and P. Owsley | 7.3 |
| Minimal Test Set for Stuck-at Faults in VLSI
M. Shamanna and S. Whitaker | 7.4 |

Session 8 – VLSI Processors

Chairman: Roy Foote

- | | |
|--|-----|
| Application Specific Serial Arithmetic Arrays
K. Winters, D. Mathews and T. Thompson | 8.1 |
| ACE: Automatic Centroid Extractor for Real Time Target Tracking
K. Cameron, S. Whitaker and J. Canaris | 8.2 |
| Vertical Bloch Line Memory
R. Katti, J. Wu and H. Stadler | 8.3 |
| High Performance Pipelined Multiplier with Fast Carry-Save Adder
A. Wu | 8.4 |

Session 9 – Included Papers

- | | |
|--|-----|
| Implications of Tracey's Theorem to Asynchronous Sequential Circuit Design
S. Gopalakrishnan, G. Kim and G. Maki | 9.1 |
|--|-----|

Space Qualified Microprocessors

H. Benz, and P. Hayes
NASA Langley Research Center
Information Systems Division
Hampton, Virginia 23665

Abstract- A general review of the current state of the art of spacecraft embedded microprocessors is presented. The system requirements placed on embedded processor hardware are discussed, including the importance of size, weight, power, performance, reliability, fault-tolerance and the Earth's natural radiation environment. The capabilities of currently used spaceborne microprocessors and directions and enhancements of known developmental programs are discussed.

Reliability in CMOS IC Processing

R. Shreeve, S. Ferrier, D. Hall and J. Wang
 Hewlett Packard
 Circuit Technology Group
 Corvallis, Oregon 973330

Abstract - Critical CMOS IC processing reliability monitors are defined in this paper. These monitors are divided into three categories: process qualifications, ongoing production workcell monitors, and ongoing reliability monitors. The key measures in each of these categories are identified and prioritized based on their importance.

1 Introduction

IC process reliability starts with a clear description of the entire IC process from IC design through final shipment (Figure 1). In this flow the original development of the IC process, packaging process, test process, and shipping process are shown to the left of the main manufacturing process flow. New processes are expected to meet minimum reliability requirements. These requirements are typically referred to as new process qualification requirements.

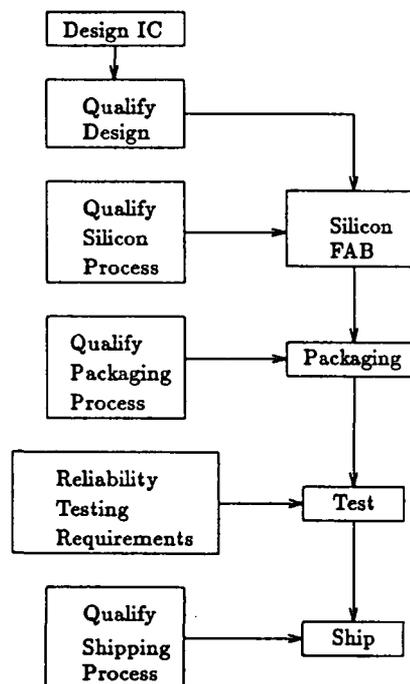


Figure 1: IC process flow

Each of the manufacturing processes like wafer fabrication can be broken down into several smaller processing steps. These steps will be referred to as workcells throughout

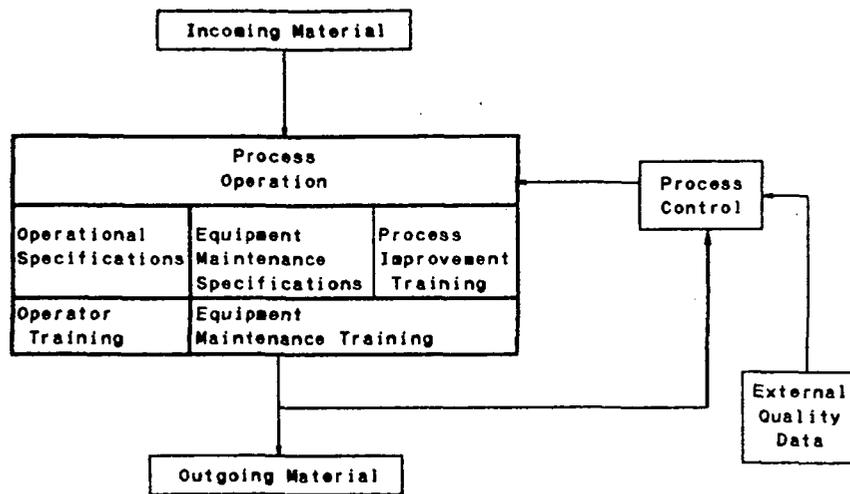


Figure 2: Workcell Description

this paper. Figure 2 shows a general process description of the workcell. The workcell is composed of three major parts: the process operation, internal workcell process control, and external feedback to the workcell. This paper carefully reviews both internal and external process control feedback for key CMOS IC workcells. This feedback is critical to the continuous improvement of the workcell process. These improvements directly affect the material consistency.

Ongoing reliability strife testing plays a key role in continuously improving the reliability of current and future IC processes. Strife testing unlike qualification testing is designed to produce IC failures through excessive environmental stress. Ongoing reliability strife testing is performed at least quarterly on material from released manufacturing processes. This testing has two purposes. First, it is intended to provide feedback so that the overall IC strength can be improved. Second, this testing provides a larger statistical basis for evaluating the consistency of the process (cumulative sample sizes for a single strife test are typically in the thousands).

This paper reviews each of these three areas in much greater detail. A commitment to continuous process improvement is assumed to be a basic operational methodology. IC reliability relies on two key components: material strength, and material consistency. Material strength refers to the ICs capability to resist degradation over time. Typically this degradation results from temperature, humidity, current flow, high voltage gradients, or mechanical stress. Material consistency refers to the ability to make each part exactly the same. IC consistency varies because of normal process variations or unexpected process exceptions (particles typically fall into this category).

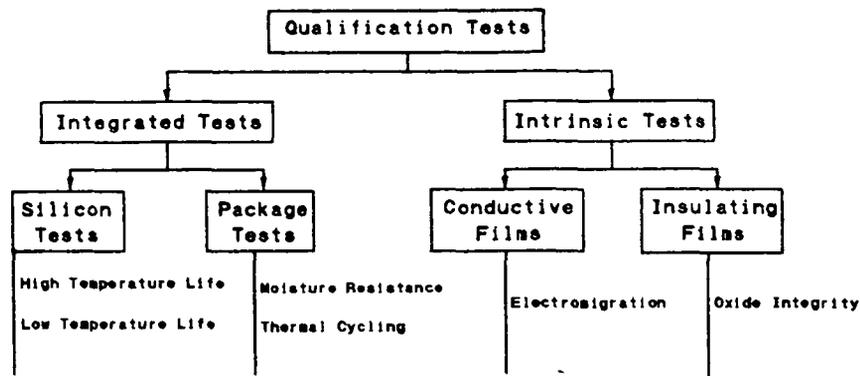


Figure 3: Process Qualification Tests

2 Qualifications

Qualifications are designed to set minimum expectations on the initial IC process. As a result, inherently weak processes are prevented from moving into manufacturing by these qualification standards. Qualification tests can be classified as either intrinsic or integrated. Intrinsic tests directly measure the intrinsic strength of specific films on the silicon die. Integrated tests measure the reliability of the entire packaged IC. Integrated tests typically verify acceptable interactions between different materials. Figure 3 shows the purpose of different qualification tests.

Integrated testing is usually given the greatest importance since it evaluates the reliability of the entire IC rather than one or two elements. However, intrinsic testing is required to supplement integrated testing because of limitations on the stress which can be applied to an integrated system.

High temperature operating life is an example of an integrated qualification test. High temperature operating life testing operates the part at junction temperatures significantly above maximum operating temperatures for extended periods (6 weeks). These higher temperatures will cause defective parts to fail in one tenth to one hundredth the time required under normal operating conditions.

Four key factors define the potential value of integrated qualification tests. These factors are listed in order of importance below:

1. Electrical Testing
2. IC Vehicle
3. Sample Sizes
4. Environmental Conditions

Specifications of qualification tests normally focus on the environmental conditions. However, our experience has shown that this factor is actually less significant than the other three factors. In other words it is typically easy to pass harsh environmental conditions if

1.2.4

the test program is compromised, the IC vehicle incompletely designed, or sample sizes of 10 to 50 parts are used.

2.1 Electrical Testing

Electrical Testing is designed to detect when the part stops functioning. Functional testing has been used for many years to detect these failures. The creation of functional vectors is typically based on a stuck-at-fault model. This model assumes that a failure is characterized by a node stuck at either 0 volts or 5 volts on the IC. In practice this type of failure mechanism is extremely rare. Typical failure mechanisms exhibit leakage current to the supplies or to an adjacent line. In both cases huge leakage currents are required to actually induce a stuck-at-fault failure. Long before the leakage induces a stuck-at-fault failure it will create reliability degradation which is the real failure mechanism in the field. Hence, it is extremely desirable to detect these low level leakages directly on the IC. This is accomplished by implementing a static current test. This test places all nodes on the IC at alternating states (0 volts, 5 volts) and then measures the leakage currents on the supply lines. All nodes on a CMOS IC are connected to either ground or Vdd through a transistor that is turned on. Hence, the leakage current between adjacent nodes can be directly detected at the supply. Useful static current measurements can be made at any level below 10 μA . Typical measurements should be in the 1 μA range to produce the appropriate sensitivity to defects. Measurements in the 100nA range are limited not by CMOS process capabilities but rather by electrical test hardware capability. For assessing silicon process reliability this single measurement is far more important than the other qualification criteria discussed.

Packaging process qualifications rely heavily on detecting defects in the silicon to package connection. As a result, I/O leakage current testing is a key measurement for detecting shorts. In addition, I/O conductivity measurements should be made to detect discontinuities at the silicon-package interface.

2.2 IC Vehicle

The IC vehicle is the part on which qualification testing is performed. Usually two different IC vehicles are required to qualify a new process. One vehicle is designed to optimize the sensitivity of the part to silicon degradation. While the other vehicle is designed to optimize the package to silicon thermal mechanical mismatch.

The silicon vehicle is intended to optimize the sensitivity of the IC layout to processing deviations (variations & exceptions). As a result, this device should be as dense as possible using minimal spacings between devices. The vehicle as a whole should dissipate extremely low standby currents so that the maximum defect sensitivity can be achieved. The design of the I/O pads should possess good ESD immunity to prevent unrealistic defects caused by electrical noise in the environmental system. Finally, it is critical that all nodes on the vehicle can be tested, exercised by the environmental test system, and failures can be mapped to the physical site of the defect. At HP we typically use an SRAM part for this

vehicle.

The silicon vehicle is used for high temperature operating life tests and moisture resistance testing. Life testing is designed to accelerate silicon defects. Moisture resistance testing accelerates both silicon and package corrosion. Silicon corrosion is directly related to the layout of the die. Layouts with tight geometries create the most difficult topologies for passivation coverage. Particulate on the die further complicate the difficulty of passivation coverage. Hence, this is the most sensitive vehicle for these two tests.

The package test vehicle is intended to optimize the mechanical stresses induced as the temperature is increased. This is the key vehicle used to qualify new packaging processes. The mechanical stresses result from differences in the coefficient of expansion between packaging and silicon materials. These stresses are optimized by creating large die and placing them in the largest package within a packaging family (i.e. package family= PDIPs, PLCCs). The package test vehicle should also be designed to incorporate direct measurements of the mechanical stress. This increases the sensitivity of the part to failure induced by stress. Corrosion structures should be incorporated to detect failures induced by the combination of moisture and ionic contaminants. Finally, the IC should be designed with a large number of I/O pads. Both the number of bonds on a part and spacing between bonds can be optimized to provide the worst case environment for reliability testing.

The package test vehicle is used for temperature cycling tests and pressure pot testing. Thermal shock, temperature cycling, and solder resistance tests are examples of different temperature cycling tests. The large size of the package test vehicle and the built in mechanical stress monitors optimize the sensitivity of this part to temperature cycling induced failures. The corrosion structures and die size optimize its sensitivity to corrosion during pressure pot testing.

2.3 Sample Sizes

Significant sample sizes are critical to detecting failures during integrated qualification tests. Parts with the same defect can degrade at varying rates. In addition, much of the electrical testing is designed to identify parts that have failed but not all of the parts that have degraded. As a result, in a few cases not all of the defective parts result in failure. This drives the need for significant sample sizes. At the very least these tests should start with sample sizes of more than a hundred parts for each environmental test.

2.4 Environmental Conditions

Most environmental tests can be classified into one of three categories:

- Life tests
- Moisture tests
- Thermal cycling tests

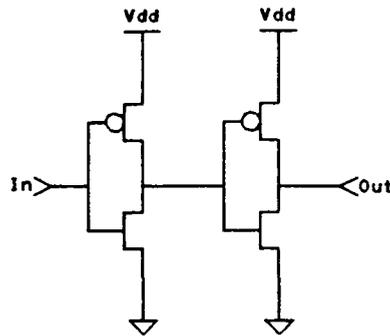


Figure 4: CMOS Inverters

Each of these tests are characterized by the environmental conditions and the operational state of the part. These parameters are consistently less important than the other issues already discussed.

High temperature and low temperature life tests are intended to primarily accelerate silicon defects. The acceleration in high temperature life testing is determined by the junction temperature of the device. Higher temperatures produce higher stresses on the part. Typical CMOS junction temperatures during high temperature life testing are 150C. During life testing the part should be exercised with a set of vectors that simulate normal IC operation. This simulation should satisfy two objectives:

1. Exercise all areas of the IC
2. Exercise the most sensitive portion of the circuit 90% of the time

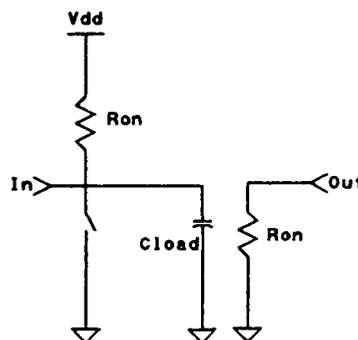


Figure 5: CMOS inverters equivalent circuit

Exercising all areas of the IC optimizes the chances of detecting unexpected problems on the IC. Figure 4 shows a typical CMOS inverter driving another inverter. Figure 5 shows the idealistic equivalent of this circuit. It is obvious from Figure 5 that current flows only when the inverter switches from one state to another. In many cases the cumulative current flow is what causes defects to become failures. The implication for reliability testing is that a circuit must be continuously exercised to identify reliability weaknesses. This is very difficult if the only objective is to exercise the entire IC. As a result, it is necessary

Test Name	Electrical Bias	Pressure (Atm)	Humidity %RH	Temperature (C)	Time (Hours)	Sample Size
65/90	Yes	1	<90%	65	240	105
85/85	Yes	1	85%	85	1000	105
Pressure Pot		2	95%	125	240	50
HAST	Yes	2	95%	125	168	105

Table 1: Summary of Moisture Resistance Tests

to select one small area of the IC that can be continuously exercised a high proportion of the time.

Several different types of moisture resistance tests are summarized in Table 1. Each of these tests use a combination of moisture and temperature to activate reliability failures. 65/90 is the only test designed to cycle the temperature and relative humidity. The purpose of this cycling was to drive moisture into cavity packages. Our experience with this test has demonstrated that it is unlikely to accelerate defects to failure. As a result, HP prefers to use 85/85 testing. Relative humidity is typically used to describe these tests. However, a much better measure is the partial pressure of water. This measure directly describes how much moisture is present in the chamber. Figure 6 shows a graph of partial pressure (for water) at various temperatures when the air is fully saturated (100% RH). Reviewing the 65/90 and 85/85 operational points on this curve illustrate that relative humidity values are quite misleading. The 85/85 test contains twice as much moisture as 65/90 even though the relative humidity is lower.

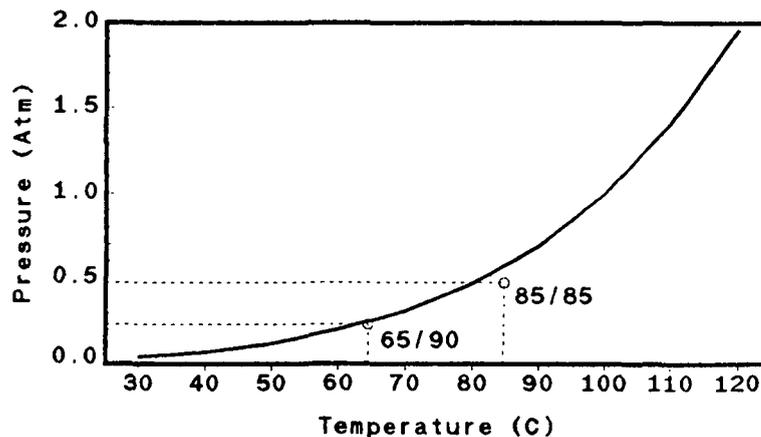


Figure 6:

Thermal cycling tests are designed to detect mechanical stress induced failures. The mechanical stress is a direct result of the differences in the coefficient of expansion for different materials. Figure 7 summarizes several different types of thermal cycling tests. All of these tests should be performed using the largest possible die and package size. Since this produces the worst case internal stress. HP generally considers thermal shock to be the worst case test because the fluorocarbon liquid forces the most rapid tempera-

ture change in the part. The soldering process tests were originally designed to confirm part compatibility with the board assembly processes. However, solderability tests offer a different temperature profile which can result in different failure mechanisms.

Name	Phase of Medium	Low Temperature	High Temperature	Number of Cycles	Dwell Time
Thermal Shock	Liquid	- 55 deg	+ 125 deg	200	5 Min.
Temperature Cycling	Air	- 55 deg	150 deg	500	5 Min.
Wave Solder	Air	25 deg	260 deg	1	10 Min.
Vapor Wave Solder	Air	25 deg	215 deg	4	1 Min.
IR Soldering	Air	25 deg	215 deg	4	1 Min.

Table 2: Summary of Thermal Cycling Test

2.5 Intrinsic

Intrinsic tests are designed to measure the intrinsic film strengths on the silicon die. Two types of films exist on a silicon wafer:

- Conductive
- Insulating

Metal layers are typical conductive films. Gate oxides, intermetal dielectrics, and passivation films are common insulators. Two special intrinsic tests are performed on these films to assure that they will continue to operate throughout the rated lifetime of the product:

- Electromigration
- Oxide Integrity

The verification of these film properties is rarely possible during integrated testing because sufficient stresses can not be applied to the part. In electromigration testing the stress is increased by applying high current densities. Oxide integrity increases the stress on nonconductive films by rapidly ramping the voltage across the film.

Electromigration testing is designed to accelerate metal migration failures. However, it can also be a useful test for accelerating failures from stress migration. In addition, this test can be used to accelerate the failure of intermetal connections (vias) or metal to substrate connections (contacts).

Oxide integrity tests are designed to measure the potential breakdown voltage of oxide films. These breakdown voltages are directly affected by weak film quality or particle contamination of the films. In addition, to normal gate oxide testing the intermetal dielectrics also need to be tested.

3 Workcell Processes

Individual work cells are the building blocks of the production process. Figure 2 shows a general description of a workcell. This section of the paper focuses on the process control monitors for each of the major workcells. This control is necessary to assure consistency from lot to lot. The process operation consists of operational personnel, equipment, in-flow, and outflow material. This paper assumes that operational specifications, operator training, equipment maintenance, specifications, equipment maintenance rate tracking, and process change training are parts of the process operation in each workcell.

Figure 1 shows the overall IC process flow. This flow consists of five production processes:

1. Layout
2. Silicon Fabrication
3. IC Packaging
4. Electrical Testing
5. Shipping

This section reviews the major workcells for each of these production processes.

3.1 Layout

The layout of the IC will ultimately determine the reliability of a specific IC design. Three key reliability issues must be considered in the layout:

1. Compliance to process layout rules
2. Electromigration
3. I/O pad protection

Layout rules should be clearly documented. In many cases layout programs and libraries already exist that prevent potential violations of these rules. In addition, a design rule check program is usually run to verify that the physical layout does meet the design rule requirements.

Electromigration is typically handled by providing margin in the initial electromigration rules. Libraries of verified designs provide additional protection from electromigration violations. Some electromigration checkers do currently exist but most are still in the development phase. As a result, it is crucial that the designer identify where potential electromigration violations could exist. This is usually a short list because of the availability of verified libraries and autorouters.

The I/O pads interface to the external world. As a result, they need to protect the IC from transient high voltage or current abuse. This is accomplished by incorporating ESD

1.2.10

protection circuitry and appropriate guardrings for latchup. All pads should be reviewed before the IC artwork is released for mask production. HP also performs ESD and Latchup testing on the first parts fabricated in a new design to verify their protection capabilities. Different ESD test equipment can produce significant differences in the measured ESD protection of an I/O pad. This is especially true with machines that conform to older versions of MIL-STD 883C [1].

3.2 Silicon Fabrication

The following silicon fabrication workcells are reviewed in this section:

- Diffusions
- Gate oxides
- Metalization
- Intermetal dielectrics
- Passivation

Table 3 summarizes the key reliability process monitors for each of these process steps.

Process Step	Process Controls
Diffusions	CD's, dose, temperature, resistivity
Metalization	Thicjness, width, grain structure, sheet resistivity
Gate Oxide	Mobile ion concentration, surface charge density, thickness
Intermetal Dielectric	Leakage currents, thickness, topography, alignment
Passivation	Coverage

Table 3: Silicon process controls for reliability

Island and well diffusion are required to produce the correct dopant concentrations and depth profiles. These issues are typically monitored by measuring the dimensional accuracy of the diffusion openings, monitoring the dose, controlling the drive in temperature, and measuring the final diffusion resistivity.

Metalized films are required to conduct electrical current from one node to another without variations in resistivity over their life. Resistivity variations could result from differences in metal width, thickness, grain structure, or sheet resistivity. As a result, it is desirable to have each of these parameters monitored as part of the metalization workcell.

Gate oxides should block current flow from the gate to the channel without degrading the electrical field. Mobile ion concentration, surface charge density, and thickness need to be monitored to assure consistency from part to part and over the lifetime of the product.

Intermetal dielectrics prevent leakage paths between adjacent metal conductors (both vertical and horizontal). Leakage currents, and thickness are two key monitors of the insulating characteristics of these materials. In addition, the film topography, and alignment to underlying layers must be monitored.

Passivation is usually the final layer deposited onto the silicon wafer. It is designed to protect the metalized layers from chemical corrosion and the transistors from chemical contamination. To provide this protection the passivation must uniformly cover all structures on the IC. Variations in the thickness of the film need to be monitored to assure film reliability.

3.3 IC Packaging

Table 4 summarizes the process steps and process controls for IC packaging. These controls are designed to assure IC packaging consistency.

Process Step	Process Controls
Leadframe	Plating uniformity, frame quality
Die Attach	Backside coverage, excess material
Bonding	Temperature, pressure, tip wear, wire quality
Molding	Temperature, pressure, injection rate
Lead forming	Tool accuracy, tool wear

Table 4: Packaging process controls for reliability

The leadframe is usually considered an incoming material. The leadframe is plated so that a wire can be bonded to the inner lead fingers. This plating must have uniform thickness and consistent purity. The leadframe is stamped or etched to create separate leads. Each of these leads need to be correctly formed and free of contamination.

The die attach step is designed to hold the silicon die to the metal leadframe for the entire life of the product. The attachment method should provide complete attachment across the entire back surface of the die. Any excessive attachment material should be carefully controlled so that it does not come in contact with the top side of the die.

Bonding is designed to make electrical connection between the die and the leadframe. Several different bonding processes exist. The bonding temperature, bonding pressure, bonding tip quality, and wire quality must be carefully controlled to assure consistency. Wire quality is defined by the wire diameter and the wire purity.

Molding is designed to protect the part from chemical and mechanical degradation. To accomplish these objectives the molding material must be uniform and dimensionally accurate. The molding temperature, pressure, and injection rate must be carefully controlled.

The final step in the packaging process is lead forming. In surface mount applications the lead alignment and planarity are critical for creating a good electrical connection to the board. The forming tool determines the results of this operation. The tool needs to meet original specification and stay within specified wear requirements.

3.4 Electrical Testing

Electrical testing is performed at both the raw wafer level and at the final packaged part level. The wafer level testing is usually critical to assuring overall wafer reliability. While package testing is designed to detect packaging induced defects.

Three different types of tests are performed at the wafer level. The first test verifies the appropriate device characteristics by measuring structures like individual transistors, contact strings, or diffusion resistances. These structures are located in the center of the scribe lines on the wafer. A few wafers are tested from each lot. If a wafer does not meet the device specifications then the entire lot is rejected.

The second wafer test is a full functional test. HP typically breaks this test into several parts:

- Open/Short I/O pin testing
- Functional testing
- I/O leakage current testing
- Supply current testing

Functional testing is usually performed at full device operational frequencies with the voltage levels adjusted to compensate for maximum temperature sensitivities. Typical functional vectors sets are expected to meet at least 95% fault coverage. A static current test is one of the supply current tests. This test is a critical element in assuring that the part is free of defects. The Qualification section of this paper explains in detail why this test is critical. One additional screen for wafer testing is called below ship limit wafer scrapping. In cases where the number of good die is less than 25% of the standard wafer yield the entire wafer should be scrapped. This prevents potentially marginal die from being shipped to the field.

The third wafer test is a visual inspection of the wafer. This screen is designed to detect gross visual defects that would affect metal or passivation quality.

Package testing consists of a full functional test of each part. In addition, lead planarity and alignment testing is performed on surface mount devices.

3.5 Shipping

The shipping process packages parts so that they will not be damaged during shipment. Three potential reliability problems must be prevented during this process:

- Electrostatic discharge damage (ESD)
- Pin planarity or alignment damage
- Excessive moisture absorption.

ESD damage can be prevented by using appropriate ESD grounding procedures while packaging and preparing parts for shipment. In addition, the internal protection structures on the part also provide protection. Regular ESD audits on the shipping area assure that the intent of ESD prevention procedures are understood and that the procedures are followed on a regular basis.

Pin planarity or alignment damage can result from poor part handling techniques. This damage can be prevented by correctly selecting shipping containers for the parts, and using appropriate part handling techniques. Workcell process control and therefore consistency are typically supported by both audits of the actual procedures used and sampling audits of the outgoing material.

Parts that absorb excessive moisture before shipment may suffer internal cracking or delaminations (popcorning) during soldering to pc boards. The best way to control this mechanism is to place parts into inventory in moisture tight packaging. Placing a moisture absorption card in each package is another step that can be taken to guarantee that material soldered onto boards does not popcorn.

4 Ongoing Strife Testing

One purpose of this testing is to identify the weakest element in the IC. Once this element is identified process improvements are developed to further increase the strength of the process. Ideally strife testing should not be a static set of tests but rather a set of tests that continuously evolve to create higher and higher part stresses. Another purpose of this testing is to increase the statistical data available on the process reliability. Running these tests on a regular basis assures that sufficient data is available to make realistic assessments of the actual product reliability. New processes should be tested on a more frequent basis than old processes to develop the statistical data base. At HP we perform strife testing every month for newer processes. After a couple years this testing is reduced to a quarterly basis.

As in qualification testing three major types of integrated tests exist:

- Operating life tests
- Moisture resistance tests
- Temperature cycling tests

To increase the part stress these tests are usually performed sequentially. The stress in operating life tests are increased by using higher supply voltages. Moisture resistance testing is almost always performed using HAST conditions (summarized in Figure 6). Thermal shock is used to induce thermal mechanical stress. These stresses are increased by cycling the parts for at least 1000 cycles.

5 Conclusions

CMOS IC reliability is determined by a combination of material strength and product consistency. Weak materials clearly result in a weak product. Inconsistent product quality will result in inherent product weaknesses that cause field failures. Qualification testing is designed to set minimum intrinsic and integrated material strength standards. Process controls are designed to assure product consistency. Products that do not meet these consistency requirements must be scrapped because they contain defects that will cause early life failures. Finally both material strength and product consistency should improve over the process life. Ongoing strife testing is designed to identify which materials possess the lowest strength and any variations in process consistency. Based on this data appropriate process improvements can be developed and implemented.

References

- [1] MIL-STD-883, "Test Methods and Procedures for Microelectronics".
- [2] HP General Silicon Specification, A-5951-7600-1, Hewlett Packard, Palo Alto, 1988.
- [3] J.M. Pimbley, M. Ghezzi, H.G. Parks, and D.M. Brown, "Advanced CMOS Process Technology," Academic Press, San Diego, CA, 1989.
- [4] S. Wolf, and R.N. Tauber, "Silicon Processing for the VLSI Era Volume 1: Process Technology," Lattice Press, Sunset Beach, CA, 1986.
- [5] M.J. Howes, and D.V. Morgan, "Reliability and Degredation," Wiley, New York, 1984.

A High Speed CCSDS Encoder for Space Applications

S. Whitaker and K. Liu

NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - This paper reports a VLSI implementation of the CCSDS standard Reed Solomon encoder circuit for the Space Station. The $1.0\mu\text{m}$ double metal CMOS chip is 5.9mm by 3.6mm , contains 48,000 transistors, operates at a sustained data rate of 320 Mbits/s and executes 2,560 Mops. The chip features a pin selectable interleave depth of from 1 to 8. Block lengths up to 255 bytes as well as shortened codes are supported. Control circuitry uses register cells which are immune to Single Event Upset. In addition, the CMOS process used is reported to be tolerant of over 1 Mrad total dose radiation.

1 General Description

This chip implements an encoder for the CCSDS standard (255,223) Reed Solomon (RS) code [1]. An RS code is a cyclic symbol error correcting code for correcting errors introduced into data during transmission through a communication channel. The CCSDS standard is a 16 symbol error correction code. The code block consists of 223 information symbols and 32 parity symbols. Each symbol is an 8 bit word. 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:

$$p(x) = x^8 + x^7 + x^2 + x^1 + x^0 \quad (1)$$

The generator polynomial is given by:

$$g(x) = \prod_{i=112}^{143} (x - \beta^i) \quad (2)$$

where $\beta = \alpha^{11}$.

The encoder represents data in the dual basis such that

$$[z_0, z_1, \dots, z_7] = [u_7, u_6, \dots, u_0] T \quad (3)$$

1.3.2

where $[z_0, z_1, \dots, z_7]$ is the symbol represented by the dual basis, $[u_7, u_6, \dots, u_0]$ is the symbol represented by the normal basis and T is the following transform matrix:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \quad (4)$$

Normal data can be derived from data represented in the dual basis using the following inverse transform.

$$[u_7, u_6, \dots, u_0] = [z_0, z_1, \dots, z_7] T^{-1} \quad (5)$$

where

$$T^{-1} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad (6)$$

A dual field is simply a different representation of the original field. The coefficients of $g(x)$ are linear operators. An operator O in the original representation of the field can be used in the dual representation by applying the following transform.

$$O_{dual} = T O_{original} T^{-1} \quad (7)$$

Additional details of the mathematics can be found in [2].

The coder circuit has data input and output ports. Data is input in a byte serial fashion at a constant rate, and is output in a byte serial fashion with a fixed one clock cycle latency. After the information bytes have been output, the 32 bytes of RS parity are appended to the data stream. The data rate for the chip is 40 Mbytes/sec when clocked at a rate of 40 MHz.

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. The interleaving depth, I , is controlled by external pins, S_0 , S_1 and S_2 .

2 Chip Operation

2.1 Initialization

Before proper circuit operation can begin, the encoder sections must be initialized and the interleave depth chosen. This is accomplished by bringing the reset input (RST) high for at least two clock pulses and setting the interleave depth control lines, S_0 , S_1 and S_2 , to the appropriate state.

S_2	S_1	S_0	I
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

At this time, it is also necessary to bring the input control (INC) inactive low to ensure no spurious messages are processed. Two clock pulses after RST is brought low, circuit operation may commence. The circuit may be re-initialized at any time but any messages being processed by the encoder section at that time will be lost. Zeros are clocked into the parity generator whenever INC is low.

2.2 Encoder Operation

Assuming the initialization sequence has been performed, encoding is performed in the following manner. INC is brought high coincidentally with the first message symbol to be encoded. It remains high while successive message symbols are clocked into the encoder on the data input bus (DI). Symbols are clocked in and out of the circuit on the rising edge of the symbol clock (CK).

INC is brought low again when the last message symbol has been clocked into the circuit. It must remain low at least $32I$ clock cycles, during which time the parity symbols will be clocked out of the circuit. This operation also fills the parity generator with zeros. If INC is held low longer than $32I$ clock cycles, zeros will appear on the the data output bus (DO). Bringing INC high after it has been low for $32I$ or more clock cycles starts the processing of the next message.

2.3 Bypass Operation

After a reset operation or after a block has been encoded and the parity read from the chip, a data bypass operation can occur. Data can flow through the encoder without being encoded by bringing the bypass input control (BIC) high coincidentally with the first byte

1.3.4

of data to be passed unprocessed by the chip. After the one clock cycle latency, the data entering on DI appears on DO and continues to pass through the chip as long as BIC remains high. While BIC is high, INC should be held low to keep the registers in the parity generator held reset.

2.4 Space Enhancement Features

The CMOS fabrication process used is reported to be tolerant of total dose radiation levels exceeding 1 Mrad [3]. In addition, the chip is designed to provide protection against Single Event Upset (SEU) in two ways. First, control memory cells are designed to be electronically tolerant of SEU's. 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.

A 16 bit shift register has been included on the chip with the input driven by the test input pin (TI) and the output driving the test output pin (TO). This test structure will enable the SEU immune memory cell to easily be tested under conditions of irradiation to verify the immunity.

3 VLSI Implementation

Full custom VLSI was used to achieve both circuit density and speed. The basic VLSI architecture implemented here is similar to a previous full custom design [4]. The additional features include interleaving, high speed operation (320 Mbits/sec), radiation hardened processing and SEU protection.

3.1 General Organization

Figure 1 shows a top level logic diagram. The chip consists of an encoder section and a test shift register. The encoder contains 32 multipliers and 32 adders which operate in parallel so that the mathematics required for the parity generation can be performed at the data input clock rate. The encoder also contains the 2048 registers (32x8x8) required to interleave the data to a maximum depth of 8. The 16 bit shift register is a test structure that will be used to verify the SEU immunity of the registers used for the control circuitry.

Data is input on the DI0-7 pins and output on the DO0-7 pins. Input data is framed by the INC control signal. Output data is framed by OUTC which is a delay of INC. When data is input to the chip, it is presented to the parity generator and also passed out the output port DO0-7. At the end of the data block, INC transitions low and the output of the parity generator passes out DO0-7. With INC low, 0's are input to the parity generator clearing out the registers. With BIC high and INC low, data flows from the DI0-7 to DO0-7 without being input to the parity generator providing a bypass mode.

3.2 Parity Generator

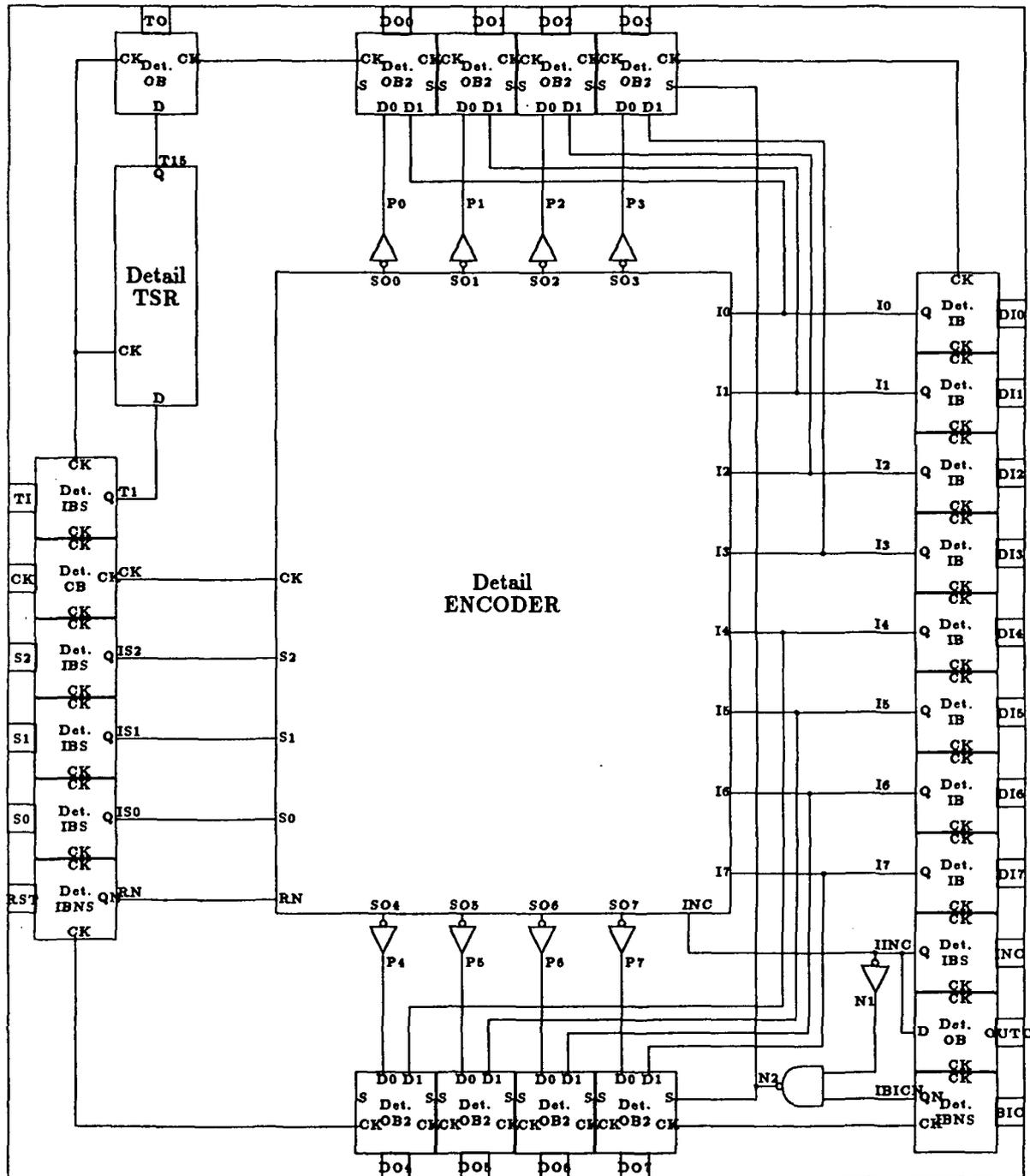


Figure 1: Top level logic diagram.

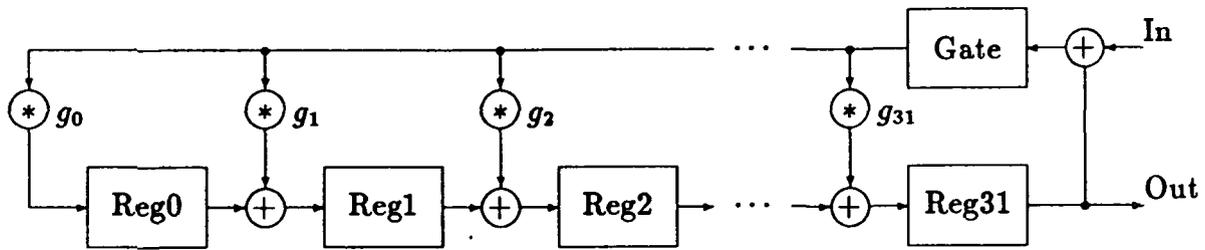


Figure 2: Parity generator block diagram.

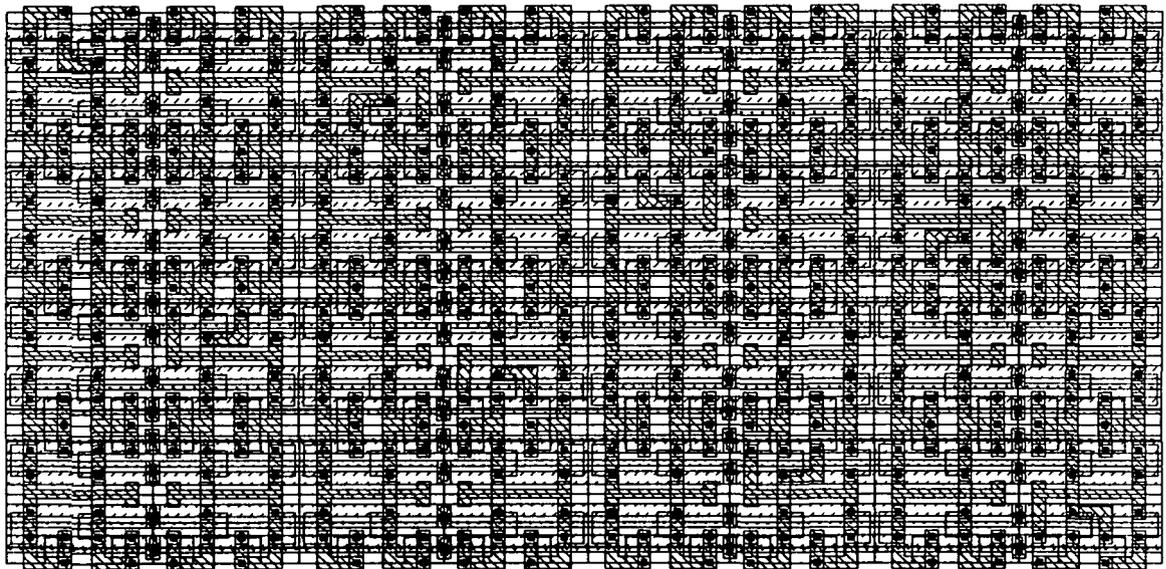


Figure 3: Layout of constant multiplier.

The parity generator was organized as a set of 32 slices. Each slice consisting of a multiply/add structure and a register stack for interleaving. Figure 2 shows a block diagram of the logic for the parity generator. Each register is a 1 to 8 bit shift register depending on the interleave depth set up during circuit initialization. The multiplier cell is a precharged exclusive or (XOR) chain. 8 of these chains form a constant multiplier. The input data word is multiplied by a constant, g_x , programmed into the multiplier as XOR cells or interconnect (ZERO) cells. The XOR cell consists of 4 NMOS transistors. The ZERO cell is a modified XOR cell which acts as an interconnect block. The layout of a constant multiplier is shown in Figure 3. The multiplication constant can be programmed with a single mask layer defining the pattern of XOR and ZERO cells in the XOR chains. For maximum speed the XOR chain is precharged from both ends. The addition function is folded into the evaluate structure for the multiplier. The XOR cell was designed in layout to consume minimum area and the registers were designed to match the pitch of two XOR cells. Half the registers were placed above the multiplier and half below.

Since the registers are twice as wide as the XOR chain, the outputs of the columns in

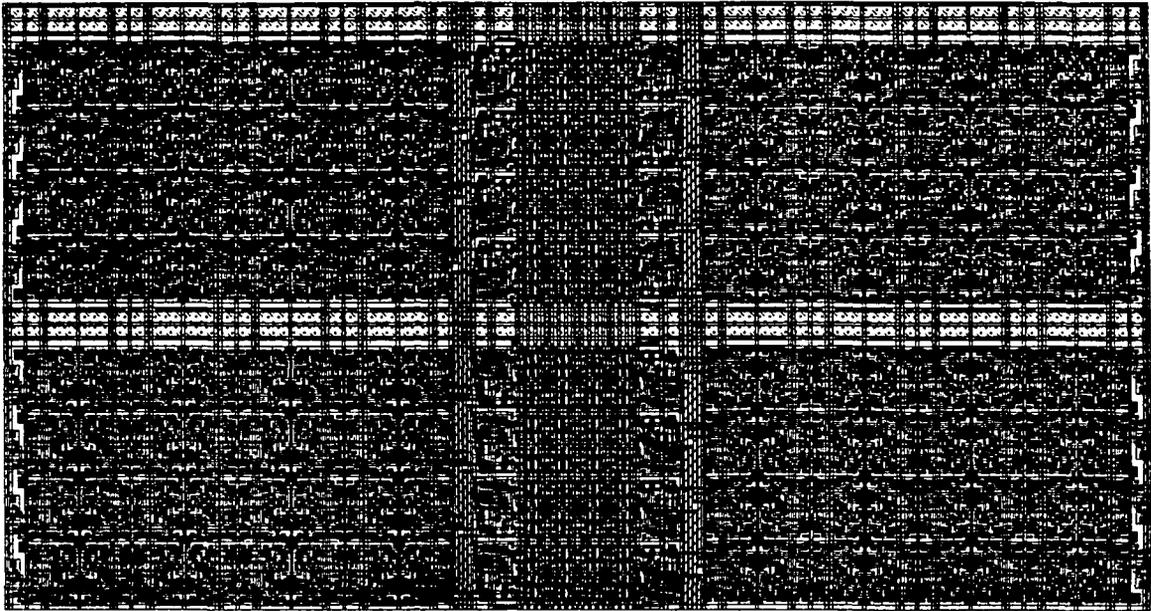


Figure 4: Layout of two adjacent slice details.

the multiplier alternate between top and bottom. The higher order nibble is output on one end of the constant multiplier and the lower order nibble on the opposite end. This requires the columns of the multiplier matrix to be rearranged such that the columns in the matrix are $[C_0C_4C_1C_5C_2C_6C_3C_7]$. Also, in order to avoid long interconnect runs between registers on the top and bottom to drive the adder inputs in the multiply/add structure, a second slice detail was drawn such that the top and bottom sections were reversed. These two slice details were then alternated in the parity core allowing connection of the adjacent slices by abutment. Figure 4 shows the layout of two adjacent slices. There is no interconnect required between any of the leaf cells. The entire structure is connected by abutment. This maximizes the speed of operation since the interconnect capacitance has been minimized.

A natural layout would place all 32 slices in a row. This would maximize the speed of operation and minimize the area required for the parity generator, but would result in a die size of approximately $10mm$ by $2mm$. This aspect ratio would be hard to accommodate in packaging and the reliability of the bond wires would be in question, especially under the stresses expected during launch. The speed was therefore compromised by folding the array in half. The control was duplicated and an interconnect bank was run from the output of Slice 15 to the input of Slice 16. The final chip layout is shown in Figure 5.

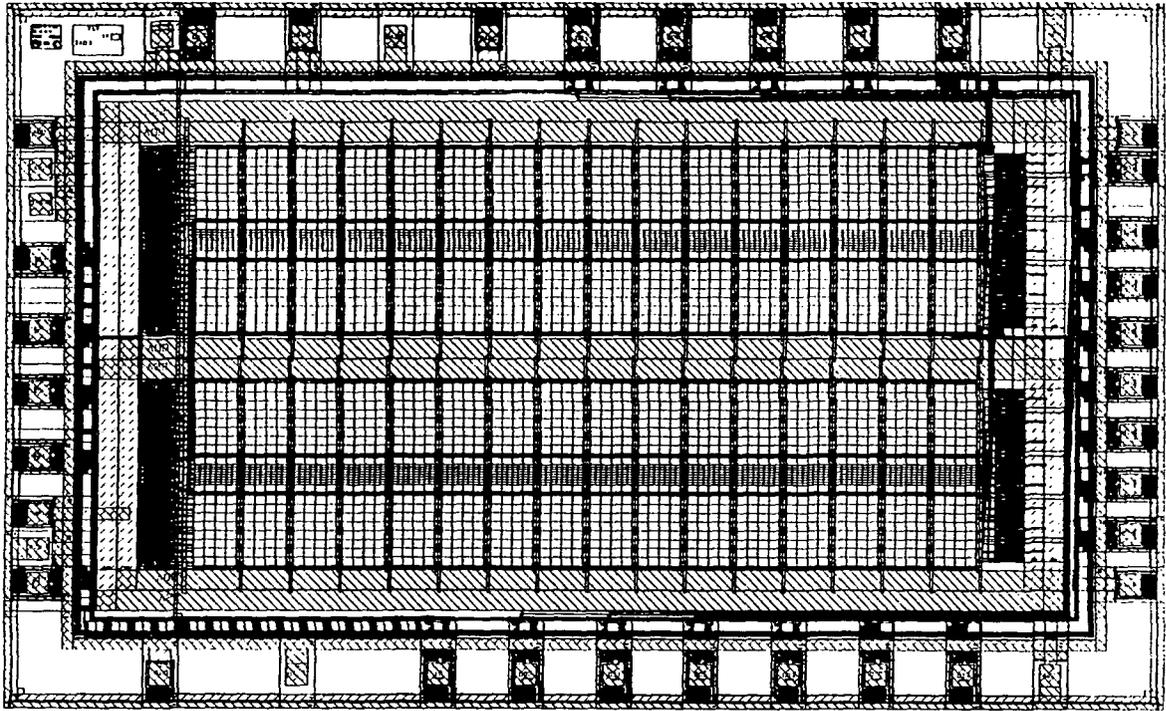


Figure 5: Chip layout.

4 Summary

A Rad Hard, SEU tolerant implementation of the CCSDS standard RS16 encoder has designed for Goddard Space Flight Center. The chip was drawn in a $1.0\mu\text{m}$ CMOS process and is being fabricated at Hewlett Packard's Circuit Technology Group. The encoder operates at a 320 Mbit/sec data rate.

Acknowledgement: This research was supported in part by NASA under grant NAGW-1406. The authors wish to acknowledge the support from Warner Miller at Goddard Space Flight Center. This chip will be commercially available as AHA 4611 from Advanced Hardware Architectures.

References

- [1] H.F. Reefs and A.R. Best, "Concatenated Coding on a Spacecraft-to-ground Telemetry Channel Performance", Proc. ICC-81, 1981.
- [2] G. C. Clark and J. B. Cain *Error Correcting Coding For Digital Communications*, New York NY, Plenum Press, 1981
- [3] CMOS34 Radiation Hardness,

- [4] G. Maki, P. Owsley, K. Cameron, and J. Shovic, "A VLSI Reed Solomon Encoder: An Engineering Approach," IEEE Custom Integrated Circuit Conference, pp. 177-181, May 1986

SEU Hardening of CMOS Memory Circuits

S. Whitaker, J. Canaris and K. Liu
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - This paper reports a design technique to harden CMOS memory circuits against Single Event Upset (SEU) in the space environment. A RAM cell and Flip Flop design are presented to demonstrate the method. The Flip Flop was used in the control circuitry for a Reed Solomon encoder designed for the Space Station.

1 Introduction

In the environment of outer space, electronic circuitry is exposed to a flux of ionized particles. If the energy level of a charged particle is high enough and that particle passes through the diffusion of a susceptible node then the contents of a MOS memory cell can be changed [1]. This is a Single Event Upset (SEU) of the integrated circuit containing the memory cell. The consequences of the SEU depend on the system function of the memory cell. Circuit design techniques which are independent of processing and which are without serious performance degradation have been reported [2]. This paper also reports a circuit design method which is process independent and maintains the performance level. With the techniques presented here, loading on the clock signal is reduced when compared with [2].

2 RAM Design

There are three fundamental concepts that can be used to design SEU immune circuitry. 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. These observations lead to the design of the RAM cell as shown in Figure 1.

The RAM cell consists of two storage structures. The top half is constructed from p-channel devices while the bottom half consists solely of n-channel devices. Transistors M2 and M4 are sized to be weak compared to M3 and M5 while M13 and M15 are sized

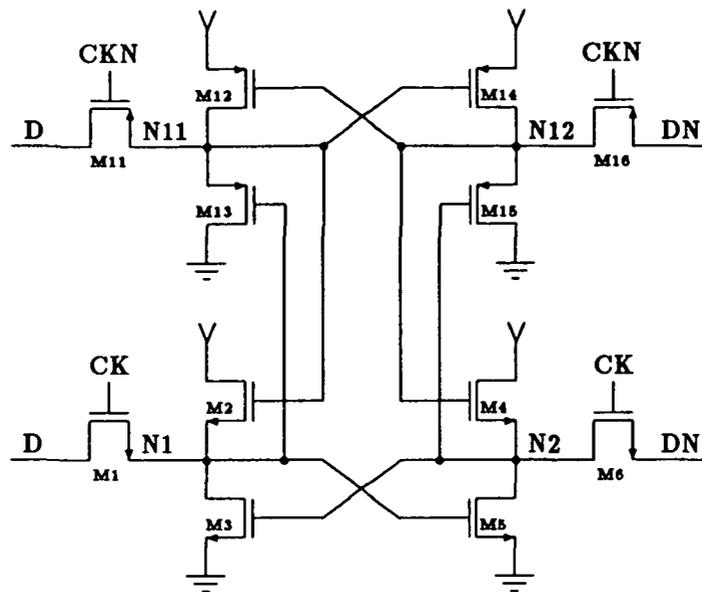


Figure 1: SEU Hardened RAM cell.

to be weak compared to M12 and M14. The other transistors are sized using the normal design considerations for a RAM cell to allow the cell to be written and read to meet the performance required.

Nodes N1 and N2 can store 0's that cannot be upset while N11 and N12 can store 1's that cannot be upset. If N11 is storing a 0 and is hit driving the node to a 1, M14 turns off but node N12 remains at a 1. M2 turns on but is weak and cannot over drive N1 keeping M13 on and restoring N11 to a 0. If N1 is storing a 1 and is hit driving the node to a 0, M5 turns off leaving node N2 at a 0. M13 turns on but is weak and cannot over drive N11 keeping M2 on and restoring N1 to a 1.

The memory cell was then designed for the desired write time and read time resulting in the device sizes shown in Table 1. The layout was drawn and is shown in Figure 2. Parasitics were extracted and SPICE simulations run to verify functionality and performance. SPICE simulations were also performed depositing $5pC$ on N11 and removing $5pC$ from node N1 verifying the cell recovery. Figure 3 shows the response of N11 during the recovery from a particle strike. The simulations were performed under worst case speed conditions ($V_{dd} = 4.3V$, $T_j = 140C^\circ$ and 3σ parameters). Figure 4 shows the response of N1 during the recovery from an SEU. Both SPICE simulations show a recovery time of only a few $nsec$. The hardness of the cell design is independent of processing and parameters. The logical feedback and ratioing of transistor strengths provides the recovery mechanism. The zero level of N11 is degraded because of the p-MOS device threshold voltage and body effect on the threshold voltage. At N1 the one level is similarly degraded in the n-MOS section. These levels are acceptable and pose no design problems internal to the cell. When reading the RAM cell, the p-MOS section will drive the data line, D, to VDD when a 1 is stored and the n-MOS section will drive DN to VSS providing rail to

Transistor	W_n	L_n	Transistor	W_p	L_p
M1	6.4	1.0	M11	6.4	1.0
M2	2.4	1.0	M12	6.9	1.0
M3	6.9	1.0	M13	2.4	1.0
M4	2.4	1.0	M14	6.9	1.0
M5	6.9	1.0	M15	2.4	1.0
M6	6.4	1.0	M16	6.4	1.0

Table 1: RAM cell device sizes.

rail operation on the cell output.

The circuitry added for SEU immunity doubles the number of transistors required by a RAM cell. The layout requires an increase of about 75% in area over the traditional 6 transistor RAM cell. The method described in [2] also doubled the transistor count, but will result in a more compact layout (only a 40% increase in area). The cell described in this paper does, however, have an advantage in the loading seen by the clock signal and the loading seen by the data drivers during a write cycle is less than that of [2]. This should result in an increase in performance. Cells implemented with these two methods using the same CMOS process need to be analyzed to verify this claim.

3 Flip Flop Design

The memory cells required by the control circuitry of the RS16 encoder [3] are D flip flops. A flip flop contains a master and a slave section. Each flip flop section is basically a RAM cell. Figure 5 shows the schematic for a flip flop section. The RAM cell outputs have been buffered using M8/M18 and M7/M17. The buffers allow Q and Q_N to have rail to rail operation and isolates the memory portion from potentially high capacitive loads. On the RS16 encoder, the control lines present capacitive loads of about $3pF$ and need to be driven by the output of the flip flop with relatively short delay times when operating at 40MHz. Table 2 lists the required device sizes to meet the internal cell load and speed requirements.

Two of the RAM sections were joined to form the flip flop of Figure 6. The control flip flops were designed to have a single clock and data input. This required inverters CK and DN. The large capacitive load seen by the output required the Q buffer. In many applications, these additional devices would not be necessary. Table 3 lists the device sizes for this additional logic. Figure 7 shows the layout of the D flip flop used on the RS16 encoder.

A shift register consisting of a string of these flip flops was added as a test circuit to the RS16 encoder to allow the SEU immunity of these cells to be easily verified. The input of the 16 bit shift register is driven by an input pad and the output of the shift register drives an output pad.

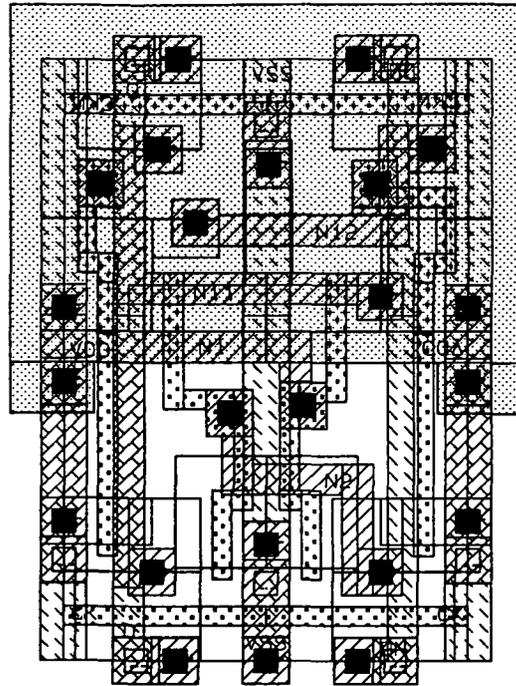


Figure 2: RAM cell layout.

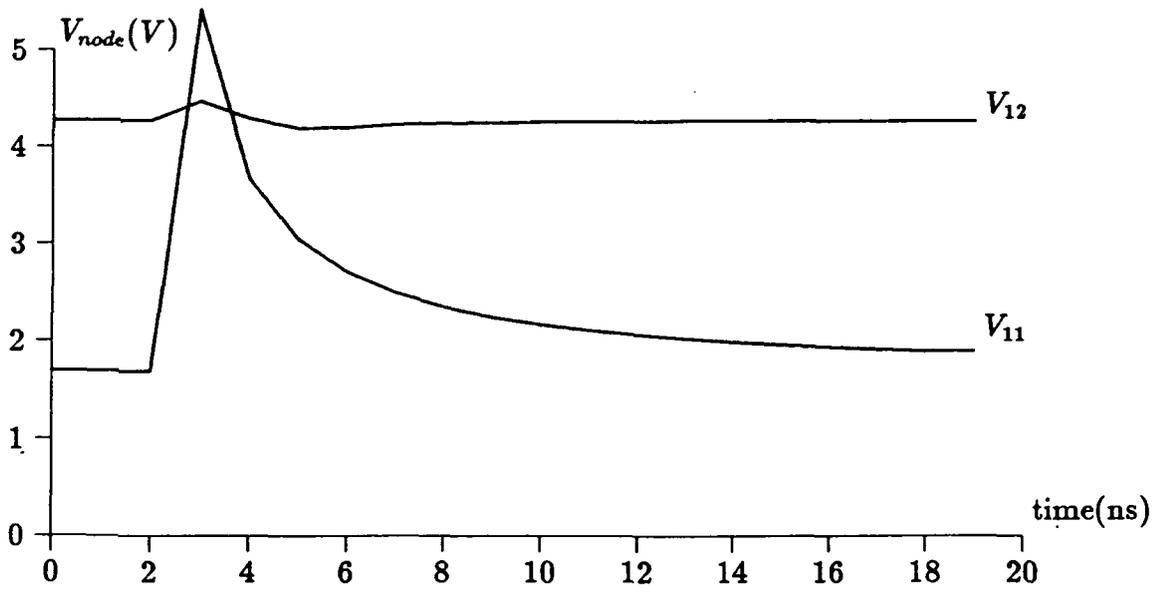


Figure 3: RAM cell N11 recovery from an SEU.

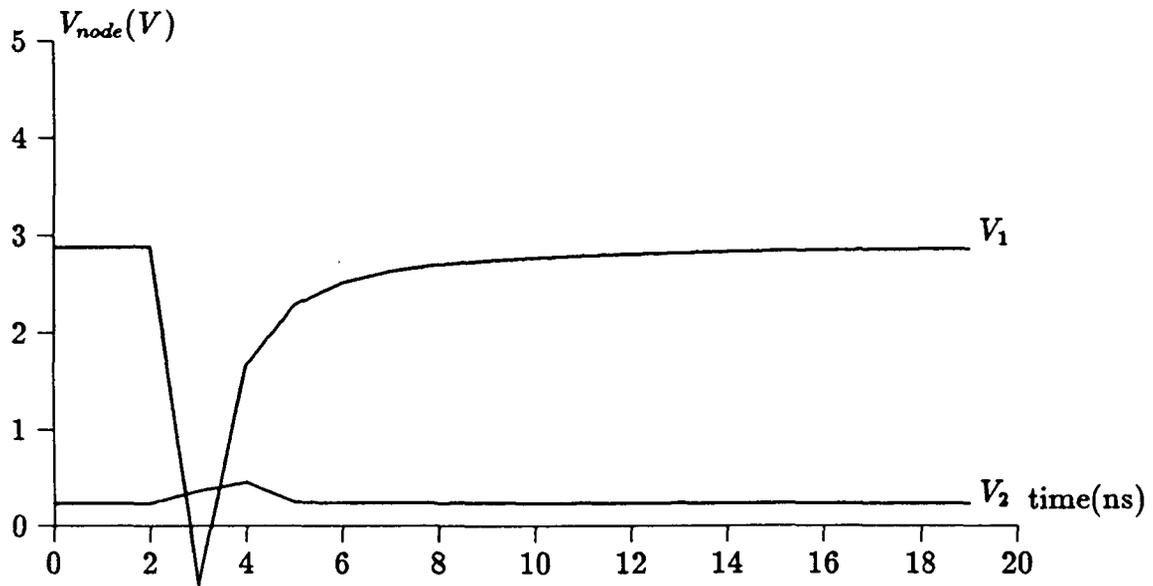


Figure 4: RAM cell N1 recovery from an SEU.

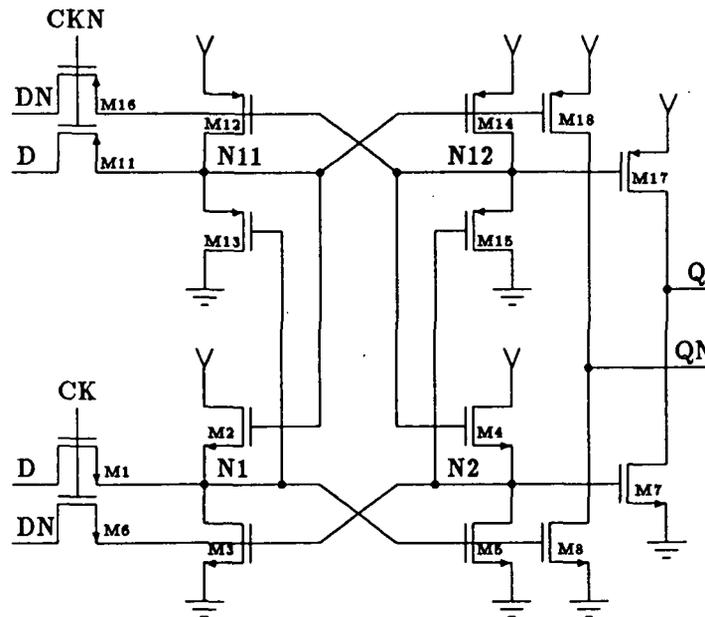


Figure 5: SEU hardened Flip Flop section.

Transistor	W_n	L_n	Transistor	W_p	L_p
M1	7.2	1.0	M11	7.2	1.0
M2	2.4	1.0	M12	7.2	1.0
M3	7.2	1.0	M13	2.4	1.0
M4	2.4	1.0	M14	7.2	1.0
M5	7.2	1.0	M15	2.4	1.0
M6	7.2	1.0	M16	7.2	1.0
M7	12.0	1.0	M17	12.0	1.0
M8	12.0	1.0	M18	12.0	1.0

Table 2: Flip Flop section devices sizes.

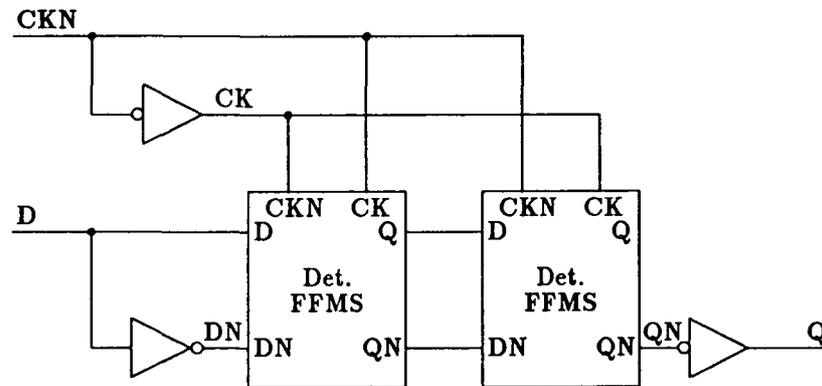


Figure 6: SEU hardened Flip Flop cell.

Node	W_p	W_n
CK	14.4	9.6
DN	12.0	6.0
Q	48.0	24.0

Table 3: Flip Flop cell device sizes.

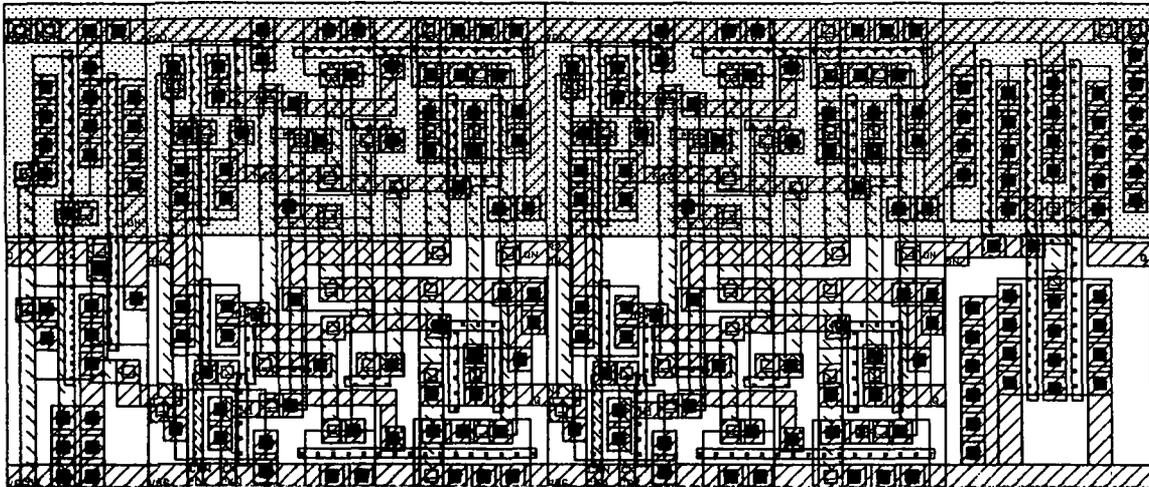


Figure 7: Flip Flop cell layout.

4 Summary

An SEU immune RAM cell has been presented which compares favorably with others in the literature. This RAM cell design was implemented as a D flip flop in the controller for a Reed Solomon encoder to be used in the Space Station. This is a preliminary report. Work needs to be performed to produce a comparison of previously reported SEU immune logic with this cell. Also when the chip fabrication is complete, the shift register will be subjected to cyclotron tests to verify the SEU immunity of this cell and results will be submitted for publication.

References

- [1] T. Ma and P. Dressendorfer, *Ionizing Radiation Effects in MOS Devices & Circuits*, New York, NY, John Wiley & Sons, 1089, Chap. 9
- [2] L. Rockett, "An SEU Hardened CMOS Data Latch Design", *IEEE Transactions on Nuclear Science*, Vol. 35, No. 6, pp. 1682-1687, Dec., 1988
- [3] S. Whitaker and K. Liu, "A High Speed CCSDS Encoder for Space Applications", 2nd NASA SERC Symposium on VLSI Design, Nov., 1990.

Assessing the SEU Resistance of CMOS Latches Using Alpha-Particle Sensitive Test Circuits¹

M. Buehler, B. Blaes, and R. Nixon
 Jet Propulsion Laboratory
 California Institute of Technology
 Pasadena, California 91109

1 Introduction

The importance of Cosmic Rays on the performance of integrated circuits (ICs) in a space environment is evident in the upset rate of the Tracking and Data Relay Satellite (TDRS) launched in April 1983. This satellite experiences a single-event-upset per day [1] which must be corrected from the ground. Such experience caused a redesign of the Galileo spacecraft [2] with SEU resistant ICs. The solution to the SEU problem continues to be important as the complexity of spacecraft grows, the feature size of ICs decreases, and as space systems are designed with circuits fabricated at non-radiation hardened foundries.

This paper describes an approach for verifying the susceptibility of CMOS latches to heavy-ion induced state changes. The approach utilizes alpha particles to induce the upsets in test circuits. These test circuits are standard cells that have offset voltages which sensitize the circuits to upsets. These results are then used to calculate the upsetability at operating voltages. In this study results are presented for the alpha particle upset of a six-transistor static random access memory (SRAM) cell. Then a methodology is described for the analysis of a standard-cell inverter latch.

The characterization of the single-event-upset (SEU) resistance of a latch requires four parameters: (a) the sensitive-diode critical upset charge, Q_c , (b) the sensitive-diode area, A , (c) the particle collection depth, D_c , and (d) the particle Linear Energy Transfer, LET. These parameters come from SPICE simulation, device layout, experimental results from particle testing, and atomic physics, respectively. The upset rate for a latch can be calculated from the Petersen Equation [3] which was developed for the 10 percent worst case Cosmic Ray environment at geosynchronous orbit:

$$R \left(\frac{\text{upsets}}{\text{bit} \cdot \text{day}} \right) = 5 \times 10^{-10} \left(\frac{A(\mu\text{m}^2)}{[Q_c(\text{pC})/D_c(\mu\text{m})]^2} \right) \quad (1)$$

In the above equation, the most difficult parameter to obtain is the particle collection depth, D_c . In the methodology presented here, the collection depth is determined from heavy ion testing of a specially designed test SRAM [4]. The SRAMs are designed to be sensitive to low LET particles such as alpha particles. This allows the immediate, low cost

¹Sponsored by the National Aeronautics and Space Administration and the Defense Advanced Research

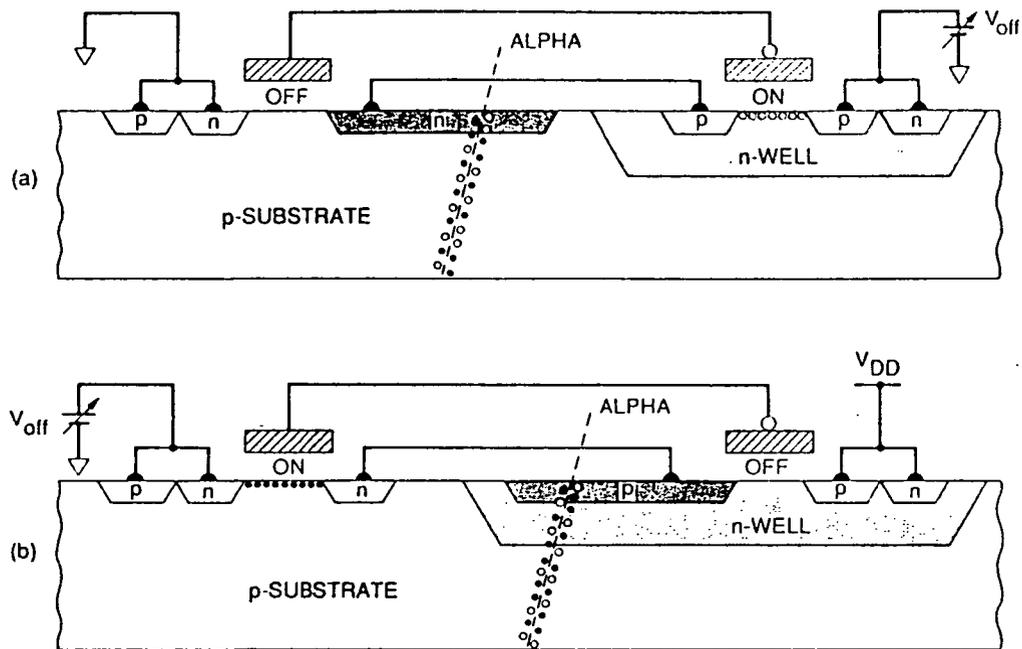


Figure 1: Electron-hole pair tracks from an alpha particle strike through (a) an n+ in p-substrate junction, and through (b) a p+ in n-well junction where truncation of the collected charge occurs

characterization of SEU susceptibility and replaces the time delayed and costly cyclotron testing.

The upset of CMOS circuits depends on the presence of reverse biased junctions. As seen in Figure 1a, n-junctions formed in the p-substrate have a large collection depth. As seen in Figure 1b, the collection depth for p-junctions formed in the n-well is truncated by the well-substrate junction. Such junctions can collect much less charge and are much more difficult to upset. In this study, we will consider the upset of only reverse-biased n-junctions formed in the p-substrate.

2 SRAM Design

For the test SRAM shown in Figure 2, the alpha sensitivity is achieved by imbalancing the cell using an offset voltage, V_{P2} , and by bloating the drain area of the pull-down MOSFET, MN2. The offset voltage, V_{P2} , is placed on the source of the pull-up MOSFET connected to the bloated drain. The SRAM is biased in a sensitive state as seen in Figure 2 where MP2 is ON and MN2 is OFF. As seen in Figure 3, the offset voltage controls the critical charge needed to upset the cell. These curves were obtained from a SPICE analysis of the circuit shown in Figure 2 using a triangle wave with a 200 ps pulse width. This pulse width is typical of an alpha particle strike in silicon and is much less than the response time of the circuit [4].

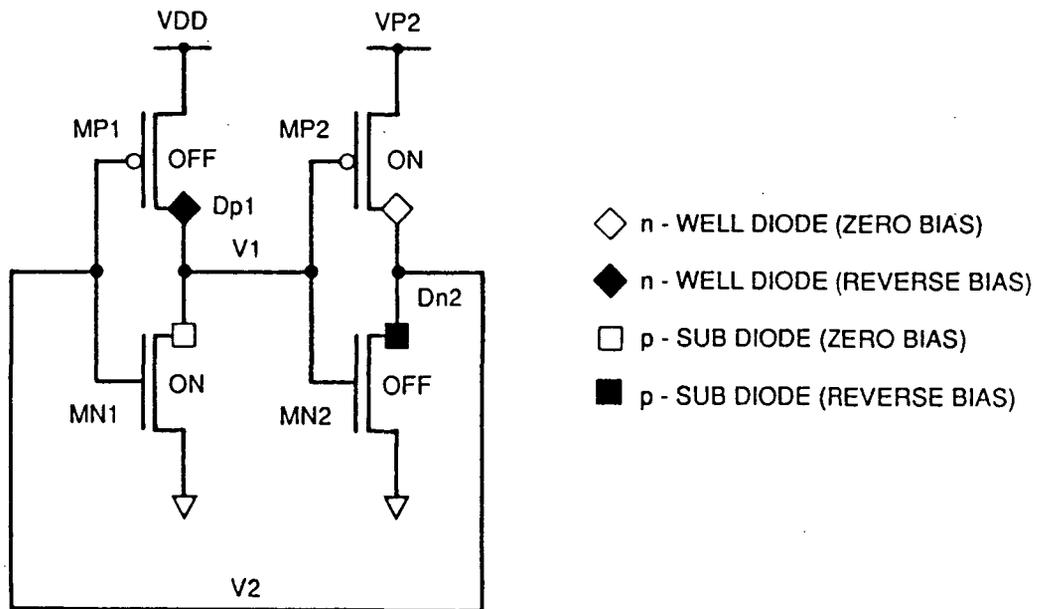


Figure 2: SRAM cell in the zero state with SEU-sensitive drain diodes DP1 and DN2

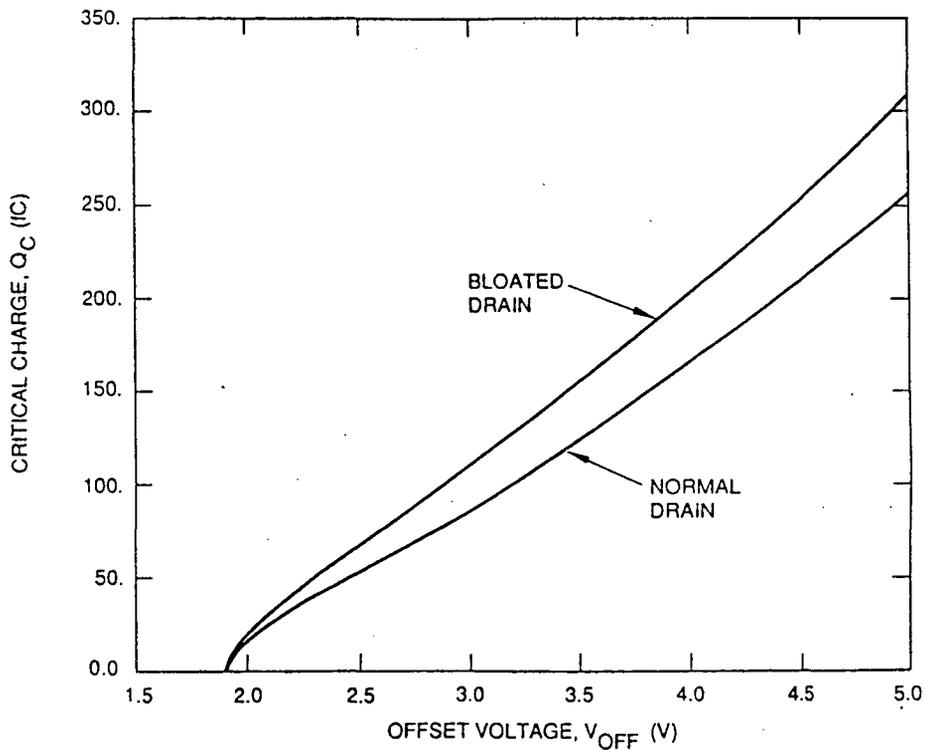


Figure 3: Critical charge characteristics of the SRAM cell using SPICE

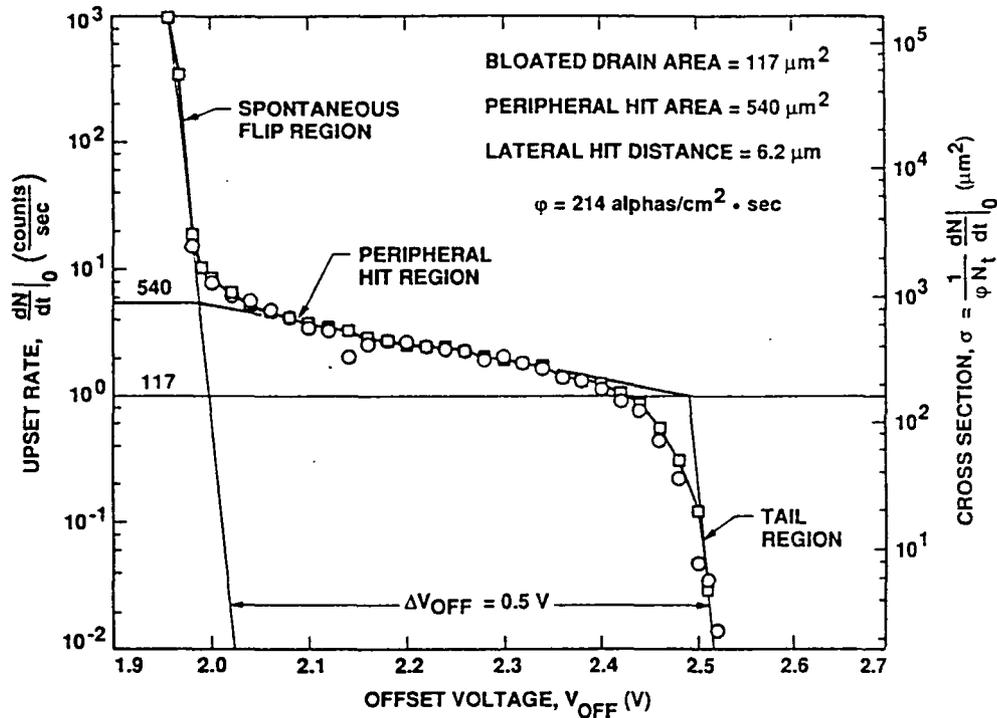


Figure 4: SRAM upset rate characteristics induced by Po-208 alpha particles 32.5 mm above the chip in a vacuum. The cross section observed by extrapolating the peripheral hit and tail regions agrees with the designed cross section of $117 \mu\text{m}^2$

As seen in Figure 4, the test SRAM offset voltage, V_{P2} , varied from 5.0 V to 1.9 V before the cells spontaneously flipped to the other state. Measurements on 2- μm CMOS n-well 4k SRAMs, indicated that a Po-208 5.1-MeV alpha particle source was able to flip the memory cells for offset voltages between 1.9 and 2.5 V. As seen in Figure 4, the offset voltage shift, ΔV_{OFF} was 0.5 V which corresponds to the critical charge induced by the alpha particle in flipping the cell. For ΔV_{OFF} of 0.5 V, the critical charge was determined from Figure 3 to be 58 fC. Finally the collection depth was determined from the charge deposition profile as shown in Figure 5 to be 8 μm .

For a bloated drain area of $ADN2 = 117 \mu\text{m}^2$, $Q_c = 58$ fC, and $D_c = 8 \mu\text{m}$, the calculated upset rate is 1.1×10^{-3} upsets/bit-day and the LET is $0.71 \text{ MeV-cm}^2/\text{mg}$. The upset rate can now be calculated for a "normal" cell with a minimum drain area of $ADN2 = 28 \mu\text{m}^2$ and no offset voltage (ie. 5.0 V). Under these conditions the critical charge is 255 fC as seen in Figure 3. Assuming a collection depth of 8 μm as determined above, the calculated upset rate is 1.38×10^{-5} upsets/bit-day and the LET is $3.08 \text{ MeV-cm}^2/\text{mg}$.

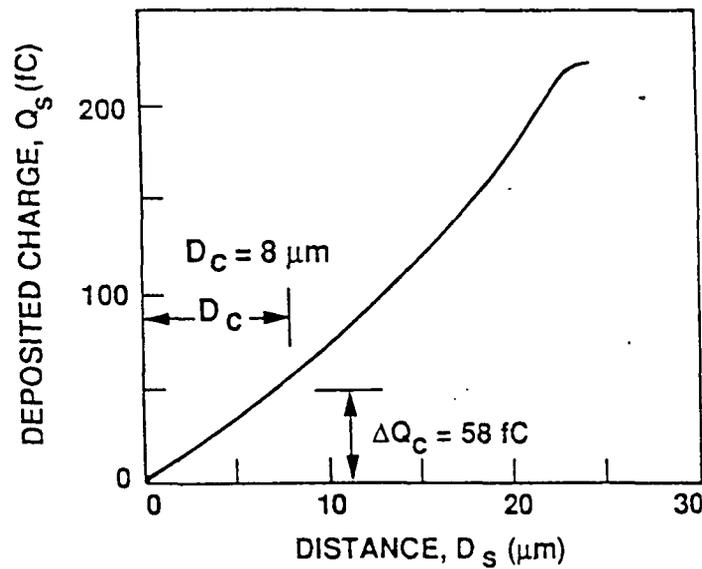


Figure 5: Deposited charge profile of a 5-Me V alpha particle in silicon

3 Standard-Cell Inverter Latch

The above scenario represents the proposed methodology for calculating the SEU resistance of latches used in ASIC designs. This methodology calls for test latches to be designed with selected nodes disconnected from the power lines and connected to offset voltage lines. Currently the latches used in JPL's standard cell library are being designed into a test latch array so their upsetability can be evaluated using alpha particles.

A schematic diagram of a test D-latch obtained from JPL's standard cell library is shown in Figure 6. This latch is shown with the D-input disabled. R is a polysilicon interconnect resistance and is about 200 ohms in the unhardened version of the latch considered here. C1 and C2 are fixed interconnect capacitances. To sensitize the latch to alpha particle induced upsets, a voltage source VP2 is placed on the source of MP2 as shown in Figure 6. The nodes that are most sensitive to upset are nodes 2 and 3 due to their reverse biased p-substrate diodes. The SPICE generated critical charge characteristics of nodes 2 and 3 are shown in Figure 7. The slope of the curves in this figure as well as those of Figure 3 have dimensional units of capacitance. This upset capacitance is a function of the physical capacitance on the node and the "on" MOSFET (restoring path) connected to the node. The upset capacitance increases when the physical capacitance is increased and decreases as the width/length ratio of the restoring MOSFET is increased. Node 3 is the most sensitive node to upset, having the smallest critical charge, primarily because the restoring path is weaker (smaller effective width/length) than that of node 2 due to the series connection of MP2 and MP4.

Using the SRAM result for the collection depth, $D_c = 8 \mu\text{m}$, the minimum LET of a particle that upsets node 2 with no offset voltage (ie. 5.0 V) where $Q_c = 2400 \text{ fC}$ is $29.0 \text{ MeV}\cdot\text{cm}^2/\text{mg}$. Likewise the minimum LET of a particle that upsets node 3 where

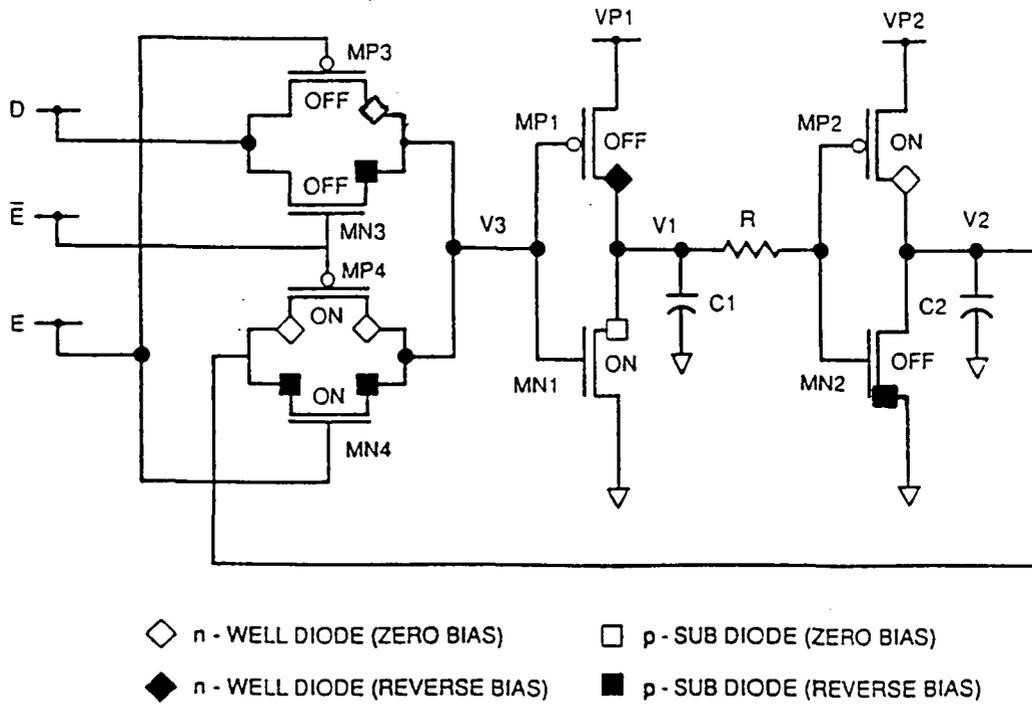


Figure 6: D-latch in the zero state showing reverse biased SEU-sensitive drain diodes

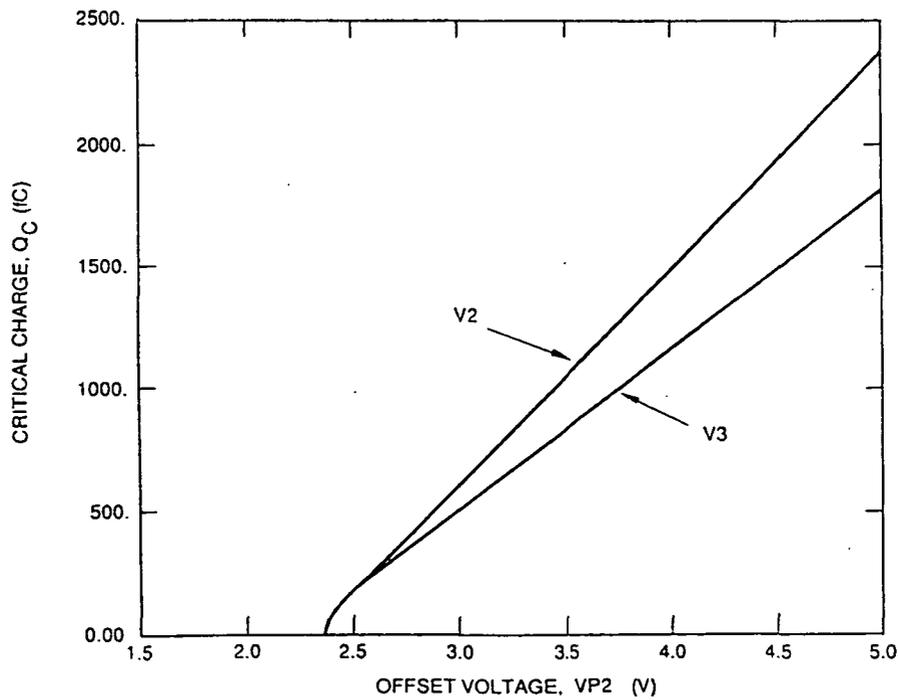


Figure 7: Critical charge characteristics of the D-latch using SPICE

$Q_c = 1800$ fC is 21.7 MeV-cm²/mg. Again, assuming a collection depth of 8 μm and using the node 2 sensitive area of 100 μm^2 and the node 3 sensitive area of 64 μm^2 , the calculated upset rate using the Petersen Equation is 1.27×10^{-6} upsets/bit-day.

4 Conclusion

An approach has been presented for verifying the susceptibility of CMOS latches to heavy-ion induced state changes that uses alpha particles to induce upsets in test circuits using inexpensive bench-level equipment. In this method, the experimental data is linked to alpha particle interaction physics and to SPICE circuit simulations through the alpha particle collection depth. JPL's standard cell latch array, currently in fabrication, will be used to validate this methodology by comparing the results obtained with alpha particles to that obtained with high LET heavy ions at a cyclotron.

References

- [1] H. Nakano and H. Schmeichel, "A Protective Method Against RAM Upsets Due to Cosmic Rays," Annual Rocky Mountain Guidance and Control Conference, (February, 1984).
- [2] G. M. Burdick, E. H. Kopf, and D. D. Meyer, "The Galileo Single-Event Upset Solution and Risk Assessment," Annual Rocky Mountain Guidance and Control Conference, (February 1986).
- [3] E. L. Petersen, J. B. Langworthy, and S. E. Diehl, "Suggested Single Event Upset Figure of Merit," IEEE Trans. on Nuclear Science, NS-30, 4533-4539 (1983).
- [4] M. G. Buehler and B. R. Blaes, "Alpha-Particle Sensitive Test SRAMs," submitted for publication in the IEEE Trans. on Nuclear Science.

Costs of Becoming a Commodity

What does low cost electronics imply ?

R. Foote¹ Hewlett Packard Disk Mechanisms
Boise, Idaho

Abstract - The disk drive business is rapidly becoming a commodity marketplace at higher and higher disk capacities. In order to remain in the business your development tactics must change to address the cost requirements and constraints of a commodity marketplace. This is a look at the electronics portion of the cost equation and what it implies on Integrated Circuit technology and potential types of companies and processes that can meet the challenges of this market.

The Disk drive industry in the last few years has undergone some dramatic changes. The size of a 10 megabyte disk drive has decreased from washing machine size to shirt pocket size and the price has dropped from tens of thousands of dollars to only a few hundred dollars.

This change has brought a whole new set of pressures on the disk drive industry. In the next few years, the usage of disk drives will move more and more into consumer products. Disk drives costs must become competitive with consumer or commodity products. The computer industry demand for more and more storage on systems and higher and higher performance places the mid range to high end disk drive developers in positions they are unfamiliar with.

To obtain an idea of the magnitude of the problem we first need to pick a capacity point and investigate what requirements this places on the components within a disk drive. Choosing 100 megabytes and the 3 1/2" industry standard form factor and a delivery of the drive to the marketplace in the late 1993 to early 1994 time frame we can make some assumptions about the price and the manufacturing cost of the disk drive itself.

In this time frame disks of this size will be selling for about \$2.00 per megabyte this implies around a \$200.00 price tag for the disk drive itself. In order to produce this product profitably we will assume that a 26% gross margin is necessary. This yields a factory cost of roughly \$159. On the average in this size of disk drive the electronics is about 32% of the total drive cost. This leaves about \$50.00 for electronics cost in a 1994 100 megabyte disk drive. This 100 megabyte disk drive will be required to have performance and functionality characteristics as good as or better than the 760 megabyte to 1 gigabyte 5 1/4" drives of today.

There exists at least two paths to achieve this cost and performance point for the electronics. One being to take the low cost 20 megabyte electronics we have today and

¹Roy Foote is a Research and Development Section Manager for Hewlett-Packard's Disk Mechanisms Division in Boise Idaho. He received a Masters Degree in Electrical Engineering from the University of Idaho in 1975 and has been with HP for the last 15 years. He has worked as a designer of printers, tapedrives, and disk drives. His major focus in the last few years has been in disk drive controllers. These controllers have contained significant custom IC work and contain some cells designed by the MRC.

somehow add features and speed while reducing costs. The other being to take the higher performance electronics of today and integrate and cost reduce by volume production in order to meet the \$50.00 magic number. The next step is to look at what individual functions are in this black box called disk drive electronics.

The following block diagram shows the functional blocks in a disk drive and the relative performance criteria for this class of disk drive.

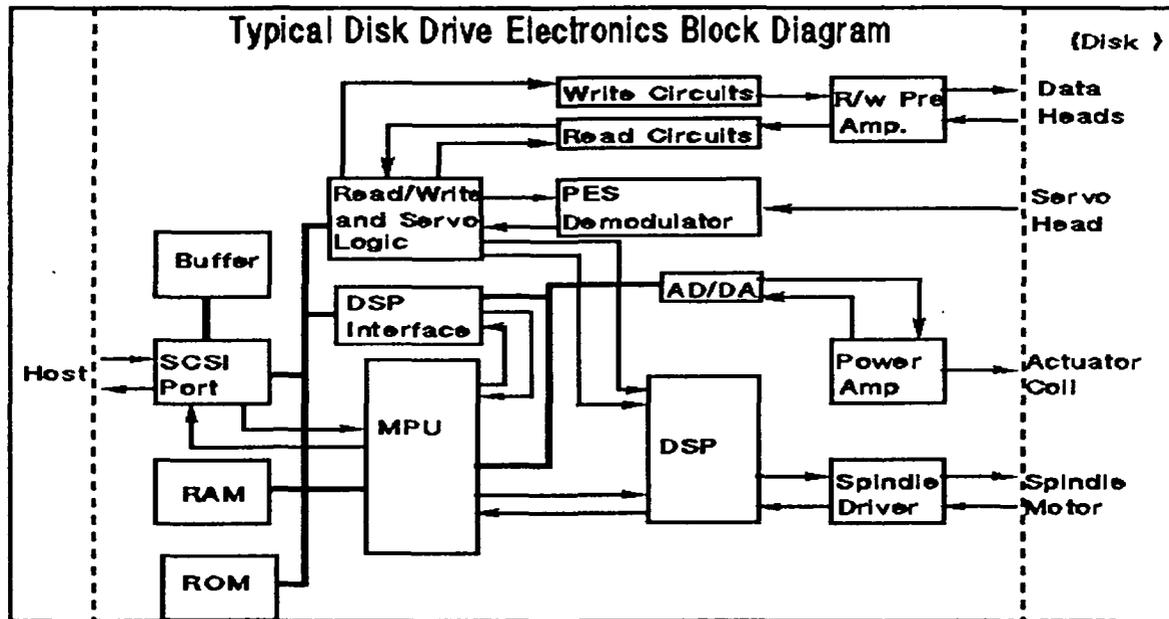


Figure 1: SCSI port 5 to 10 megabytes/second Disk Data Rate 24 to 40 megabits/second Two Track Buffer

In the previous diagram the functions on the left side of the drawing are digital in nature. The functions on the right are mainly analog. The electronic parts included in the \$50.00 cost are those from the left side of the drawing moving to the right up to but not including the data heads, servo head, actuator coil, and spindle motor. These parts are covered by the budget for the mechanism portion of the disk drive.

Disk drive electronics has traditionally been partitioned by a low level interface which is serial in nature. A typical interface of this type is ESDI, (Enhanced Small Device Interface). Today there is also a higher level interface which bridges the ESDI interface to the host (in this example SCSI (Small Computer Systems Interface)).

The ESDI portion of the disk electronics includes the servo system, the read and write preamp, and the compensation systems. This portion of the controller contains its own microprocessor and firmware to which performs the ESDI interface function. The rest of the electronics, the SCSI chip, microprocessor, RAM, ROM buffer and some disk data formatting circuitry make up the ESDI to SCSI bridge portion of the electronics.

In a low end disk drive the ESDI and SCSI microprocessor functions are executed by a single microprocessor. In order to meet our cost goals we will assume a single micropro-

cessor system. In combining the functions into one microprocessor we tend to remove the barrier of ESDI and a distinct dividing line between where ESDI ends and SCSI bridge begins no longer exists.

This combination also allows us the freedom to combine more of the digital functions onto larger integrated circuits and to consider combining analog functions with digital functions to reduce part and pin count. These combinations all tend to reduce total parts cost and the space required to place the parts onto printed circuit boards.

The next step is to determine what the chip boundaries could be, in order to evaluate the complexity and size of each of the new integrated circuits.

The following diagram shows one of the choices for combining functions. This choice is rather arbitrary. Dashed lines surrounding the functions indicate the proposed chip boundaries.

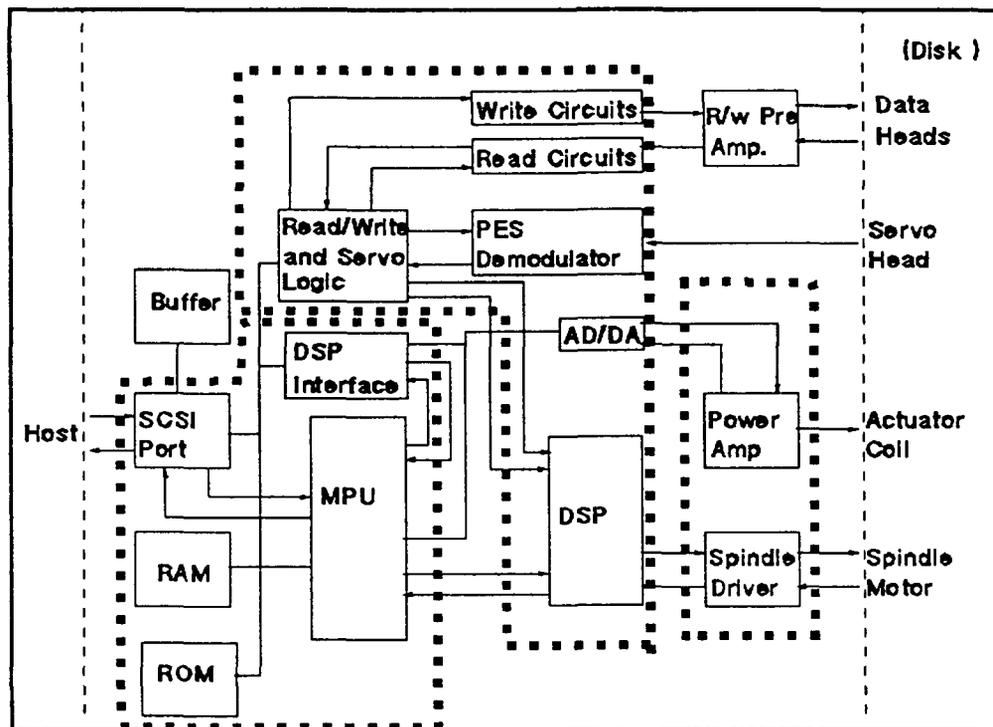


Figure 2: Typical disk drive electronics for block diagram

The first combination is that of ROM, RAM, SCSI, MPU, and the DSP interface.

The second combination is read write and servo logic, read and write circuits, PES (Position Error Signal) demodulator, A to D, D to A, and DSP (digital signal processor).

The third and final combination is that of the power amplifier for the actuator and the spindle driver.

The buffer and the read write preamp have been left as individual parts.

After choosing a partitioning of the electronics, maximum cost estimates for each combination can be made. These estimates should be based on the relative complexity of the parts and their approximate die size and package count.

Functional Cost Budget

<i>Function</i>	<i>Cost</i>	<i>Board Space</i>
Buffer	\$4.00	.5in ²
Preamp	\$3.00	.5in ²
Mpu-etc.	\$10.00	1.2in ²
DPS-etc.	\$10.00	1.2in ²
Power	\$5.00	1.0in ²
OVHD	\$18.00	1.2in ²
Totals	\$50.00	5.6in²

Relative complexity is the next issue in estimating what this level of integration requires of the designer and of the fabrication facility.

The buffer: It is apparent that this function will be an off the shelf part which must be in high volume in order to meet the cost goals. The Buffer will be purchased from one of the leaders in DRAM production.

The read write preamps: This function is drive dependent and may be designed for each successive drive to better match the head and read write systems.

The power section: This is also drive dependent and may require some customization of circuits for each drive it is used on.

The DSP- etc.: This part involves more general algorithms and can be designed for use in more than one drive. This part is a mixed signal part and requires processes like analog CMOS or BI-CMOS. The analog portions are in the signal conditioning circuits portion of the read and write Circuits and have a bandwidth requirement of up to 40 megabits per second. The PES demodulator circuits and the A to D, D to A circuits are mixed signal but the bandwidth requirements are lower. The DSP itself and the read write and servo logic are digital in nature and can be readily implemented in common CMOS geometries.

The MPU-etc.: This part involves a design challenge of placing as much digital logic as possible on chip. The MPU, it's ROM, RAM, and portions of the SCSI function require a process with good speed and excellent density properties. The SCSI portion may require 40 milliampere drivers for the SCSI bus.

Some basic assumptions can be made about each combination and its associated marketplace. These assumptions will guide the evaluation of which vendors are capable of producing parts with the required cost structures and design expertise.

Starting with the buffer: This part will most likely be an industry standard DRAM part. These parts are used in other applications besides disk drives. The volume on this part will be driven by all it's applications and therefore will not require much disk volume

DRAM TRENDS

year	1990	1994	1998
size	4mbit	64mbit	256mbit
geometry	.8micron	.3micron	.2micron
speed	60nsec	50nsec	40nsec

influence to meet our cost goals for the part.

The MPU-etc.: This part will require high density and low cost processes. In choosing a vendor we need to take a look at the volume manufacturers of RAMs, ROMs, and MPUs.

These manufacturers can be placed loosely into two categories. Those with a state of the art DRAM process and an active involvement in the DRAM market and those without a DRAM involvement.

Looking at the DRAM manufacturers yields some interesting information. Most of the leaders are pushing the limits of each process as it becomes available. A current estimate of the evolution of DRAM processes is shown in the following chart.

DRAM manufacturers are pushing technology as hard as they can while concentrating on low cost production. Most DRAM vendors are filling their fabrication facilities with DRAM parts in order to recover the high costs of a building a fab.

About one to two years after they begin production on a new fab, migration of existing and new designs like ASIC's and custom parts into the fab occurs. This allows significant cost reduction and gives the manufacturer the ability to make a profit on a production line which paid off it's cost's while producing DRAMS.

The analysis of what cost structures and densities are necessary makes manufacturers with significant DRAM involvement very attractive for the MPU-etc. part.

The other major group is those IC manufacturers with similar design and density capabilities but without a significant DRAM involvement. This business is usually driven by a unique product or process which allows them to have significant market share in a particular area.

One example of this is the Microprocessor business. Microprocessor based businesses are driven by the next speed level required by the computer industry. Examples of this type of company are INTEL and Motorola and their successive processor families like the 68000, 68020, 68030 ... and the 86, 286, 386, 486

Looking at the geometries proposed by these companies yields similar geometric points as those in the DRAM business.

Next we need to investigate the influence of volume on these parts and what the results are in choosing a vendor.

Volume produces different results on pricing is influenced by the vendor of choice. A particular vendor may quote pricing based on an agreed upon annual volume, monthly volume, or even lifetime volume. This tends to cloud the issue of which vendor has the

MICROPROCESSOR TRENDS

year	1990	1991	1994
# of Transistors	1.2 million	2.0 million	4.0 million
geometry	.8micron	.65micron	.5micron

best price for a particular custom IC. In order to obtain the best price a common metric for quotation is required. For this example will use monthly volumes as the price point of comparison.

An average vendors prices appear to be significantly driven by volume with volumes less than 100,000 pieces per month. On an example chip with a complexity of 34,000 transistors, 80% of the volume price reduction was achieved by increasing volume from 5,000 to 100,000 pieces per month. A 20% reduction was achieved in going from 100,000 to 500,000 pieces per month. A curve showing the cost versus volume trends for three different chips of increasing complexity is shown below. The chips have the complexity of 34,000, 50,000, and 80,000 transistors.

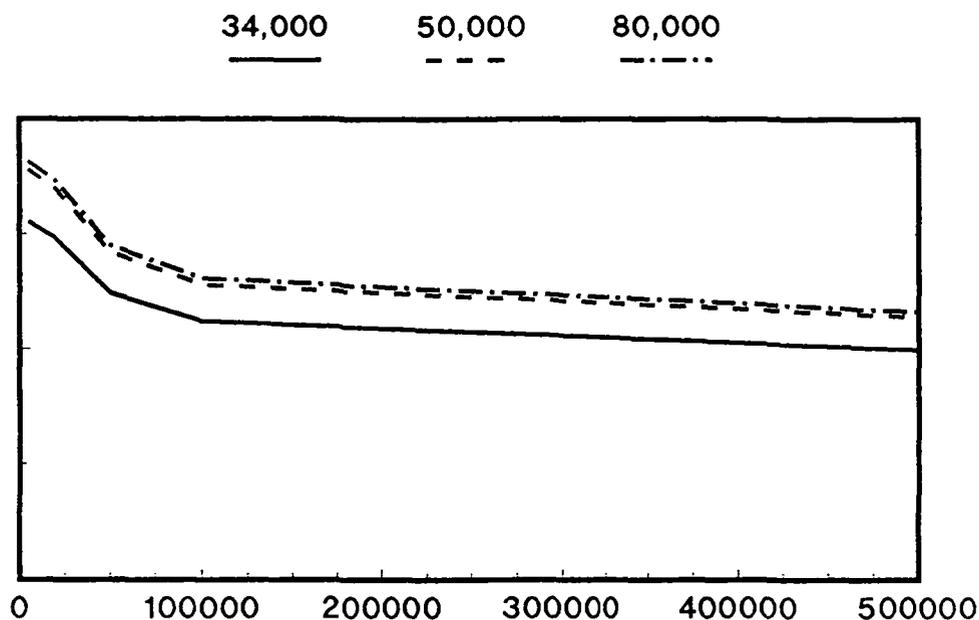


Figure 3: Typical cost versus volume curve for 1991

If volume is 100,000 units per month the cost structure for each fabrication facility should be about the same if they all are running their facilities near capacity. If the volumes are less than 100,000 units per month the price of fabricated parts will be effected by the overhead of the facility and the how close they are to production capacity.

For our example will assume a volume of 100,000 pieces per month, and limit our

choices to those fabrication facilities with adequate production and capabilities to meet our needs.

The next question to resolve is time to market. Time to market in a commodity business is the key to obtaining market share. If you arrive late with the next product you can approach fewer customers with your product and your volume will be lower. Being late forces your costs up and greatly reduces margin.

What gives a particular fabrication facility the ability to meet Time to market requirements?

A supplier must have all the business basics under control. The process must handle the densities and complexities of the designs to be implemented. Prototype and production cycles must meet the needs of a typical product development cycle.

Perhaps the most important area in achieving time to market is turning designs into layouts. A foundry needs a good tool set for design and layout and you need experienced engineers using the tools. Deciding who will do this portion of the design, your engineers, the foundries engineers, or a combination of both is also required.

For this example will assume the foundry will supply the engineering talent to take higher level designs and reduce them to silicon. This decision tends to narrow the choices of vendors. These vendors must meet all the previous qualifications and have experience in this type of electronics. A vendor with a unique approach which improves time to market will have a better chance of meeting needs.

An example of a special approach which improves time to market is that of using large functional cell blocks (Megacells) and combining them together to build the desired chip functions. The cells we need for this particular chip are MPU, ROM, RAM, DSP interface, and SCSI interface cells. Focusing the choice for the MPU etc. chip to those vendors with the ability to combine Megacells together reduces the field of viable vendors to a small number.

One other large consideration in a project of this type is NRE (Non recoverable Engineering) expense. There are three main ways to handle NRE. The first being a direct charge for all development work, engineering work, masks etc.. This requires payment of these charges during development or when first parts are received. On a typical custom design of 30,000 to 40,000 transistors NRE can cost from \$100,000 to \$1,000,000. The charge depends on how much of the engineering was done by your engineers.

The second way to handle NRE is to amortize it over the life of the product. This increases part cost over the life of the part and requires a minimum number of parts be purchased or a penalty fee is charged if you don't.

The third way is for the supplier to pay the NRE. Developing this kind of relationship requires more work and requires an exchange of engineering expertise or insight for the NRE. Training a particular integrated circuit vendor in the nuances of disk drive electronics is an example of this type of a relationship. Allowing the vendor to produce parts which can also be sold to anyone in the disk drive industry further cements this relationship.

As happens in most commodity industries being radically different is not better and standard may be better than better. One of the biggest keys to success in a commodity business is time to market. If involvement is maintained in the design and specification of

an industry standard part, design and implementation of the part in a product can occur a few months ahead of the rest of the industry. This three to six month advantage can be all that is necessary to keep and grow market share.

The DSP-etc. part brings up some unique requirements of a vendor which the MPU-etc. part did not. Having the same vendor for these two parts is always desirable but may not be practical. The DSP-etc. part contains a large portion of the analog design required in a disk drive.

With bit densities on disk drives continuously increasing at about 30% per year the requirements on the Read and Write circuits are always changing from product to product. A vendor must have a solid analog process capable of expanding and growing with the requirements in the disk drive industry and engineering tools and talent good enough to allow changes in data rate of between 30% to 50% each year. Analog IC designs require models which are very well matched to a particular process and geometry. A library of components which are used in disk drive applications is also important. In choosing the vendor for this part, experience with or a commitment to the disk drive marketplace is also a strong desire.

Adding up the requirements reduces the list of vendors to those which are currently supplying parts to the disk drive business today. Carefully evaluating each one of these players for long term survival and technical strengths yields a small group to begin in depth evaluations with.

The last combinational chip to explore is the Power chip. This part like the DSP-etc. part requires a significant amount of analog expertise. Although the type of expertise is not always available in pure analog design houses, as power is significantly different to deal with than low noise analog. Using the previously mentioned analog criteria and the additional requirements of a power device reduces the field to a few players.

Comparing the lists from each chip now yields two or three companies with the broad spectrum of capabilities required to do all the chips required and a desire to sell in this marketplace.

This exercise has been one of enlightenment, in two main areas. These areas are the capability of the industry as a whole and the vendors in particular. Having written down some assumptions at the beginning it has been useful to review those assumptions again. Some of the companies which would have been considered as most likely to be on the final "short list" were dropped out of the competition very early. Others who weren't even on the original list have been moved on to the list for consideration.

One of the original assumptions on volume was that parts needed to be industry standard and used by a large portion of the industry in order for our cost goals to be met. The information on volume showed that a customer with a volume of 100,000 parts per month or greater will get most of his cost reduction achieving that volume. This allows us to make the conclusion that unique parts are probably cost effective if reasonable volumes can be achieved.

Each vendor has optimized one or two areas of his business and therefore has a skill set which can either fit very well into our environment or not match well at all. Some companies have tremendous manufacturing expertise and quality but have a design capability which

is two to five years behind the market leaders in design. Some vendors have tremendous design tools and no real world class fabrication capability.

In general the disk electronics business is becoming a strategic target market for an ever increasing number of integrated circuit manufacturers. Some companies have targeted a particular portion of disk drive electronics. There are specialist companies with expertise in interface (SCSI and IBM-PC-AT) and disk formatter IC's and other companies with analog specific parts for the read write area and the servo area. The study showed no one company with all of the bases covered for disk drive electronics. It did however show a few with the tools and capabilities to handle all of the requirements of disk drive electronics. Working closely with one of the few it appears possible to achieve the goals of a \$50.00 dollar electronics cost for all the electronic parts of a disk drive within the time frame required by the industry.

Session 2
Reliable Circuit Design

Chairman: Jon Gibson

Reliable VLSI Sequential Controllers

S. Whitaker, G. Maki and M. Shamanna
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - A VLSI architecture for synchronous sequential controllers is presented that has attractive qualities for producing reliable circuits. In these circuits, one hardware implementation can realize any flow table with a maximum of 2^n internal states and m inputs. Also all design equations are identical. A real time fault detection means is presented along with a strategy for verifying the correctness of the checking hardware. This self check feature can be employed with no increase in hardware. The architecture can be modified to achieve fail safe designs. With no increase in hardware, an adaptable circuit can be realized that allows replacement of faulty transitions with fault free transitions.

1 Introduction

This paper presents a VLSI architecture for controllers that provides real time fault detection and reliability enhancements. The reliability advances are based on a new VLSI architecture for synchronous sequential circuits. This controller design supports important features such as real time fault detection, fail safeness and fault tolerance. The class of faults that is covered by this design are stuck-at, stuck-open and stuck-on. Controllers on two full custom VLSI data compression chips for NASA have been implemented using this architecture [1].

Most digital systems include a controller. This can be either a general machine such as a microprocessor, or a dedicated, custom designed sequential state machine. Dedicated controllers can be implemented as programmable PLA based structures, or as a random logic designs. The realization of state machines based on random logic often results in the most compact and highest performance circuits, but the logic is a function of the state assignment, flip flop type and flow table (actual sequence). Controllers can also be implemented in PLA structures, reducing the layout effort, but are less area efficient and have reduced system performance. PLA based controllers can be reconfigured to some extent but the reconfigurability is limited by the number of minterms available in the PLA.

An architecture that retains the traditional strengths of dedicated state machines, but offers the programmability of a microcontroller was presented in [2]. This architecture produces controllers whose logic is invariant with respect to the actual sequence desired. State machines designed using this method approach the performance and size of random logic based state machines and have a programmability superior to a PLA based design.

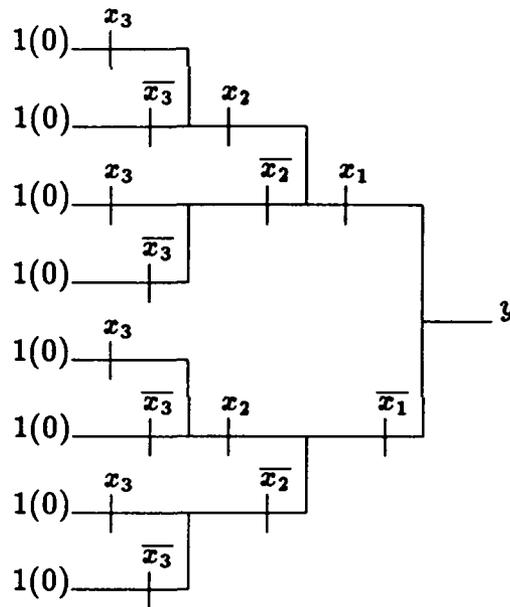


Figure 1: General three-variable BTS network.

2 Binary Tree Structured Logic

Pass transistor logic can have significant advantages in speed and density when compared with gate logic [3,4]. A pass transistor network realized in Binary Tree Structure (BTS) form often requires fewer transistors and displays attractive fault detection characteristics [3]. In general, a BTS circuit is characterized by having a maximum of $2^n - 1$ nodes, each node having exactly two branches. One branch is controlled by variable x_i and the other by \bar{x}_i . The maximum number of transistors in a BTS network is $2^{n+1} - 2$. A general BTS network contains the maximum number of transistors and represents a complete decoding of an input space and hence only constants are input to the network. A general BTS network is employed here to formulate the next state equations for sequential circuits. Figure 1 shows a general BTS network which implements all three-variable functions, any of which can be realized by simply changing the pass variable constants, $1(0)$, at the input to the appropriate branch.

The delay through a series string of pass transistors is proportional to the square of the number pass transistors. This limits the size of a sequential circuit using BTS structures to about 5 state variables. However, this is not a significant limitation. Large state controllers can almost always be partitioned into small distributed state machines. Two full custom developments [5,6] illustrate this partitioning. About 100 bits of state control excluding counters and pipe delays were required on the 200,000 transistor Reed Solomon decoder for the NASA Hubble Space Telescope [6]. The state control was partitioned such that no state machine required more than 5 state variables. About 225 bits of state control excluding counters and pipe delays were required on the 210,000 transistor Auto Centroid

	I_1	I_2	I_3
A	C	B	A
B	D	C	B
C	E	D	C
D	F	E	D
E	A	F	E
F	B	A	F

Table 1: Example flow table.

Calculator chip designed for Lawrence Livermore Laboratory [5]. Again the state control was partitioned such that only one individual state machine required more than 5 state variables.

3 State Machine Design

3.1 Architecture

The logic that forms each next state equation, Y_i , consists of the following elements: a storage device (normally a flip-flop), next state excitation circuitry which generates the next state values to the flip-flop, and input logic. Present state information is fed back by state variables y_i to the excitation logic. The excitation logic is a combinational logic function of the input and state variable information. In general, the information needed to generate all possible next state values for the circuit is resident within the excitation logic. The current input and state variables select the specific next state value.

In order for the circuit to implement arbitrary state transitions, the next state circuitry must assume a unique form. First, the hardware for each next state variable must be identical. Second, specific next state information must not be hardwired into the logic that forms the next state equations. Rather, specific next state values must be presented from an external source. The architecture presented here yields a circuit that can realize any flow table up to a maximum of m input states and 2^n internal states without a change in hardware.

Conceptually, the new architecture operates as follows: For each predecessor state of S_i , there exists a pass transistor path in the excitation network that presents the next state value, S_i , to the flip-flops. Predecessor states for state S_i under an input I_p are all states which have S_i as a next state entry. In Table 1, the predecessor state for C under I_1 is A. Whenever the circuit is in a predecessor state of S_i , the next clock pulse will effect a transition to S_i . The pass transistor network consists of a single pass implicant that covers each predecessor state such that when any state is entered, a unique pass transistor path is enabled that passes the proper next state value to the flip-flops.

All equations are identical when they are realized with general BTS networks that completely decode all the internal states. That is, if there are n state variables, then the BTS network must decode all 2^n states. The value for the next state entries for each

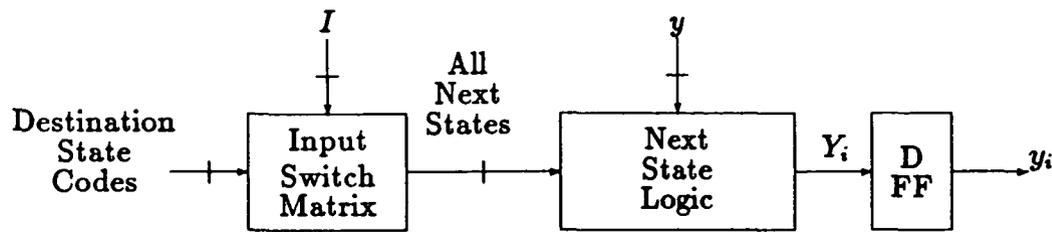


Figure 2: General block diagram.

predecessor state for S_i is the code for S_i and the constants for this code are input to the BTS network. A general block diagram is shown in Figure 2, where the next state logic is a general BTS network.

3.2 Operation

The following illustrates specifically how this architecture works. Let Table 2 depict an example for a general 3 state variable, 3 input state machine. I_1, I_2 and I_3 are the inputs, $S_0 \dots S_7$ are the present states, and $N_{S_0 I_1}, N_{S_0 I_2} \dots N_{S_7 I_3}$ are the next states. This can be generalized so that $N_{S_i I_j}$ are the next states for S_i under input I_j . $N_{S_i I_j}$ has been abbreviated as N_{ij} . The set of N_{ij} also comprise the destination state codes. Let the state assignment be $S_0 = 000, S_1 = 001, S_2 = 010, \dots, S_7 = 111$.

The next state logic is a general BTS circuit with paths that decode each state. The input switch matrix is a pass transistor matrix, that passes the destination state codes to the next state pass network as shown in Figure 3. The circuit realization of this network operates in the following manner: All of the destination state codes N_{ij} are presented to the input switch matrix. For each input state I_i , all of the destination states in I_i are presented to the next state logic. The present state variables, y , select one and only one next state entry which is passed to the flip-flops. If the machine is in state S_1 and input I_2 is asserted, then N_{12} would be passed to the input of the flip-flop for next state variable Y_i . The current input state selects the set of potential next states that the circuit can assume (selects the input column) and the present state variables select the exact next state (row in the flow table) that the circuit will assume at the next clock pulse.

3.3 Design Example

Consider the state assignment and next state entries shown in Table 3. The circuit in Figure 3 shows the logic for implementing each state variable y_i . Each state is covered by a path through the BTS network that forms the next state logic. The logic of Figure 3 is replicated three times and the inputs are driven by the destination state information which is taken from Table 3. Figure 4 shows the programming of the input switch matrix for next state variable Y_3 . Except for the constant values driving the input switch matrix, the design equations and pass transistor realizations for each state variable are identical.

	I_1	I_2	I_3
S_0	N_{01}	N_{02}	N_{03}
S_1	N_{11}	N_{12}	N_{13}
S_2	N_{21}	N_{22}	N_{23}
S_3	N_{31}	N_{32}	N_{33}
S_4	N_{41}	N_{42}	N_{43}
S_5	N_{51}	N_{52}	N_{53}
S_6	N_{61}	N_{62}	N_{63}
S_7	N_{71}	N_{72}	N_{73}

Table 2: General eight-state three-input flow table.

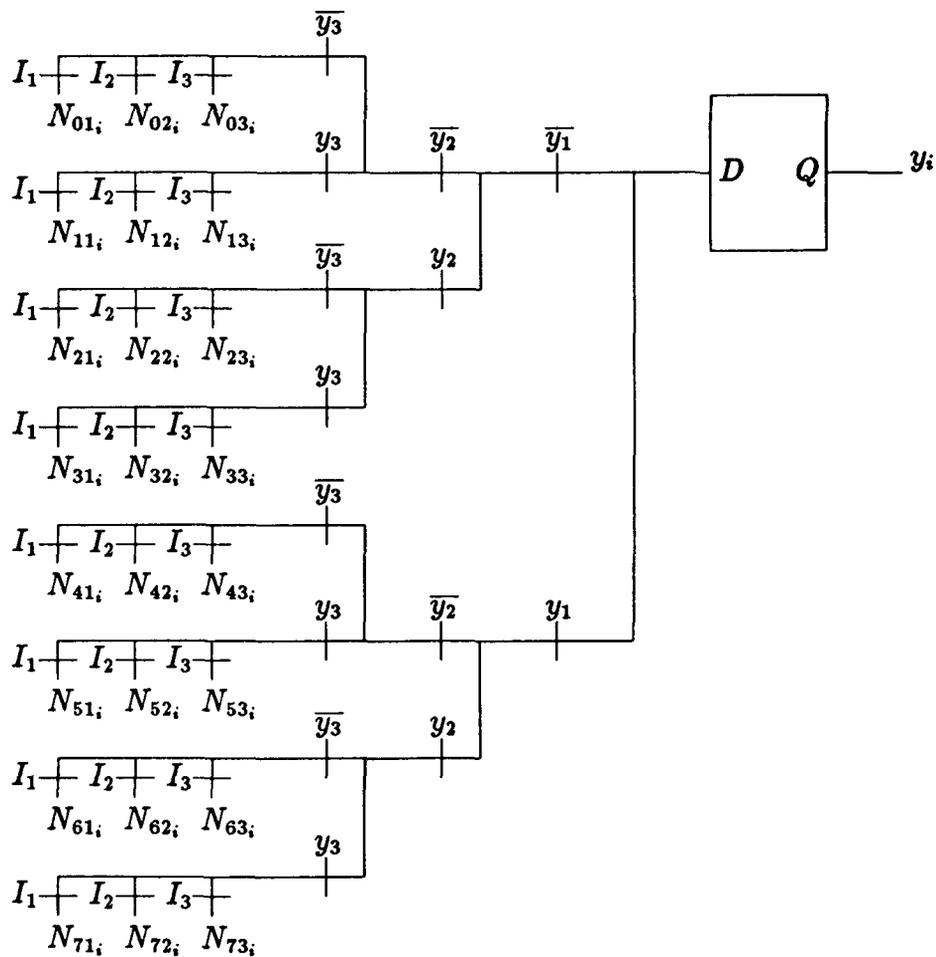


Figure 3: General eight-state three-input next state equation circuit.

2.1.6

y_1	y_2	y_3		I_1	I_2	I_3
0	0	0	A	010	001	000
0	0	1	B	011	010	001
0	1	0	C	100	011	010
0	1	1	D	101	100	011
1	0	0	E	000	101	100
1	0	1	F	001	000	101

Table 3: Example flow table with next state entries.

Since only six of the eight available states are utilized, the paths decoding S_6 and S_7 can have arbitrary next state constants. Here they are set to 0.

It is no longer necessary to derive the pass logic configuration for each next state equation. The next state information is only used as the input pattern to the input switch matrix. Since the next state information is stored in the input switch matrix, only the programming of the destination codes needs be changed to implement a different flow table.

3.4 Safe Operation

In some circuit designs it is possible to enter states that are not specified in the original flow table and it is then impossible to return to an original specified state. If this occurs, the circuit is termed unsafe [7]. The above architecture can be guaranteed to operate safely by simply defining the next state for all unspecified states as any of the originally specified states. Since the BTS network generates the next states for all possible states, there is no increase in the hardware necessary to produce a safe design.

4 Reliable Design

4.1 Real Time Fault Detection

In the following discussion, specified internal states denote those states that are specified in the original flow table and fault states denote those that do not appear in the original flow table. For example, in Table 3 state 001 is a specified state and 110 is a fault state. A circuit never enters a fault state under fault free conditions.

A key feature of the design presented in the previous section is that each state variable utilizes independent logic. Because of this feature, a failure in any component in the BTS network affects at most only one state variable. The occurrence of a fault may force the circuit into a state that is at most a distance one from the intended state.

Meyer has shown that a minimum distance-two state assignment will provide real time fault detection if the fault detector can detect the presence of a fault state within one clock period [8]. With a minimum distance-two state assignment, a single fault forces the circuit into a fault state. When this happens, the fault detector must simply detect the presence of fault states to detect the presence of a fault. For example, suppose the state assignment

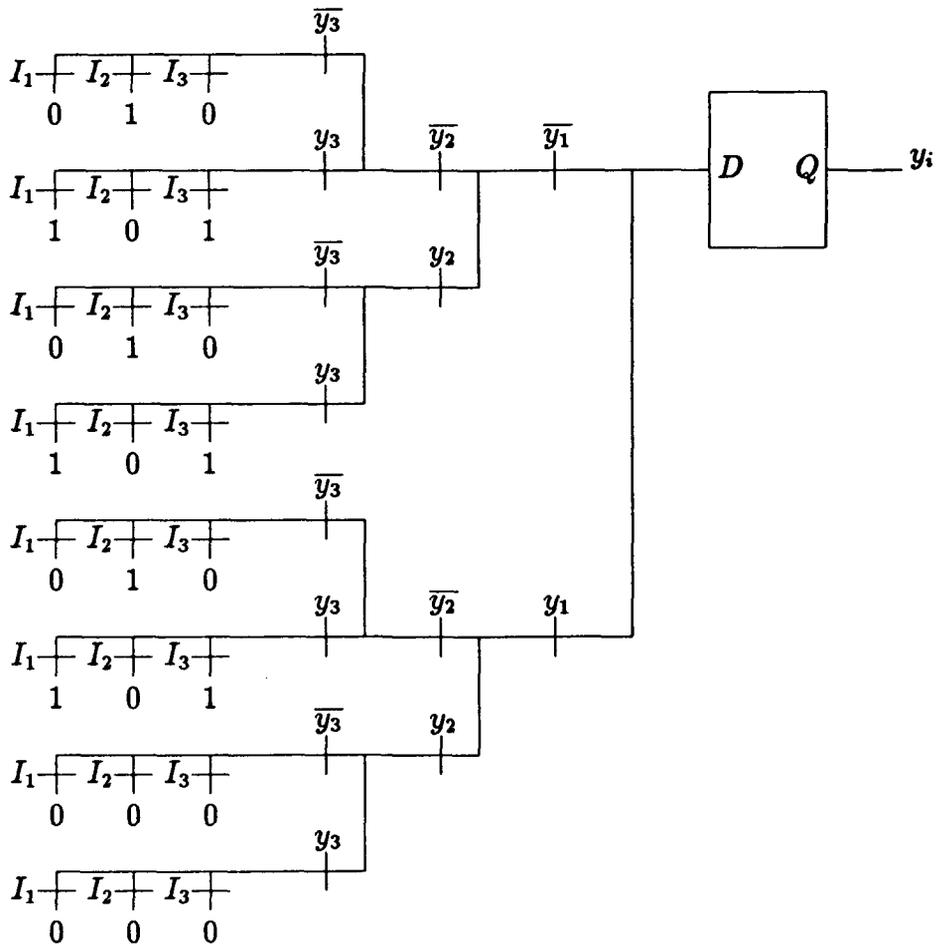


Figure 4: Programming of the input switch matrix for next state equation Y_3 .

encodes all specified states to possess even parity and encodes the fault states to have odd parity. The fault detector would simply detect the presence of odd parity over the state variables. The hardware for this fault detector could be realized by adding another BTS network that is identical to any state variable circuit. However, a smaller more efficient fault detector would be a simple exclusive-or gate. A typical VLSI trade-off has to be made at this point. If design time is important, then replicating the BTS network is best; if minimum area is critical, then implementing the exclusive-or gate is best.

Open-circuit faults often create problems for fault detection because the charge stored on the node can mask failures. For example, if a transistor experiences an open circuit, the input to the D flip-flop will float when the path with the faulted transistor is enabled. If the previous input value to a flip-flop is 1(0) and the next value ought to be 1(0), then the state variable will not change when an open circuit is present. This assumes that the inputs change at a rate faster than the time it takes to discharge the input node to the flip-flop. In the case where there is no change in state, the circuit remains in a proper state and the fault is not detected. The circuit will malfunction whenever the previous input value to a flip-flop is 1(0) and the next value ought to be 0(1). In this situation, the input should transition but the presence of an open circuit does not allow a transition. Whenever the faulted path ought to force a transition in a flip-flop and does not, the presence of an open circuit will result in an unchanged state variable and the circuit entering a fault state which is detectable. In general, the fault detector detects only those faults that cause the circuit to assume a fault state. Open-circuit faults which cause a circuit to remain in a proper state are not detectable.

Only one extra state variable is needed to translate a minimum variable state assignment into a minimum distance-two state assignment. As stated above, a BTS circuit identical to a state variable can be used as the fault detector. Therefore, two additional BTS circuits are required to provide real time fault detection. If n BTS circuits are needed to implement a non-fault detecting circuit, then $n + 2$ BTS circuits are needed for real time fault detection. The depth of the each BTS circuit increases by one upon adding an additional state variable. Since each additional BTS circuit is identical, the extra VLSI layout and checking efforts are small.

Checking the checker is an important issue. To generate confidence in the checking hardware, it is important to have a mechanism that can achieve a self check. Detection of faults in the checking circuit itself is difficult because the states that the fault detector decodes are states that the circuit never enters under fault free conditions. One method to check for detector faults is to force the circuit into a fault state during an off-line test. A complete test would require that the circuit cycle through all fault states. Since the next state entries for the fault states are unspecified, it is a simple matter to specify the next state entries such that the circuit will cycle through the fault states.

Cycling through the fault states in the checking circuit is illustrated with the example shown in Table 4. In this example, let the next state entry of all fault states be specified such that the circuit cycles through all the fault states when a fault state is entered. Shown in Figure 5 are the next state entries which implement this condition. The specified states are noted on the K-map and their next state entries are left blank. The fault detector

y_1	y_2	y_3	y_4		
0	1	0	1	1	2
1	1	1	1	2	3
1	1	0	0	3	4
1	0	1	0	4	5
1	0	0	1	5	6
0	1	1	0	6	7
0	0	1	1	7	1

Table 4: Fault example

y_3y_4 \ y_1y_2	00	01	11	10
00		1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

	0111	3	1011
0010	1	1110	5
7	1101	2	0001
0100	6	1000	4

Figure 5: Cycle operation

2.1.10

output is noted in the top half of each cell with 1 denoting a fault state. The fault states show the next state entry in the bottom half of each cell. Notice that the states cycle among themselves: 0001 → 0010 → 0100 → 0111 → 1101 → 1110 → 1000 → 1011 → 0001.

A BTS circuit can easily be altered to enter the above cycle. A single change in any one of the constants feeding an input to the BTS network, like the one shown in Figure 3, will cause the circuit to enter the fault states. The input that effects the change can be an external signal to the controller. The important features of this fault checker are that it can be tested, there is no increase in hardware and the cycle test can be invoked externally.

4.2 Fail Safe Operation

A fail safe circuit is designed such that a fault can never produce an "unsafe" output. Whenever a fault occurs in a sequential circuit, the circuit must be forced into a well defined set of fault states with safe outputs. A circuit must not be allowed to assume random states, because unsafe outputs could be generated [9].

In an n -variable minimum distance-two state assignment, there can be no more than 2^{n-1} specified states; at least half of the total states are fault states. Under worst case conditions, a single fault will force the circuit into a fault state. For fail safe operation, the next state entries for the fault states must be specified in such a manner that they will prevent the circuit from re-entering any specified state. The circuit architecture defined earlier will allow the next state entries for fault states to be specified in any desired manner without a hardware penalty. A judicious choice of next state entries can provide the desired fail safe qualities.

Let S_0 be the state where all state variables are 0. A fail safe design must consist of the following elements:

- Minimum distance-two state assignment
- S_0 and all states a distance one from S_0 are specified as fault states
- All fault states are programmed with a next state entry of S_0
- Outputs are programmed to be safe in all fault states

Since there is no sharing of hardware, the occurrence of a single fault can immediately affect no more than one state variable. Fail safe operation is guaranteed for the following set of fault conditions. Let the circuit be in state S_i with the next state entry S_j under input I_p . The combination of S_i and I_p selects a unique pass transistor path that presents the next state value for S_j to the inputs of the flip-flops. Assume that a fault is propagated to cause faulty operation in a next state variable.

Stuck-at-fault within the BTS network A fault can occur in one of two locations:

1. A stuck-at-fault is present within the BTS network, including the primary input lines. In this case, the next state will be fault state $S_j \oplus e$, where e is an n -tuple of weight 1. Since the next state for all fault states is S_0 , the circuit transition is $S_j \rightarrow S_j \rightarrow S_0$. Since the next state for S_0 is S_0 , the circuit is safe.
2. A fault occurs at the output of the BTS network for state variable Y_k . In this case, fault state $S_0 \oplus e$ is assumed, where e is an n -tuple that is all 0 except bit position $k = 1$. Since the next state entry for this state is the fault state S_0 , the circuit will not transition further, and is therefore safe.

Consider the example in Figure 4 and the circuit shown in Figure 6. The next state entry for the fault states are denoted with an "F" and are coded 0 for S_0 . Let the current state be 6 (0110); the next state is 7 with code 0011. The pass transistor path corresponding to 0110 is enabled for each state variable, passing the code for state 7 to the output of the BTS network. If there is a stuck-at-1 fault along this path, then the possible next states are 1011, 0111 or 0011 depending on which next state variable is affected. Notice that a stuck-at-1 fault for either Y_3 or Y_4 is masked since a 1 is suppose to be passed. If states 1011 or 0111 are entered, then the next state will be 0000, unless the outputs of the BTS network for Y_1 or Y_2 are stuck-at-1, in which case the circuit will assume state 1000 or 0100. Since present state and next state are the same in all three cases, the circuit remains stable. A similar situation occurs for a stuck-at-0 fault except state S_0 is entered and $e = 0$.

Transistor stuck-on If a given transistor is stuck-on, then two paths are enabled at some node. Along one path is the code for the specified state, and along the other is the code for S_0 . If the two signals agree, the fault is masked. If they differ, then a conflict is present and the exact logic level is determined by the electrical characteristics of the network. If the path for code S_0 dominates and propagates a 0 to the output, then the circuit will assume a fault state. At this point one of two things can happen:

1. The fault state can enable an entirely different set of transistors causing the stuck-on fault to be isolated and guarantees that the circuit enters fault state S_0 .
2. The fault state does not isolate the stuck-on fault. S_0 will still be entered because only 0's are being passed when the circuit is in the fault state.

Continuing with the above example, let the circuit be in state 6. If the transistor controlled by y_4 in path 0111 is stuck-on, then there is a conflict between 0 (coming from f) and 1 (coming from 7) at the first node. If the 1 dominates the 0, the circuit will operate fault free. If the 0 dominates the 1, then the next state will be 0001 or 0010 if Y_3 or Y_4 are affected. Either of these states will force the circuit to S_0 and the operation is then fail safe.

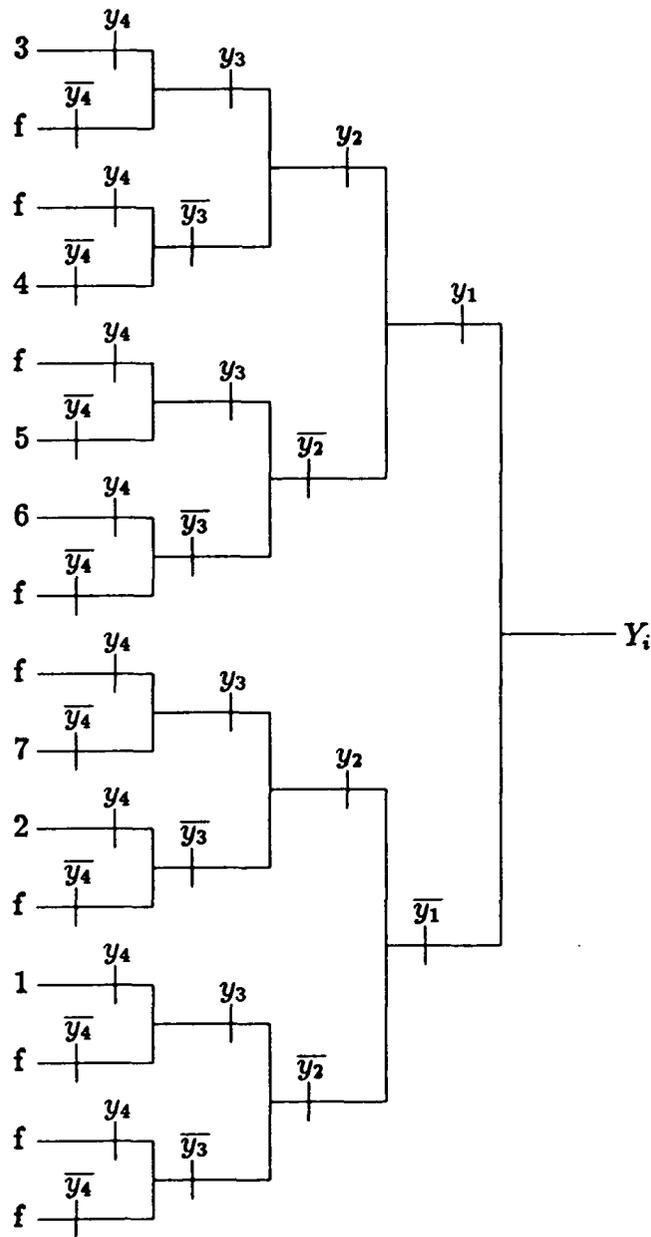


Figure 6: BTS network for fail safe operation.

State variable stuck-at-1 If the output of flip-flop y_i assumes a stuck-at-1 value, then an entire set of transistors controlled by y_i or its complement are turned on. Let the circuit be in state S_i where $y_k = 0$ and the output of the flip-flop for Y_k becomes stuck-at-1. The BTS circuit will respond as if the circuit is in fault state $S_i \oplus e$, where e is all 0 except for bit $k = 1$. The next state entry for this state is 0 for all state variables which will force the circuit to $S_0 \oplus e$. This state is another fault state with a next state entry of S_0 . Therefore, the circuit is stable and safe.

This circuit cannot be guaranteed to operate in a fail safe manner for stuck-open or for state variable stuck-at-0 faults. These faults can disable the pass transistor path that drives the flip-flops and can cause a tristate input to the flip-flops. A tristate input to the flip-flops may or may not force the circuit to a fault state. The circuit can remain in the same valid state until the charge at the BTS output node leaks off. Many clock cycles could occur and since the circuit is in a valid state, the fault detector and fail safe circuitry would not respond to the failure.

4.3 Fault Tolerance

A designer can achieve fault tolerance using either of the following procedures:

1. Place an error-correcting code on the state assignment as proposed by Meyer [8]. For single fault tolerance, all fault states adjacent to a specified state are encoded with the same next state entry as the specified state. If a single fault occurs, a fault state adjacent to the specified state is assumed. However, since the states adjacent to each specified state have the same next state entry, the circuit will transition to the proper next state, or at least to a state within a distance 1 of the specified state.
2. Use $n + 1$ safe circuits to achieve an n fault tolerant circuit [10]. Either a simple AND or a simple OR gate could be used to produce the fault tolerant output.

4.4 Adaptive Operation

One of the attractive features of the architecture proposed here is its adaptive nature. The constants, which are input to the BTS network, can either be derived from connections to V_{dd} and V_{ss} lines or they can come from a register or other storage cell. Destination constants (codes) coming from a register can be changed. With each change of the destination constants an entirely different set of transitions can be implemented.

If it would be possible to identify transitions that are affected by faults, then changing the flow table could be accomplished to avoid the faulty transition. For instance, if a given transition from state S_i produced a malfunction and it was determined that some transistor along the path decoding S_i had failed, then that particular pass transistor path could be avoided. A spare non-fault state S_s could assume the role of S_i and every transition to S_i could then transition to state S_s . This change can be effected with a new set of constants that define S_s as the next state for all predecessor states of S_i . Moreover,

all next state entries for S_i can be mapped to S_j . Therefore, state S_i is no longer entered and the particular pass transistor path that contained the fault is not used again.

5 Conclusion

A new architecture is presented that has attractive features for producing reliable sequential controllers. A design procedure that is applicable to VLSI is given for realizing the new architecture. Without any increase in hardware, a circuit can be designed to be safe. Through the addition of only one more state variable and its associated BTS network, a circuit can be realized that has real time fault detection and fail safe capabilities. The new architecture also allows for the implementation of adaptable circuits that are capable of initiating alternative transition sequences to replace those that are faulty. The adaptable nature of the circuit is achieved through no increase in the fail safe design hardware.

References

- [1] J. Venbrux, and N. Liu, "VLSI Chip-set for Data Compression Using the Rice Algorithm," NASA SERC Symposium on VLSI Design, Moscow, Idaho, pp.41-51, Jan. 1990.
- [2] S. Whitaker, S. Manjunath and G. Maki, "Sequence Invariant State Machines", submitted to the IEEE Journal of Solid State Circuits.
- [3] G. Peterson and G. Maki, "Binary Tree Structured Logic Circuits: Design and Fault Detection," Proceedings of IEEE International Conference on Computer Design: VLSI in Computers, pp. 139-144, Oct. 1984.
- [4] D. Radhakrishnan, S. Whitaker and G. Maki, "Formal Design Procedures for Pass-Transistor Switching Circuits," IEEE Journal of Solid State Circuits, pp. 531-536, Apr. 1985.
- [5] K. Cameron, "ACE: Automatic Centroid Extractor for Real Time Target Tracking", accepted for publication at IEEE Northcon, Seattle, Washington, Oct., 1990.
- [6] S. Whitaker, K. Cameron, P. Owsley and G. Maki, "Custom CMOS Reed Solomon Coder for the Hubble Space Telescope", accepted for publication in the IEEE Military Communications Conference.
- [7] R. Wickersham and G. Maki, "Safe Asynchronous Sequential Circuits," IEEETC Vol. C-23, pp. 494-500, May 1974.
- [8] J. F. Meyer, "Fault Tolerant Sequential Machines," IEEETC, Vol C-20, Oct. 1971.
- [9] G. Maki and D. Sawin, "Fail Safe Asynchronous Sequential Machines," IEEETC Vol. C-24, pp 675-677, June 1975.

- [10] D. Sawin and G. Maki, "Fault Tolerant Asynchronous Sequential Circuits," IEEETC, Vol. C-23, pp. 651-657, July 1974.

This research was supported in part by NASA under the NASA Space Engineering Research Center grant NAGW-1406 and by the Idaho State Board of Education under grants 88-038 and 89-041.

Efficient Design of CMOS TSC Checkers

Anita Biddappa, Manjunath K. Shamanna, Gary Maki and Sterling Whitaker
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - This paper considers the design of an efficient, robustly testable CMOS Totally Self-Checking (TSC) Checker for k -out-of- $2k$ codes. Most existing implementations use primitive gates and assume the single stuck-at fault model. The self-testing property has been found to fail for CMOS TSC checkers under the stuck-open fault model especially due to timing skews and arbitrary delays in the circuit. A new four level design using CMOS primitive gates (NAND, NOR, INVERTERS) is presented, which retains its properties under the stuck-open fault model. Additionally this method offers an impressive reduction ($> 70\%$) in gate count, gate inputs and test set size when compared to the existing method. This implementation is easily realizable and is based on Anderson's technique[1]. A thorough comparative study has been made on the proposed implementation and Kundu's[8] implementation and the results indicate that the proposed one is better than Kundu's[8] in all respects for k -out-of- $2k$ codes.

1 Introduction

Totally Self-Checking (TSC) checkers are circuits which detect errors concurrently under normal operation. They are used to check the validity of various codes of which m -out-of- n codes are very important. An m -out-of- n code is one in which all valid code words have exactly m ones and $n - m$ zeros. An k -out-of- $2k$ code is a special case of m -out-of- n code and is the least redundant code for error detection[11] and hence is of particular interest. This paper deals only with k -out-of- $2k$ codes.

Carter and Schnieder[4] were the first to propose Self-Checking circuits. Anderson and Metze[1] extended this concept to develop TSC checkers for k -out-of- $2k$ codes based on the conventional stuck-at fault model. The conventional stuck-at fault model is inadequate in defining certain failures in CMOS circuits. Hence, the fault model has been augmented to include stuck-open and stuck-on faults[14,3,5]. Furthermore, it has been shown that tests for stuck-open faults in CMOS combinational logic circuits could potentially be invalidated by arbitrary delays or due to timing skews in the input changes[7,12,8]. Tests which do not suffer from this potential invalidation are called *Robust Tests*.

Until now, only two approaches have been reported to realize robustly testable CMOS TSC checkers[6,8]. The approach followed by Jha and Abraham[6] has been found to be

inadequate[9]. Kundu and Reddy[8] have proposed a four-level realization of robust CMOS TSC checkers for k -out-of- $2k$, k -out-of- $2k + 1$, $k + 1$ -out-of- $2k + 1$ and $k + 1$ -out-of- $2k + 1$ codes. However, a robustly testable complex gate realization of TSC checker is yet to be reported. The main contribution of this paper is the design of a four level robustly testable CMOS TSC implementation based on Anderson's technique for k -out-of- $2k$ codes. This offers considerable savings ($> 70\%$) in gate count, gate inputs and test set size over the existing method.

2 Notations and Definitions

This section summarizes the notations and definitions used in this paper. The notation used by Kundu and Reddy[8] and Anderson and Metze[1] will also be followed here. A few of the notations are as follows:

- A: set of all valid m -out-of- $2m$ code words.
- vertex: binary word.
- n -vertex: binary word with n ones.
- f, g : outputs of the TSC checker.
- f_1 : subset of A such that $f_1=1$ and $g_1=0$ for any input from f .
- g_1 : subset of A such that $f_1=0$ and $g_1=1$ for any input from g .
- $\langle T_1, T_2 \rangle$: two pattern Stuck-open fault test.
- T_1 : initializing input for the stuck-open fault test.
- T_2 : test input for the stuck-open fault.
- $T(k_a > i)$: majority function which is true if the number of ones in the vertex k_a is greater than i .

Definition 1 A circuit is self testing if for every fault from the fault set, the circuit produces a noncode output for at least one code input.

Definition 2 A circuit is fault secure if for every fault from the fault set, the circuit never produces an incorrect code output for every code input.

Definition 3 A circuit is totally self checking if it is both self testing and fault secure.

Definition 4 A binary n -tuple is said to 1-cover (0-cover) another binary n -tuple if the former has ones (zeros) in every position the latter has ones (zeros).

Definition 5 A vertex X is a true vertex of a function F iff $F(X)=1$.

Definition 6 A true vertex V_p of a function F is called a minimal true vertex iff there does not exist another true vertex Y of the function, $Y \neq V_p$ such that V_p 1-covers Y .

Definition 7 A 0-cofactor (1-cofactor) of order n of a product term of uncomplemented variables is obtained by substituting n zeros (ones) for n ones (zeros) in the product.

If a minimal true vertex V_p has n literals and if a 1 at position k of V_p is changed to a 0 to obtain a 0-cofactor of V_p it is denoted by $V_p(k)$.

Definition 8 A 0-cofactor (1-cofactor) is a false (true) vertex.

Definition 9 A hazard is defined as the appearance of an undesired logic value (transient) at the output of a circuit which is produced due to one or more changes at the inputs of the circuit.

3 TSC Checkers

Smith[13] and Anderson and Metzger[1] have proposed the following conventions for TSC checkers;

1. All invalid code words of weight less than m are mapped to the checker outputs:
 $f = g = 0$.
2. All invalid code words of weight greater than m are mapped to the checker outputs:
 $f = g = 1$.
3. All valid code words of weight equal to m are mapped to the checker outputs: $f = 1$,
 $g = 0$ or $f = 0$, $g = 1$.

Let F be a minimal sum of products expression realized by a two level NAND-NAND logic network N , where the first level gates correspond to the minterms of F . Then we have the following properties.

Property 1: If a minimal true vertex V_p is input to N , then the output of the circuit and the outputs of all first level NAND's except the gate covering V_p are 1.

Property 2: If $V_p(k)$, a 0-cofactor, is input to N , then the outputs of all first level gates are 1 and that of the second level gate is 0.

Property 3: If a two pattern input sequence $\langle V_p, V_p(k) \rangle$ or $\langle V_p(k), V_p \rangle$ is applied to N , then only one input to any gate changes. Furthermore, the outputs of all first level gates except the one covering V_p are 1 (hazard-free) for both inputs in the two pattern sequence.

Property 4: If a two pattern input sequence consisting of two 0-cofactors of different orders is presented to N , such that one 0-cofactor covers the other, then the output of every first level NAND gate and the second level NAND gate is 1 (hazard-free) and 0 (hazard-free) respectively for both inputs in the two pattern sequence.

Property 5: If a two pattern input sequence consisting of a true vertex and a 1-cofactor or two 1-cofactors of different orders, such that one 1-cofactor covers the other is applied to N , then there exists a gate whose inputs are all 1's for both inputs in the two pattern sequence

implying that the output of the second level NAND is always 1 (hazard-free) for both the inputs.

3.1 Kundu's Technique

This technique yields a four level implementation which is robustly testable. Here the k -out-of- $2k$ code is partitioned into subsets f_1 and g_1 according to Smith's Criterion[13]. The checker outputs f and g are then expanded with respect to an input variable x_i , chosen arbitrarily such that,

$$\begin{aligned} f &= x_i f_{1i} + \hat{f}_{1i} \\ g &= x_i f_{2i} + \hat{f}_{2i} \end{aligned}$$

where f_{1i} , f_{2i} , \hat{f}_{1i} , \hat{f}_{2i} are functions of the variable x_i through x_n excluding x_i . The functions \hat{f}_{1i} , \hat{f}_{2i} , and f_{1i} , f_{2i} are realized by NAND-NAND and NOR-NOR circuits respectively.

This design is hard to realize for large values of k and also necessitates the use of a large number of gates with very high fan-in.

3.2 Anderson's Technique

To design a k -out-of- $2k$ TSC, the $2k$ input bits are first divided into two groups of equal length ($n_a = n_b = k$). The checker output functions f and g are realized as follows:

$$\begin{aligned} f &= \sum_{i=0}^k T(k_a \geq i)T(k_b \geq k - i) \Big|_{i=\text{odd}} \\ g &= \sum_{i=0}^k T(k_a \geq i)T(k_b \geq k - i) \Big|_{i=\text{even}} \end{aligned}$$

where k_a and k_b will refer to the number of 1's occurring in each group. This technique was limited to two level realization using AND and OR gates and was developed under the stuck-at fault assumption. Under the extended fault model the self-testing property, which is a necessary condition for a circuit to retain its TSC properties, is difficult to satisfy for TSC CMOS circuits. In fact, it was shown by Manthani and Reddy[10] that a two level realization of Anderson and Metze's TSC checker[1] is not robustly testable. We now proceed to present a four level CMOS gate implementation based on this technique.

4 CMOS Checker Implementation

First we present a robustly testable CMOS checker design using primitive gates.

Procedure 1

1. Partition the $2k$ input bits ($a_1, \dots, a_k, \dots, a_{2k}$) into two groups of equal length $A_1 = \{a_1, \dots, a_k\}$ and $A_2 = \{a_{k+1}, \dots, a_{2k}\}$.

2. Find f and g using the equations:-

$$f = \sum_{i=0}^k T(k_a \geq i)T(k_b \geq k - i) \mid_{i=\text{odd}}$$

$$g = \sum_{i=0}^k T(k_a \geq i)T(k_b \geq k - i) \mid_{i=\text{even}}$$

where k_a and k_b refers to the number of one's in A_1 and A_2 respectively.

3. Implement both f and g in NAND-NAND form with each individual majority function implemented in two level NAND-NAND logic except the terms $T(k_a \geq 1)$ or $T(k_b \geq 1)$ which are to be realized using NOR logic.

The four level implementation obtained using Procedure 1 is robustly testable. If the expressions for f and g in the above procedure are now expanded in the form of a true sum of products expression to yield a two level realization, then the TSC checker fails to be self-checking under stuck-open faults.

Testing Strategy

Consider Figure 1. To test for stuck-open faults at the inputs of the circuit block implementing $T(k_a \geq i)$, it is to be ensured that $T(k_b \geq k - i)$ should be 1 for $\langle T_1, T_2 \rangle$. Under these conditions, $T(k_a \geq p) \mid_{(p>i)} = 0$ and $T(k_b \geq q) \mid_{(q>k-i)} = 0$. Hence, all the other inputs of the NAND gate (N_4), excepting the path under test, are 1 (hazardfree) during $\langle T_1, T_2 \rangle$ ensuring robust testing. Similarly for testing stuck-open faults at the inputs of $T(k_b \geq k - i)$, $T(k_a \geq i)$ should be 1 for $\langle T_1, T_2 \rangle$. This implies that $T(k_a \geq p) \mid_{(p>i)}$ and $T(k_b \geq q) \mid_{(q>k-i)}$ are 0 during $\langle T_1, T_2 \rangle$ which ensures that all the inputs of the NAND gate (N_4), except the path under test, are 1 (transientfree). Thus the tests are robust in nature.

Theorem 1 *The four level circuit designed using Procedure 1 is robustly testable for all single stuck-open and stuck-at faults.*

Proof: Consider the expressions for f and g :

$$f = T(k_a \geq 1)T(k_b \geq k - 1) + T(k_a \geq 3)T(k_b \geq k - 3) + \dots + T(k_a \geq k - 1)T(k_b \geq 1)$$

..... if k is even.

$$f = T(k_a \geq 1)T(k_b \geq k - 1) + T(k_a \geq 3)T(k_b \geq k - 3) + \dots + T(k_a \geq k)T(k_b \geq 0)$$

..... if k is odd.

$$g = T(k_a \geq 0)T(k_b \geq k) + T(k_a \geq 2)T(k_b \geq k - 2) + \dots + T(k_a \geq k)T(k_b \geq 0)$$

..... if k is even.

$$g = T(k_a \geq 0)T(k_b \geq k) + T(k_a \geq 2)T(k_b \geq k - 2) + \dots + T(k_a \geq k - 1)T(k_b \geq 1)$$

..... if k is odd.

Each of the different implementations of f and g when k is even or odd will be considered separately for proving the circuit's robustness during testing. Chandramouli's [3] procedure for testing stuck-open faults is followed here. This envisages the application of a sequence of a pair of test vectors to detect the stuck-open faults in CMOS logic circuits. The first test vector initializes the output of the faulty gate and the second sensitizes and propagates the fault to the output. Furthermore, Chandramouli [3] has shown that the tests for all input stuck-open (ISOP) faults and output stuck-open faults (OSOP) at all primary gate inputs will test cover all the stuck-open faults in an Non-Internal Reconvergent fan-out circuit. Also, the test vectors for detecting the ISOP and OSOP faults are identical but reversed in their order of application. It should also be noted that, since the implementation is four level, when testing a primary input fault all the four gate levels must be sensitized.

4.1 Realization of g when k is even

Consider the realization of g with k even. Each majority function, $T(k_a \geq i) |_{i \neq 1, k}$ and $T(k_b \geq i) |_{i \neq 1, k}$, is realized by two levels of NAND gates. The third level NAND gate realizes $T(k_a \geq i)T(k_b \geq k - i)$ and the fourth level NAND gate realizes g . To simplify the proof, the whole circuit is partitioned into a finite number of subcircuits and then each one of these subcircuits is shown to be robustly testable.

Circuit corresponding to any $T(k_a \geq i)T(k_b \geq k - i)$

Consider an arbitrary lead y of any first level NAND gate corresponding to $T(k_a \geq i)$. Each first level NAND corresponds to a true vertex of the unate function $T(k_a \geq i)$. Select a two pattern test $\langle V_p, V_p(y) \rangle$. The test vector V_p is chosen such that all the i leads of the gate are 1's during T_1 implying that the weight of n_a is i and that of n_b is $k - i$. $V_p(y)$ applies a zero to the input y while all other inputs of this gate remain at 1, so that the weights of n_a and n_b are $i - 1$ and $k - i + 1$ respectively. According to Property 3, the second level NAND gate is sensitized robustly (all the sensitized leads remain at one during T_1 and T_2). Since V_p and $V_p(y)$ are 0-cofactors of the majority functions $T(k_a \geq p) |_{p \geq i+2}$, all the other inputs feeding the fourth level NAND are 1 during the test sequence $\langle V_p, V_p(y) \rangle$ and are hazard-free. Also the output of the second level NAND, $T(k_b \geq k - i)$, is 1 and is hazard-free. Hence, $\langle V_p, V_p(y) \rangle$ is a robust test for ISOP on the lead y . Similarly the reverse sequence $\langle V_p(y), V_p \rangle$ forms a robust test for OSOP fault corresponding to the lead y . In a similar manner, it can be shown that any input/output stuck-open fault corresponding to the first level NAND gates of the majority functions $T(k_b \geq k - i)$ can be robustly tested.

Circuit corresponding to $T(k_b \geq k)$ term

The term $T(k_b \geq k)$ corresponds to a single k input NAND from which an arbitrary lead u is selected. The test pattern $\langle V_p, V_p(u) \rangle$ is then applied. The test vectors are such that weights of n_a are 0 and 1 and that of n_b are k and $k - 1$ during T_1 and T_2 respectively.

Since the weight of n_a is never greater than 1, during the test sequence the outputs of all second level NANDs corresponding to $T(k_a \geq p) |_{p \geq 2}$ are 0 and hence the outputs

of all third level NAND's are 1 (hazard-free). Also the output of the gate corresponding to $T(k_a \geq k)$ is a 1 (hazard-free) during the test sequence. Since no transients occur in the circuit, the test pattern $\langle V_p, V_p(u) \rangle$ is a robust ISOP test for the lead u . Similarly, $\langle V_p(u), V_p \rangle$ is a robust OSOP fault test corresponding to lead u . On the same lines, it can be shown that the input gate implementing $T(k_a \geq k)$ is also robustly testable.

4.2 Realization of g when k is odd

This implementation is similar to the implementation of g when k is even. As usual, the terms $T(k_a \geq k)$ and $T(k_a, T(k_b \geq i)) |_{i \neq 1}$ are realized by NAND gates except $T(k_b \geq 1)$ which is realized by a k input NOR gate.

Circuits corresponding to any $T(k_a \geq i)T(k_b \geq k - i) |_{i \neq 1}$, and $T(k_b \geq k) |_{k \neq 1}$

The arguments when k is even are also valid here. Hence, these circuits can be robustly tested.

Circuit corresponding to $T(k_b \geq 1)$

Consider an arbitrary lead m of the NOR gate corresponding to $T(k_b \geq 1)$. A two pattern test $\langle U_s, U_s(m) \rangle$ is selected such that U_s applies 0 to all k leads of the NOR gate such that weight of n_b is 0 and $U_s(m)$ applies a 1 to the lead under question and 0 to the rest of the leads. Since the weight of n_b is 0 and 1 in T_1 and T_2 respectively, the output of all second level NAND gates corresponding to all $T(k_b \geq p) |_{p \geq 2}$ is always 0. Hence the outputs of all third level NAND gates except the one corresponding to the NOR gate under consideration are 1 (hazard-free) during the test sequence. Also the output of the NAND gate corresponding to $T(k_b \geq k)$ is 1. Hence, the test sequences $\langle U_s(m), U_s \rangle$ do not suffer from test invalidation. So, the circuit is robustly testable.

4.3 Realization of f when k is odd

This case is similar to that of g when k is odd. Hence using arguments similar to the one where k is odd in the realization of g , it can be shown that the circuit is robustly testable.

4.3.1 Realization of f when k is even

The majority functions $T(k_a \geq 1)$ and $T(k_b \geq 1)$ are implemented by NOR gates while the other majority functions are implemented by two level NAND logic.

Circuit corresponding to any $T(k_a \geq i)T(k_b \geq k - i) |_{i \neq 1, k-1}$

The arguments corresponding to the case when k is even in the realization of g are valid here. Hence this circuit is robustly testable.

Circuit corresponding to $T(k_b \geq 1)$

The arguments corresponding to the realization of g when k is odd are valid here. Hence this circuit is robustly testable.

Circuit corresponding to $T(k_a \geq 1)$

Using arguments similar to the case of $T(k_b \geq 1)$, it can be easily shown that this circuit is robustly testable.

QED.

The following example will demonstrate the design of the proposed TSC checker.

Example 1 *Design a CMOS TSC checker for a 3-out-of-6 code using Procedure 1*

The six input bits are partitioned into two groups: $A_1 = (a_1, a_2, a_3)$ and $A_2 = (a_4, a_5, a_6)$ where a_1, a_2, \dots, a_6 represent the input code bits. The expressions for f and g are given by:-

$$\begin{aligned} f &= T(k_a \geq 1)T(k_b \geq 2) + T(k_a \geq 3)T(k_b \geq 0) \\ &= (a_1 + a_2 + a_3)(a_4 a_5 + a_4 a_6 + a_5 a_6) + (a_1 a_2 a_3) \\ g &= T(k_a \geq 0)T(k_b \geq 3) + T(k_a \geq 2)T(k_b \geq 1) \\ &= (a_4 a_5 a_6) + (a_1 a_2 + a_1 a_3 + a_2 a_3)(a_4 + a_5 + a_6) \end{aligned}$$

The majority functions $T(k_a \geq 1)$ and $T(k_b \geq 1)$ in the realizations of f and g respectively are implemented using NOR logic while the remaining majority functions are implemented using NAND logic as shown in Figures 1 and 2. This four level implementation can be robustly tested using the guidelines presented in [3].

5 Comparisons

As mentioned before, there is only one method[8] currently available, for the design of robustly testable CMOS TSC checkers wherein the cases of k -out-of- $2k$, $k-1$ -out-of- $2k+1$, $k+1$ -out-of- $2k+1$ and k -out-of- $2k+1$ codes have been covered. The proposed method covers the case of k -out-of- $2k$ codes. A comprehensive study of the number of gates used by our method and Kundu's method[8] has been made and general relations have been established which are shown in Tables 1 through 4.

Table 5 illustrates the comparison between the proposed and Kundu's method[8] for $k=2, 3, 4, 5$ and 6 . The comparisons have been made with respect to gate inputs and test vector pairs. Figures 3 and 4 illustrate the percentage savings in test vector pairs and gate inputs respectively with increasing values of k . These graphs show an almost exponential nature with about 32 % and 60 % savings in gate inputs and test vector pairs for even the trivial case of $k=2$ and increases to well over 90 % for cases $k \geq 6$. The tremendous increase in gate inputs is not only a result of increased number of gates but also gates with high fan-in index. For a typical case of 5-out-of-10 code TSC checker, Kundu and Reddy's method[8] uses 258 gates with four gates having a fan-in of over 60. On the contrary, the proposed design uses 102 gates with 3 gates involving a maximum fan-in of 10.

Testing of the circuits as designed in [8] is complex and offers fewer choices of test vectors for testing a stuck-open fault, where as the proposed one is easily testable and offers more choices of test vectors for testing the same fault. Also, the probability that a fault is tested sooner is high. Assuming a random occurrence of test vectors, the probability

distribution that the circuit is tested is multinomial in nature . It can be easily shown that the probability of the proposed realization being tested completely in a shorter period of time is considerably more compared to Kundu and Reddy's realization[8]. Also, the waiting time is considerably less.

The proposed design is very simple and has been acknowledged by various researchers in this field. The partition of all valid code words into f_1 and f_2 as described in [8] is a bit tedious for larger values of k . Additionally converting f_{1i}, f_{2i} into the product of sums form from the sum of products form is highly cumbersome and laborious for larger values of k (> 4). No such Boolean simplification is needed for our realization. Also, the proposed design is very simple and is based on Anderson and Metzger's technique[1], the simplicity of which has been acknowledged by various researchers in this field.

As opposed to our method, the amount of fan-in of the second level NOR and NAND gates in Kundu's method[8] is considerable and increases dramatically with k . Furthermore, the proposed method offers a large reduction in chip area because of the considerably fewer number of gates and gate inputs.

6 Conclusions

In this paper we have considered the problem of designing robustly testable CMOS TSC checkers for k -out-of- $2k$ codes. The robust testability criterion under the extended fault model has been achieved by following a four level implementation rather than the regular two level implementation. A comparison between the proposed method and the presently available method[8] has been carried out and results tabulated. Savings of over 90 % in the number of gate inputs and test vector pairs have been achieved for even simple cases ($k=6$). The savings in the number of gate inputs and test vector pairs appear to be exponentially increasing reaching over 90 % for even small values of k ($k \geq 6$). The proposed design not only offers considerable savings in gates, test vectors, chip area, etc. as compared to the presently available method, but it also offers the benefit of simplicity of design.

References

- [1] Anderson, D. A., and Metzger, G., " Design of Totally Self-Checking Check Circuits for m -out-of- n Codes", *IEEE Transactions on Computers*, vol. 22, 1973, pp. 263-269.
- [2] Bose, B., and Lin, D. J., "PLA Implementation of k -out-of- n Code TSC Checker", *IEEE Transactions on Computers*, vol. 6, 1984, pp. 43-68.
- [3] Chandramouli, R., " On Testing Stuck-Open Faults", *Proceedings of 13th Fault Tolerant Computing Symposium*, 1983, pp. 258-265.
- [4] Carter, W. C., " Design of dynamically checked Computers", *IFIP*, vol. 2, 1968, pp. 878-883.

- [5] El-Ziq, Y. M., and Cloutier, R. J., "Functional-level Test Generation for Stuck-open Faults in CMOS VLSI", *International Test Conference*, 1981, pp. 536-546.
- [6] Jha, N. K., and Abraham, J. A., "Totally Self-Checking CMOS Circuits Using a Hybrid Realization", *International Symposium on Fault Tolerant Computing*, 1985, pp. 154-158.
- [7] Jain, S. K., and Agrawal, V. D., "Test Generation for MOS Circuits Using D-Algorithm", *Proceedings of the Twentieth Design Automation Conference*, 1983, pp. 64-70.
- [8] Kundu, S., and Reddy, S. M., "Design of TSC Checkers for Implementation in CMOS Technology", *International Conference on Computer Design*, 1989, pp. 116-119.
- [9] Kundu, S., and Reddy, S. M., "On the Design of TSC CMOS Combinational Logic Circuits", *International Test Conference*, 1986, pp. 486-499.
- [10] Manthani, S. R., and Reddy, S. M., "On CMOS Totally Self-Checking Circuits", *International Test Conference*, 1984, pp. 866-877.
- [11] Paschalis, A. M., Nikolos, D., and Halastsis, C., "Efficient Modular Design of TSC Checkers for M-out-of-2M Codes", *IEEE Transactions on Computers*, vol. 3, 1988, pp. 301-309.
- [12] Reddy, S. M., Reddy, M. K., and Kuhl, J. G., "On Testable Design for CMOS Logic Circuits", *Proceedings of 1983 International Test Conference*, 1983, pp. 435-445.
- [13] Smith, J. E., "The design of totally self-checking check circuits for a class of unordered codes", *Journal of Design Automation and Fault-Tolerant Computing*, vol. 1, 1977, pp. 321-342.
- [14] Wadsack, R. L., "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits", *Bell System Technical Journal*, vol. 57, 1978, pp. 1449-1474.

k	Gate inputs		Test vector pairs	
	Kundu's	Proposed	Kundu's	Proposed
2	24	15	30	12
3	81	40	103	34
4	344	99	415	88
5	1497	226	1749	214
6	6230	515	7364	500

k	% Savings in Gate inputs	% Savings in Test vector pairs
2	31.8%	60%
3	50.6%	67%
4	71.2%	78.8%
5	84.9%	87.8%
6	91.7%	93.2%

Table 1: Comparison of the Proposed method and Kundu's method

<i>Gate type</i>	<i>Inputs</i>	<i>Number of gates</i>
NOR	k	$1 + \sum^k C_r^{k-1} C_{k-r} _{2 \leq r \leq k}$
NOR	k-1	1
NOR	$1 + \sum^k C_r^{k-1} C_{k-r} _{\substack{2 \leq r < k \\ r \equiv \text{even}}}$	1
NOR	$1 + \sum^k C_r^{k-1} C_{k-r} _{\substack{3 \leq r < k \\ r \equiv \text{odd}}}$	1
NAND	$({}^{2k}C_k/2) - 1 - \sum^k C_r^{k-1} C_{k-r} _{2 \leq r \leq k-2}$	1
NAND	k	$({}^{2k}C_k/2)$
NAND	n	1
NAND	2	4
Invertors		2

Table 2: Hardware requirements for Kundu's Realization when k is even

<i>Gate type</i>	<i>Inputs</i>	<i>Number of gates</i>
NOR	k	$1 + \sum^k C_r^{k-1} C_{k-r} _{2 \leq r \leq k}$
NOR	$k-1$	1
NOR	$\sum^k C_r^{k-1} C_{k-r} _{\substack{2 \leq r \leq k \\ r \equiv \text{even}}}$	1
NOR	$2 + \sum^k C_r^{k-1} C_{k-r} _{\substack{3 \leq r \leq k \\ r \equiv \text{odd}}}$	1
NAND	$1 + \sum^k C_r^{k-1} C_{k-r} _{\substack{1 \leq r < k \\ r \equiv \text{odd}}}$	1
NAND	$(2^k C_k / 2) - 1 - \sum^k C_r^{k-1} C_{k-r} _{\substack{1 \leq r < k \\ r \equiv \text{odd}}}$	1
NAND	k	$m = (2^k C_k / 2)$
NAND	$(2^k C_k / 2)$	1
NAND	2	4
Invertors		2

Table 3: Hardware requirements for Kundu's Realization when k is odd

<i>Gate type</i>	<i>Inputs</i>	<i>Number of gates</i>
NAND	k	4
NAND	$r \mid \begin{matrix} 2 \leq r < k \\ r \equiv \text{even} \end{matrix}$	${}^k C_r$
NAND	$r \mid \begin{matrix} 3 \leq r < k \\ r \equiv \text{odd} \end{matrix}$	${}^k C_r$
NAND	${}^k C_r$	4
NAND	$(k-r)$	$2({}^k C_r)$
NAND	2	$(k-1)$
NAND	$k/2 + 1$	1
NAND	$k-1$	$2k$
NAND	$k/2$	1
NOR	k	2
Invertors		2

Table 4: Hardware requirements for Proposed Realization when k is even

<i>Gate type</i>	<i>Inputs</i>	<i>Number of gates</i>
NAND	k	4
NAND	$k-1$	$2k$
NAND	$r \mid 1 \leq r < k$	$({}^k C_r)$
NAND	${}^k C_r$	4
NAND	$(k-r)$	$2({}^k C_r)$
NAND	2	$(k-1)$
NAND	$(k+1)/2$	1
NAND	$(k)/2$	1
NOR	k	2
Invertors		2

Table 5: Hardware requirements for Proposed Realization when k is odd

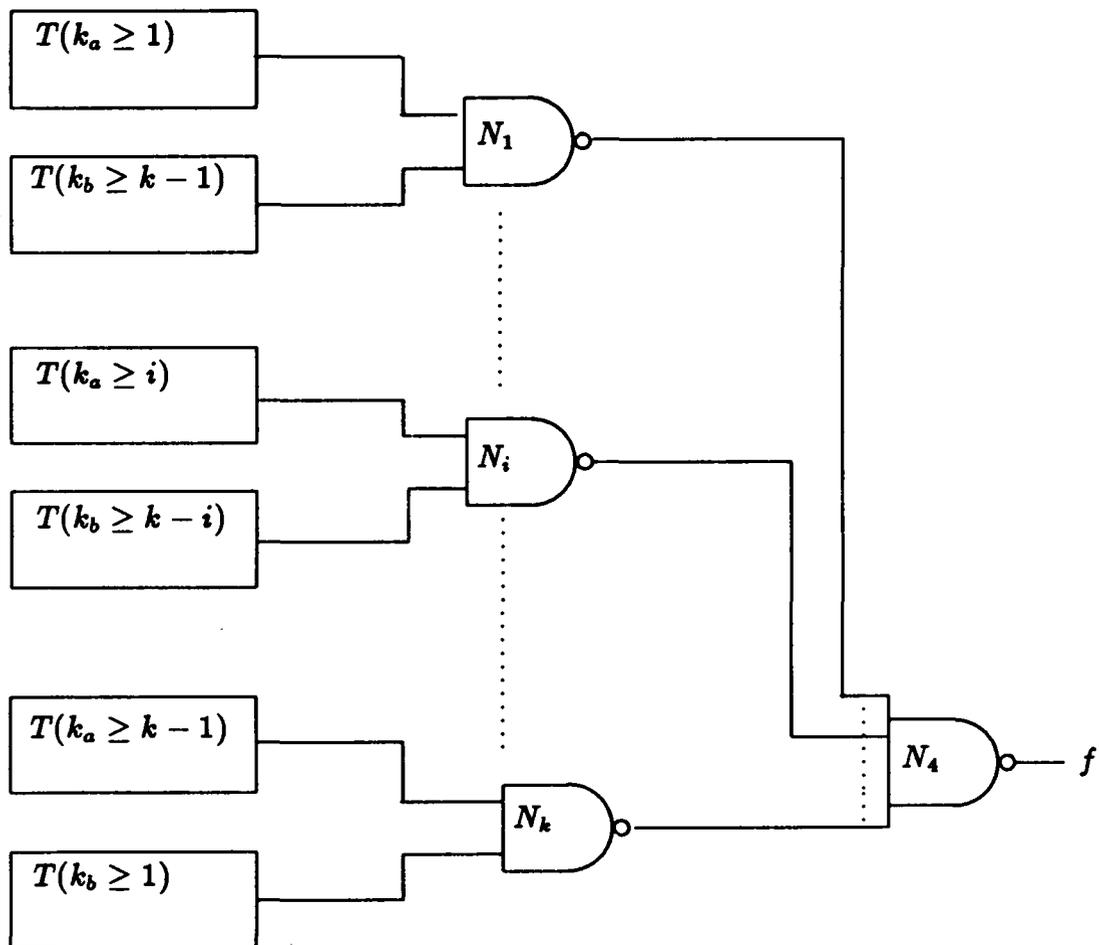


Figure 1: Illustration of the General Testing Strategy

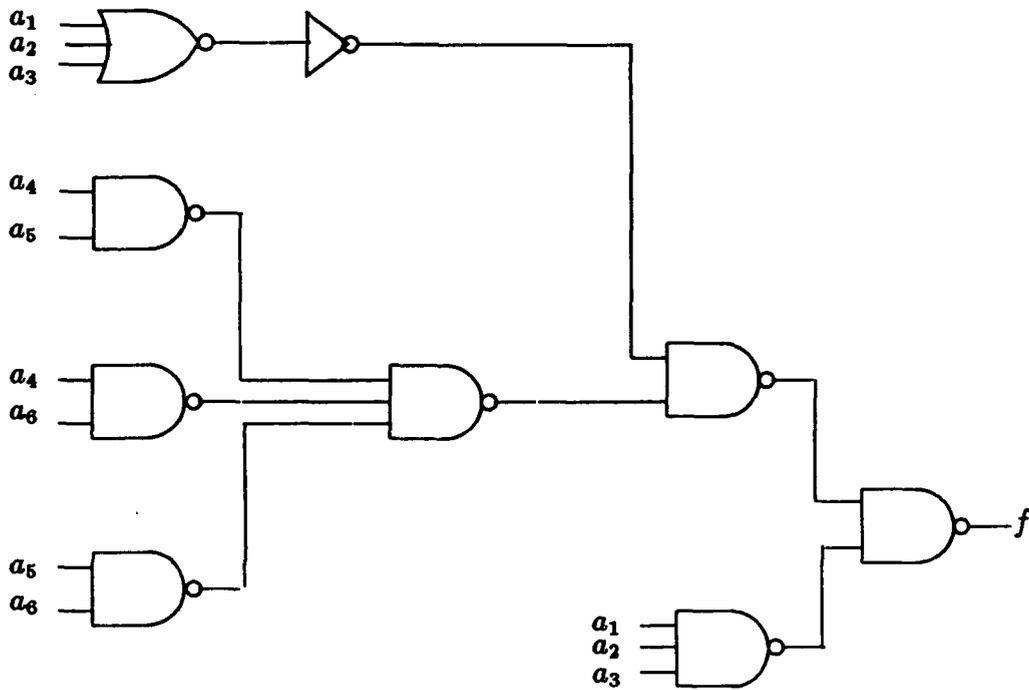


Figure 2: TSC Checker output f for Example 1

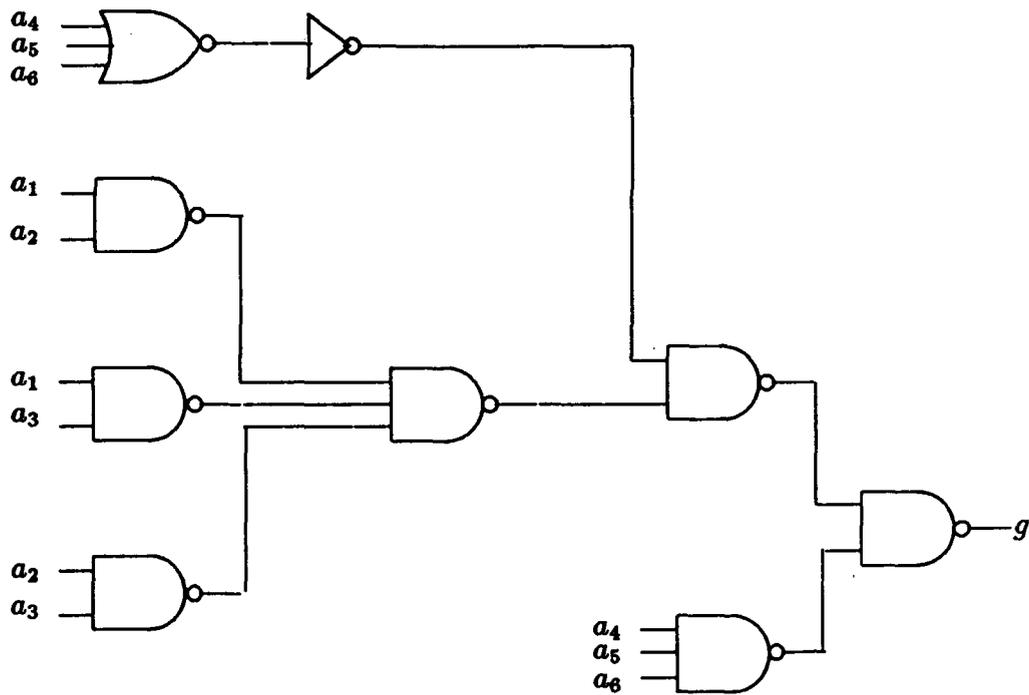


Figure 3: TSC Checker output g for Example 1

A Fail-Safe CMOS Logic Gate

V. Bobin and S. Whitaker
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract- This paper reports a design technique to make Complex CMOS Gates fail-safe for a class of faults. Two classes of faults are defined. The fail-safe design presented has limited fault-tolerance capability. Multiple faults are also covered.

1 Introduction

All fault-tolerance techniques are based on fault models. Most of the work in fault-tolerance has been directed towards a fault-set which consists of the so-called *stuck-at* faults; *stuck-at 0* and *stuck-at 1* [1]. This fault model assumes that a line or net is stuck at the logical value 0 or 1 as a result of the failure. Traditionally, in the treatment of fault-detection and fault-tolerance, additional assumptions have been made, such as mutual independence of faults, identical probability for all possible faults, and that of a single fault in a given circuit. At higher levels of integration, all these assumptions become unacceptable. Many of the failures are the results of process characteristics and are technology- and layout-dependent. Thus all faults are not equally probable. In a given circuit, there may be some faults that are more likely than others. It is well known that all physical defects cannot be represented by the simple stuck-at model. A single defect at the physical level could result in multiple faults throughout the circuit especially when it affects a signal line feeding multiple points. Faults of structural origin may be better modeled as *shorts* or *opens* [1].

Redundancy at the gate level as well as careful coding of states have been used by many schemes to achieve logical fail-safeness [2,3]. The dominant integrated technology today is CMOS. In CMOS, two of the most likely faults are due to the *stuck-on* and *stuck-open* failures of individual transistors [4]. In this paper, the effects of these types of MOS transistor failures on the functioning of complex CMOS logic gates are considered. The respective faults are termed transistor stuck-on faults and transistor stuck-open faults. The effect of stuck-at faults on input signal lines is to make transistors either stuck-on or stuck-open. Thus stuck-on and stuck-open faults of the MOS transistors and stuck-at faults on signal lines make up the fault-set addressed by this paper. A design method which makes the logic gate fail-safe is suggested. The effects of specific classes of multiple faults are also taken into account. There are two specific faults that cannot be handled by the fail-safe design presented here. However, in general, this design is fail-safe for multiple faults also. It is seen from simulations that in several cases, the gate design presented here is fault-tolerant, but fault-tolerance capability depends on the sizes of the individual transistors and also on their configuration.

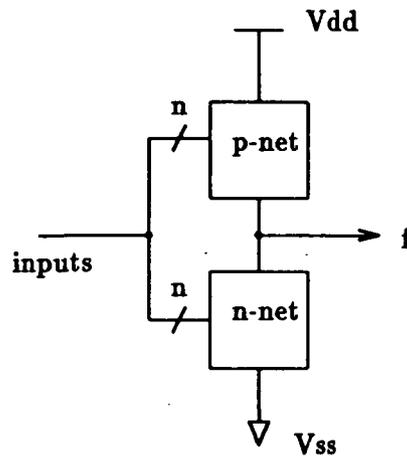


Figure 1: Complex CMOS Logic Gate

2 Complex CMOS Logic Gates

The general structure of complex CMOS logic gates is as shown in Figure 1. It consists of a network of pMOS transistors called the p-net or the pull-up network connected between the Vdd bus and the output node and a network of nMOS transistors called the n-net or the pull-down network connected between the Vss bus and the output node. The input signals are applied to both the pull-up and pull-down network. During normal operation only one path, either from Vdd to the output or from Vss to the output is enabled at any given time by the input combination applied [5]. Thus the output is pulled high to Vdd or pulled low to Vss leading to the logical operation. Since there is no completed path from Vdd to Vss during normal operation, there is no static power dissipation in the gate which is one of its desirable properties. There is however, dynamic power dissipation during the switching of the gate from one logic state to another because the capacitance at the output node must be charged to the proper logic level.

There are parasitic capacitances associated with each node in the network as well as the MOS transistor gate terminals where the inputs are applied. These capacitances cause delays during switching of the logic gates, which are proportional to the magnitude of the capacitances. The speed of operation of the gate is determined by the sizes or W/L ratios of the transistors, the configuration of the transistors, and the magnitude of the capacitances in the circuit. The low to high switching speed is governed by the size and configuration of the pMOS transistors in the pull-up network whereas the high to low switching speed depends on the size and configuration of the nMOS transistors in the pull-down network [5].

3 Fail-Safe Logic Design

Fail-safe logic gates are defined as follows [2,6].

Definition 1 A logic gate is said to be 0-fail-safe (1-fail-safe) if any failure causes only incorrect 0 (1) output. An output 1 (0) is always correct.

This means that upon failure, the 0-fail-safe (1-fail-safe) circuit assumes a 0 (1) output state. Thus an output of 0 (1) could be faulty or correct whereas an output of 1 (0) is always correct.

3.1 The Static Latch

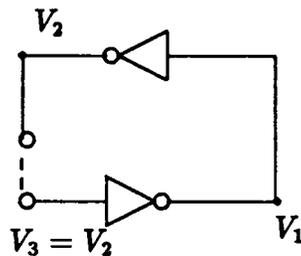


Figure 2: The Static Latch

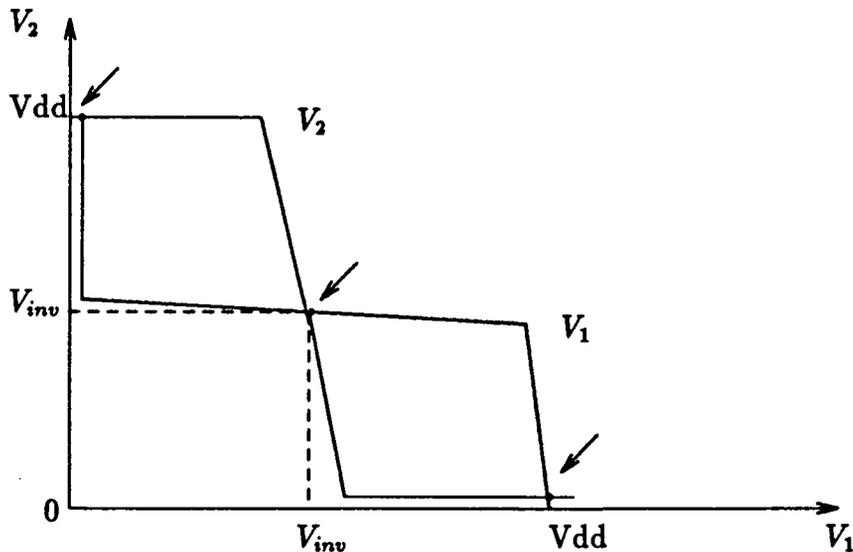


Figure 3: Three Static Solutions of Static Latch

A latch is used in a digital circuit to hold a logic state. The static latch remembers its last state until a new input is applied and as long as the power supply is on. The static latch contains elements with power gain that continually restore the integrity of the signals representing the state. The static latch is analogous to restoring logic [7]. The simplest form of static latch consists of a pair of cross-coupled inverters as shown in Figure 2.

The bistable latch has three quasi-static solutions as shown in Figure 3. In Figure 3, the solutions are found graphically from the individual transfer curves of the two inverters. As can be seen from the figure, the static solution in the middle is an unstable equilibrium

2.3.4

point, since a slight perturbation about this point pushes the latch to one of its stable points. If the two inverters of the latch are identical, then by symmetry, the unstable equilibrium point is where both the nodes are at the same voltage V_{inv} . From the transfer curves of the inverters it can be seen that unless the input voltage is very close to V_{inv} , the inverter output is a valid logic level, either high or low. This property is used in the fail-safe gate described next.

3.2 Fail-Safe Complex CMOS Gate

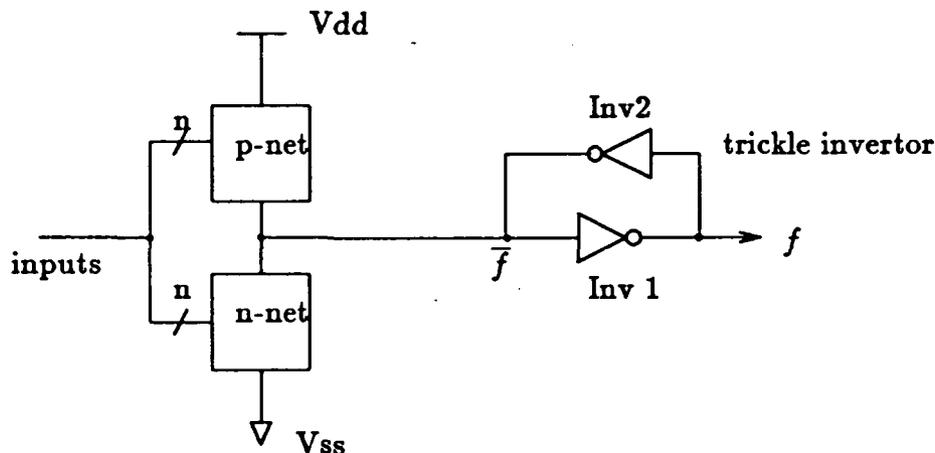


Figure 4: Fail-Safe Complex CMOS Gate

The fail-safe complex CMOS gate has the same general structure as the regular complex CMOS gate except for the addition of two inverters Inv1 and Inv2 connected back to back at the output node. These two inverters connected back to back constitute a static latch structure at the gate output.

Definition 2 A *Trickle Inverter* is a CMOS inverter whose transistors have W/L ratios less than 1.

The inverter Inv2 is a trickle inverter so that its output can be overdriven by the output of the p-net or the n-net. The gate output is determined solely by the input combination applied. This structure is shown in Figure 4. The purpose of the static latch is to provide the necessary positive feedback to make the gate output a definite logic level in the event of MOS transistor failures.

Definition 3 A *Safe Gate* is a complex CMOS logic gate with two inverters connected back to back at its output node. The inverter whose output is connected to the output node of the regular complex CMOS structure is a trickle inverter. The input node of the trickle inverter is driven by the primary output node of the safe gate.

A fail-safe complex CMOS gate is a safe gate.

Definition 4 *A Basic Gate is a safe gate with the static latch at the output removed; i.e., it is the complex CMOS gate from which the safe gate is derived.*

Thus the basic gate is designed using the complement of the function \bar{f} rather than the function f itself. If f is 0-fail-safe, \bar{f} is 1-fail-safe and vice versa.

The circuit in Figure 5 is a fail-safe complex CMOS gate which implements the function shown. SPICE simulations were carried out on this circuit to study the effects of various faults on the logical operation of the gate. The SPICE simulations used parameters from a typical $2\mu\text{m}$ process to model the nMOS and pMOS transistors. As an example for the effect of a fault on the operation of the *safe gate*, assume that the nMOS transistor m10 has failed by being stuck-on solidly. This failure is modeled by turning the transistor permanently on. Its gate is kept at Vdd (5V) permanently. Consider the input transition from $[A, B, C] = [0, 1, 1]$ to $[A, B, C] = [0, 1, 0]$. The gate output f should change from a logic level of 1 to a logic level of 0. SPICE simulation of this transition shows that \bar{f} changes from 0V to 2.93V, whereas f does indeed change from 5V to 0.1V. Thus in this case, the fail-safe gate has tolerated this stuck-on fault.

Fault-tolerance capability depends on the actual circuit configuration and the relative sizes of the transistors. It also depends on the extent of the failure; in other words, on the fault model. However, fail-safeness is independent of these. It is seen that f which is the output of Inv1 gives a better logic level compared to the output of the *basic gate*. This is true in general, because any failure in the *basic gate* can lead to a degradation of the logic level at the node \bar{f} . This is the reason why the output of Inv1 is designated the primary output of the fail-safe CMOS gate. It is clear that if the transistors of Inv1 fail, the output of the *safe gate* can be at an undefined logic level. Therefore in this discussion it is assumed that Inv1 is free of faults.

If one of the nMOS transistors in the fail-safe complex CMOS gate described here (excluding the one in Inv1) is stuck-on, the input to Inv1 can approach 0 erroneously, but never 1. This holds if several of the nMOS transistors are stuck-on also. The same is the case when several of the pMOS transistors (except the one in Inv1) are stuck-open or when both occur simultaneously, i.e., pMOS transistors stuck-open and nMOS transistors stuck-on. These faults are also analogous to the input signal lines being stuck-at-1. In all these cases, the primary output of the fail-safe complex CMOS gate can be made 1 erroneously, but never 0.

In the fault-set addressed in this paper, there are six possible kinds of faults; nMOS transistor(s) stuck-on, nMOS transistor(s) stuck-open, pMOS transistor(s) stuck-on, pMOS transistor(s) stuck-open, gate input(s) stuck-at 1, and gate input(s) stuck-at 0. These elements of the set are grouped into two classes of faults as defined below.

Definition 5 *If nMOS transistors are stuck-on or pMOS transistors are stuck-open or both happen simultaneously, then Class A faults occur. This includes gate inputs stuck-at 1.*

Definition 6 *If pMOS transistors are stuck-on or nMOS transistors are stuck-open or both happen simultaneously, then Class B faults occur. This class also includes gate inputs stuck-at 0.*

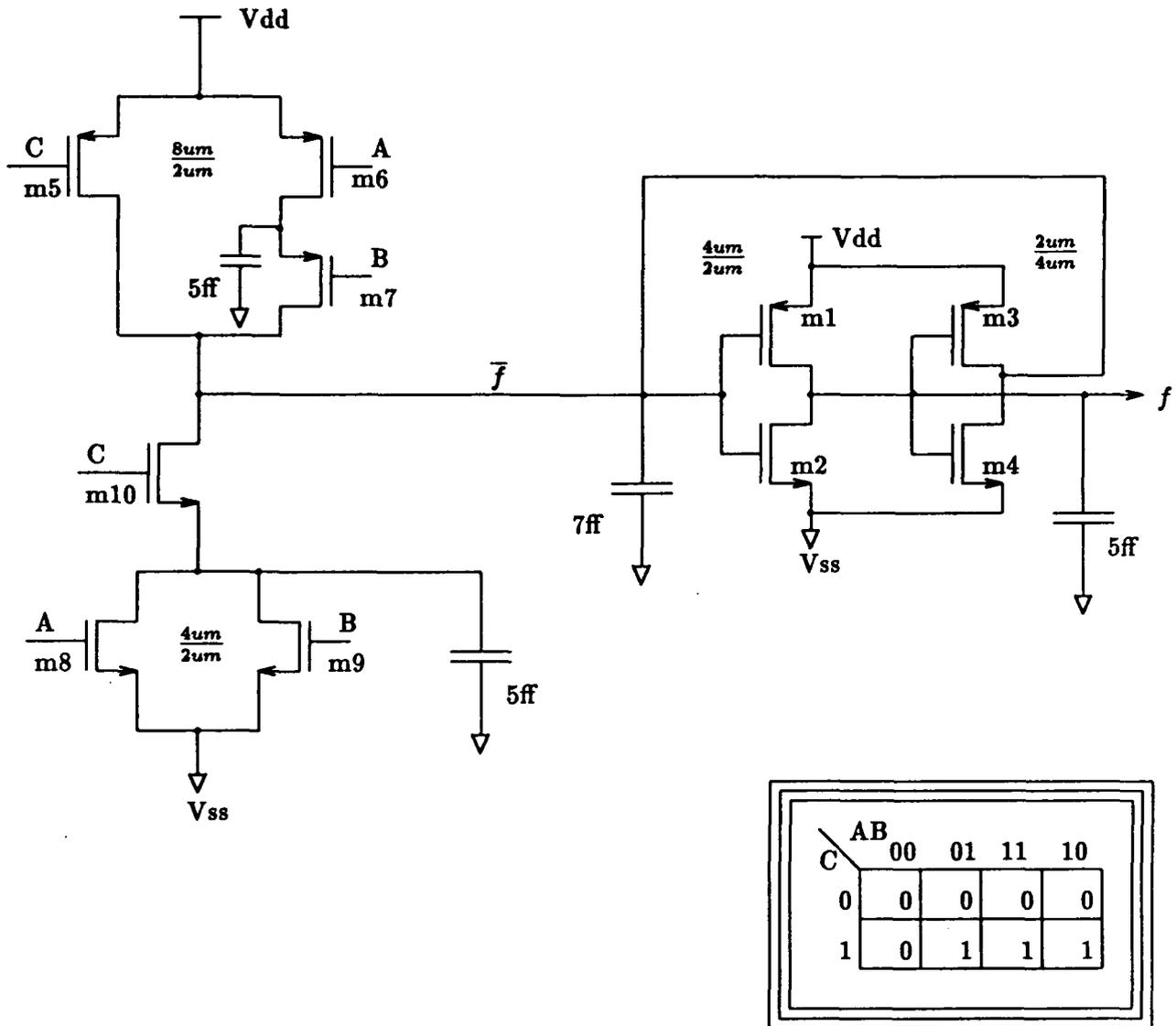


Figure 5: Fail-Safe Complex CMOS Gate Example

In the discussion on static CMOS latches, it was mentioned that the output of the static latch is a valid logic level except for a very small range of input voltage around its unstable equilibrium point. In the case of the fail-safe complex CMOS gate, the static latch structure consists of Inv1 and Inv2. Since Inv2 is a trickle inverter, the voltage level at the node \bar{f} is determined mainly by the transistors in the *basic gate*. Therefore the voltage at the primary output (node f) of the fail-safe complex CMOS gate depends on the voltage at node \bar{f} and the voltage transfer characteristic of Inv1. For the Inv1 of Figure 5 with $V_{dd}=5V$, SPICE simulation shows that the output is a valid 1 (greater than 4.1V) for an input that ranges from 0V to 1.85V and the output is a valid 0 (less than 0.9V) for an input range from 2.03V to 5V. Thus the output of the *safe gate* gives a valid logic level as long as the input node of Inv1 is not forced to a voltage between 1.85V and 2.03V due to failure. Since this is a small range of voltage, the probability of this occurring is small. The probability of an invalid logic level occurring at the output of the *safe gate* due to transistor failures depends on the sharpness of the voltage transfer characteristics of Inv1 and Inv2; the probability becomes smaller as the inverter characteristics become sharper.

The primary output of the fail-safe complex CMOS gate is a logic 1 or a logic 0 even in the presence of faults due to the positive feedback arrangement at the output. When *Class A* faults occur, the primary output of the gate can become 1 erroneously, but not 0. Thus for *Class A* faults, the *safe gate* is 1-fail-safe. It follows that for *Class B* faults, the *safe gate* is 0-fail-safe. Within a given class of faults, multiple faults are also covered; i.e., the *safe gate* remains fail-safe even for multiple transistor faults as long as *Class A* and *Class B* faults do not occur simultaneously. This is because the gate is fail-safe for a given class of faults and this property is maintained irrespective of the number of faulty transistors.

When *Class A* and *Class B* faults occur simultaneously, the *safe gate* loses its fail-safe property. In the case where pMOS transistors and nMOS transistors are stuck-open at the same time, the primary gate output can behave in several different ways.

1. If both the n-net and the p-net are stuck-open in the basic gate, the primary output of the *safe gate* retains its previous value because of the feedback arrangement. The output need not be safe.
2. If both transistors in the inverter Inv1 are stuck-open, the primary output floats, but the output node of the basic gate is true. Thus the primary output is not safe since a floating output is not safe.
3. The primary output is not affected when both transistors of the trickle inverter Inv2 are stuck-open. As long as the p-net and the n-net in the basic gate are not stuck-open at the same time, the primary output is safe.

Besides the three possibilities discussed, there are other ways in which the primary gate output can behave depending on the positions of the transistors stuck-open. The other possible way in which *Class A* and *Class B* faults occur at the same time is when nMOS transistors and pMOS transistors are stuck-on at the same time. For this case, in general, the primary gate output depends on the relative resistances of the p-net and the n-net. The primary output is not safe in this case.

The fail-safe complex CMOS gate design is useful when one class of faults is more probable than the other. The distinction between *Class A* faults and *Class B* faults underlines the fact that all faults are not equally probable. The probability of occurrence of different faults depends on the environment and also on the manufacturing process and layout of the integrated circuit. For example, in an environment where the MOS devices are exposed to ionizing radiation, their threshold voltage varies in the negative direction (the nMOS threshold voltage "rebounds" or starts varying in the positive direction at very high total dose levels, about 10^6 rads(Si)) [8]. Thus nMOS devices are likely to be stuck-on whereas pMOS devices are more likely to be stuck-open. Similarly, in certain manufacturing processes, more than 60 percent of MOS chip failures were attributed to the positive charges trapped in the gate oxide [9]. Thus it can be seen that for both the examples given, *Class A* faults are more probable than *Class B* faults and fail-safe design can be done accordingly.

4 Modeling Transistor Faults

The transistor fault-model should reflect the actual mechanism of failure of the device. From the discussion on the threshold voltage variation of MOS transistors when exposed to ionizing radiation, it is seen that nMOS devices tend to turn on more easily, whereas pMOS devices become harder to turn on. This is true until the nMOS transistors begin to "rebound", and this does not happen until the total dose level becomes extremely high. Another effect is that nMOS transistors start to exhibit a leakage current from drain to source, which is relatively small in magnitude. Thus in the worst case, nMOS devices could be stuck-on whereas pMOS devices could be stuck-open in a radiation environment. In reality however, they are likely to be made *partially stuck-on* and *partially stuck-open*. This means that they are not solidly on in the digital sense, or totally off, but are weakly conducting in the stuck-on condition and can conduct very weakly in the stuck-open condition.

For SPICE simulation purposes, *Class A* transistor faults occurring in a radiation environment were modeled by changing the threshold voltage of the transistors. In the threshold voltage model, nMOS transistors stuck-on were simulated by changing their threshold voltage to about -0.2 Volts, so that they were slightly on even with a gate voltage of about 0 Volts. Using the same principle, pMOS transistors stuck-open were simulated with a threshold voltage of about -5.5 Volts, so that they were off even with a gate voltage of about 0 Volts applied to their gates, with a Vdd of 5 Volts. Thus in this model, nMOS transistors stuck-on had a high resistance. The actual values of the modified threshold voltages used, have no effect on the fail-safeness of the logic gate, but have a bearing on the fault-tolerance capability of the gate. This threshold voltage model has nMOS transistors stuck-on with a high equivalent resistance and pMOS transistors totally non-conducting in their stuck-open condition. One consequence of the nMOS transistors stuck-on having a high resistance was that the primary output of the *safe gate* tolerated the nMOS transistor stuck-on faults. These faults would not have been tolerated if the transistors were stuck-on with very low resistance. The fault-tolerance capability also

depends on the sizes of individual transistors in the circuit. The pMOS transistors stuck-open were totally non-conducting in the model used. Thus the faults were not tolerated, although the primary output still remained safe. If the pMOS transistors conduct to a certain extent in their stuck-open state, then some of the faults would be tolerated by the *safe gate*.

As an example for the fault simulation using the threshold model, consider the circuit in Figure 5. Consider the input transition from $[A, B, C] = [0, 1, 1]$ to $[A, B, C] = [0, 1, 0]$. SPICE simulation of this transition with the threshold voltages of all nMOS transistors (including Inv1) set at -0.2V shows that the output f changes from 4.99V to 0V, whereas \bar{f} changes from 0V to 4.99V. It should be noted here that the nMOS transistors are only partially stuck-on in this case, so that the fault was easily tolerated.

Simulation of the faults where the transistors are stuck-on solidly, or totally stuck-open, was done by keeping the gates of the corresponding transistors at the respective logic level. The stuck-at faults on the input signal lines were also simulated in a similar manner. Thus transistors stuck-on solidly or completely stuck-open were simulated by keeping their gates stuck-at-1 or stuck-at-0. In Figure 5 assume m5 stuck-open completely. For the input transition of the previous example, SPICE simulation shows that the primary output remains steady at 5V. Thus the fault is not tolerated, but the 1-fail-safe property for the *Class A* fault is maintained. The output is safe.

A simple model for transistors stuck-open and stuck-on would be to replace them with resistances. This is based on the principle that in the steady state (dc), the transistor can be replaced by a resistance of appropriate value if its drain current and drain to source voltage are known. SPICE simulation of a typical nMOS transistor with a W/L ratio of $\frac{4\mu m}{2\mu m}$ shows that it has a dc resistance varying from 1K to 3K in the "on" state. Thus the conducting transistor has a resistance of the order of a Kilohm. If it is assumed that a stuck-open transistor has a resistance that is 2 orders of magnitude higher, and a stuck-on transistor has a resistance that is an order of magnitude lower in the worst case, then the stuck-open transistor can be replaced by a 100K resistor, and the stuck-on transistor can be replaced by a 100 Ohms resistor. In the circuit of Figure 5 assume m10 stuck-on with a resistance of 100 Ohms. For the input transition described in the two previous examples, SPICE simulation of the fail-safe complex CMOS gate with the stuck-on transistor replaced by the 100 Ohms resistor shows that the primary output f remains at 5V, while the node \bar{f} goes from 0V to 1.02V. Thus the primary output remains safe, even though the fault is not tolerated. It must be noted that for the device size considered, 100 Ohms is a rather small resistance even for a transistor conducting heavily. It was interesting to note that the pMOS stuck-open faults in the two inverters of the *safe gate* were tolerated when the high resistance model was used. The stuck-open fault in the *basic gate* was not tolerated, but the 1-fail-safe property was maintained by the *safe gate*.

5 Summary

A CMOS gate which is fail-safe for a given class of faults was presented. SPICE simulations indicate that the fail-safe gate can tolerate several faults. Fail-safeness is preserved for multiple faults also. Fail-safe systems can be duplicated for the correction of failures in one system [2]. Basic fail-safe circuits should be studied further for the realization of fail-safe logical hardware design.

Acknowledgement: This research was supported in part by NASA under grant NAGW-1406, and by the Idaho State Board of Education under research grant 89-041. The authors would like to express sincere gratitude to Barbara Martin for her help in the preparation of the manuscript.

References

- [1] R. Negrini, M. Sami, and R. Stefanelli, *Fault-Tolerance through Reconfiguration of VLSI and WSI Arrays*, Cambridge, MA, The MIT Press, 1989, Chap. 2
- [2] H. Mine and Y. Koga, "Basic Properties and a Construction Method for Fail-Safe Logical Systems", *IEEE Trans. on Electronic Computers*, Vol. EC-16, No. 3, pp. 282-289, June 1967
- [3] Y. Tohma, Y. Ohshima, and R. Sakai, "Realization of Fail-Safe Sequential Machines using k-out-of-n Code", *IEEE Trans. on Computers*, Vol. C-20, pp. 1270-1275, Nov. 1971
- [4] R. Chandramouli, "On Testing Stuck-Open Faults", *Proceedings of the 13th Fault Tolerant Computing Symposium*, pp. 258-265, 1983
- [5] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design*, Addison-Wesley, 1985, Chap. 4
- [6] D. Sawin, III and G. Maki, "Fail-Safe Asynchronous Sequential Machines", *IEEE Trans. on Computers*, Vol. C-2, pp. 675-677, June 1975
- [7] L. Glasser and D. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley, 1985, Chap. 5
- [8] T. Ma and P. Dressendorfer, *Ionizing Radiation Effects in MOS Devices and Circuits*, New York, NY, John Wiley and Sons, 1989, Chap. 5
- [9] C. Timoc et al., "Logical Models of Physical Failures", *Proceedings of the International Test Conf.*, pp. 546-553, 1983

Supply Current Diagnosis in VLSI

J. F. Frenzel

Electrical Engineering Department
University of Idaho
Moscow, ID 83843

P. N. Marinos

Electrical Engineering Department
Duke University
Durham, NC 27706

Abstract - This paper presents a technique based upon the power supply current signature (cd) which allows for the testing of mixed-signal systems, in situ. Through experiments with a microprocessor, the cd is shown to contain important information concerning the operational status of the system which may be easily extracted using approaches based on statistical signal detection theory. The fault-detection performance of these techniques is compared to that achieved through auto-regressive modeling of the cd.

1 Introduction

The growth of mixed-signal technology has created a great need for new methods of system testing and fault prognostication. The main objective of this research was to develop a unified test methodology, applicable to digital as well as analog systems, that would reduce fault modeling requirements, eliminate completely the need of any partitioning of hybrid systems into their respective analog and digital subsystems for purposes of testing, and simplify the test generation process. To satisfy such a broad objective, it became necessary to search, both theoretically and experimentally, for system observables carrying information about the functional status of the system, and methods for extracting such information in a manner useful for purposes of fault detection and system prognosis.

1.1 Review of Supply Current Analysis

As early as 1975 it was postulated that monitoring of the supply current could provide certain advantages in the testing of digital integrated circuits [20], [21]. Yet, supply current testing lay essentially dormant until the explosion of CMOS technology led researchers to reexamine the benefits afforded by current testing. Levi was one of the first to comment upon the characteristics of CMOS technology which make it particularly amenable to what is referred as " I_{DD} Testing" [8]. This initial treatise was continued by Malaiya and Su, culminating in procedures for applying I_{DD} testing and estimating the effects of increased integration on measurement resolution [9], [10].

Recently, several researchers have examined I_{DD} testing as a method of quantifying reliability. Hawkins et al. have reported on numerous experiments where I_{DD} measurements have forecast potential reliability problems in devices which had previously passed conventional test procedures [17], [12]. This application has prompted research dedicated to improving the accuracy of measuring I_{DD} [7], [4]. Maly et al. have proposed a built-in current sensor which provides a pass/fail flag when the current exceeds a predetermined threshold. Combined with a switching mechanism, it provides a means of removing the faulty device from operation once excessive current flow is detected [18], [16], [15].

1.2 Power Supply Current Signature Analysis

All of the research on supply current testing, to-date, has been focused on comparisons of the quiescent current to a simple threshold for purposes of fault detection. No effort has been made to examine the AC characteristics of the supply current waveform for indications of potential failures. While Dorey et al. acknowledge the potential information to be gained from a study of switching currents, they dismiss this area due to the complexities of waveform acquisition and analysis [13]. Only recently, Hashizume et al. utilized an autoregressive (AR) model of the supply current waveform for detecting faults in combinatorial logic through pattern recognition [14]. By analyzing the *entire* cd as a continuous-time signal, it is possible to develop a test methodology, applicable to both analog and digital technology, capable of fault prognostication. However, estimating the coefficients needed in the AR model of the cd is computationally burdensome.

In this paper we will develop and evaluate an efficient method for extracting information from the cd using statistical signal detection theory. Section 2 describes the simulation of microprocessor functional faults which were used to evaluate the test technique. A model for the cd is presented in Section 3, and based upon limited assumptions, a method for detecting an unknown fault component, referred to as "the likelihood ratio test", is introduced. The performance of this technique against the simulated microprocessor failures is examined and a method for system prognosis presented. In Section 4 we compare the fault detection performance of AR modeling to that achieved by the likelihood ratio test. Finally, Section 5 summarizes the results of this research and presents recommended usage of the test technique.

2 Simulation of Microprocessor Functional Faults

In this section we describe the simulation of functional failures using the Intel 8086 microprocessor. The Intel 8086 was running at 2.45 MHz on an SDK-86 development board. The power to the processor was isolated from the board supply and the current drawn by the processor sampled at a 102.4 MHz rate using an AC current probe and a digitizing oscilloscope.

Three classes of functional faults were investigated: *data storage and transfer faults*, *register decoding faults*, and *instruction decoding faults* [6]. Simulated failures were introduced by modifying either initial register contents or the instructions stored in program

memory. It should be stressed that the instructions were modified such that the number of one's in the instruction code or operand(s), referred to as the *weight*, was constant for any given byte. This is important, as the amount of current drawn by the input buffers during the instruction fetch cycle contributes greatly to the overall current. If the weight of a particular byte was altered in order to simulate a fault, then the change in current drawn by the input drivers when that byte was fetched from memory might obscure any additional current variation. Consequently, due to strict adherence to this principle, it is possible to attribute deviations in the current signature to the presence of the simulated fault, rather than the modeling technique.

2.1 Data Storage and Transfer Faults

Data storage (or transfer) faults were modeled by altering the contents of a register used by a move instruction. The reference case initialized the register DX with the operand AAAA. For the simulation of fault-1 DX was initialized with the operand 5555 which has the same weight as the operand used in the reference case. Under fault-2, DX was initialized with the value ABAA which has a weight one greater than that of AAAA. The deviations of the resulting signatures from the reference signature are shown in Figure 1.

From these experiments we might project that it will be difficult to detect data faults that do not change the weight of either the operands or the result of a particular instruction, as the differences observed under data fault-1 appear to be random. A possible explanation is that the supply current signature is actually the sum of many individual currents; if one transistor draws less current due to a change in logic state, yet an equivalent transistor in another bit position draws more current due to the opposite state change, then the two effects will cancel, leaving no discernible differences in the resulting signature.

2.2 Register Decoding Faults

Register decoding faults were modeled by altering the register field of individual instructions, thus enabling the selection of incorrect source and/or destination registers. In each case, the total weight of the register field was kept constant and all registers were initialized to the same value to prevent the introduction of any artificial effects.

Fault-1 involved modifying the instruction MOV BX,DX, encoded as 8B DA, to 8B D3 which caused the execution of MOV DX,BX. This modeled the occurrence of a register decoding fault which caused the selection of incorrect source and destination registers. Figure 2 shows the difference between the reference signature and that observed under the simulated fault.

As a second example, fault-2, we chose to model the selection of an incorrect source register. This was done by modifying the register field of the MOV instruction to D9, which caused the execution of MOV BX,CX. Because the register addresses of CX and DX have the same weight, it was expected that this fault would cause very little change in the cd. However, as shown in Figure 2, the difference between the reference signature and that obtained under the simulated fault shows a large peak at the point which corresponds

2.4.4

to the activation of the source register. From this we conclude that although CX and DX appear equivalent, since they are both general purpose registers and their register addresses are of equal weight, there are electrical variations which cause a large difference in the amount of current drawn during their use.

2.3 Instruction Decoding Faults

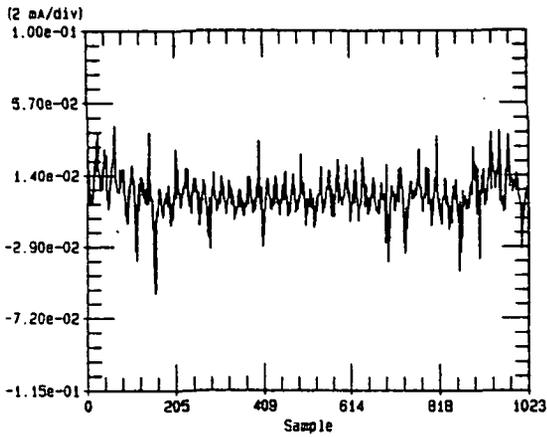
Instruction decoding faults were introduced by modifying the opcode fields of individual instructions, taking care to preserve the weight of the opcode. Two instruction decoding faults were modeled; fault-1 was introduced by changing the opcode field of the MOV instruction from 8B to 8E, which resulted in the execution of the instruction MOV DS,DX. The second fault was injected by changing the opcode field to 39 which executed the instruction CMP DX,BX. In both cases the weight of the opcode field remained consistent with that used to produce the reference signature. The differences between the reference signature and the signatures recorded under fault-1 and fault-2 are shown in Figure 3. These differences are significantly greater than the differences observed under data faults or register decoding faults. Evidently, the execution of an incorrect instruction severely impacts the current signature, allowing for simple detection of the fault. This is an important advantage of Power Supply Current Signature analysis, as control faults are typically the most difficult to detect. Control faults may give rise to the execution of spurious instructions which could contaminate register contents. Detection of such spurious instructions involves the time-consuming propagation of the processor state to observable outputs.

3 The Likelihood Ratio Test

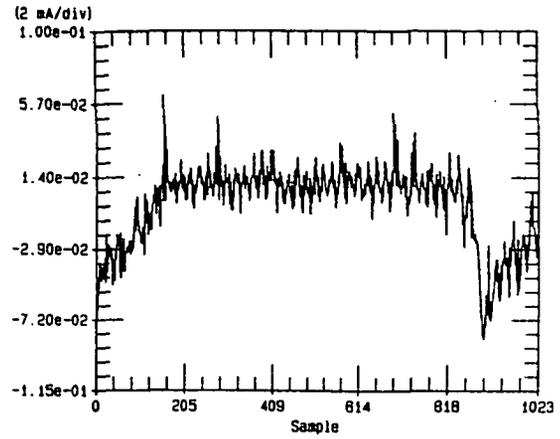
It has been demonstrated, using the results of SPICE simulations and circuits comprised of 7400-series devices, that examination of the cd may be used for purposes of fault detection [23]. A method of analysis, referred to as "transition matching", was developed and demonstrated to be extremely effective. However, when applied to more complex devices, transition matching proved to be insufficient at completely utilizing the information contained within the cd. This experience exemplified the danger associated with optimizing an analysis technique for a particular system; each system will exhibit its own characteristics and fault responses, depending upon the technology (CMOS, ECL, hybrid, ...) and level of integration (module, board, ...). If we wish to obtain a methodology which may be applied successfully across all boundaries, then we must develop such a technique without placing any restrictions upon the form of the cd or the fault effects. In this section we will present a cd model that limits these assumptions and develop a method of analysis based upon statistical signal detection.

3.1 cd Model Development

We have chosen to model the observed cd as the sum of three signal components as shown in Equation 1. The observable supply current drawn by a device under test, $z(t)$, is equal to

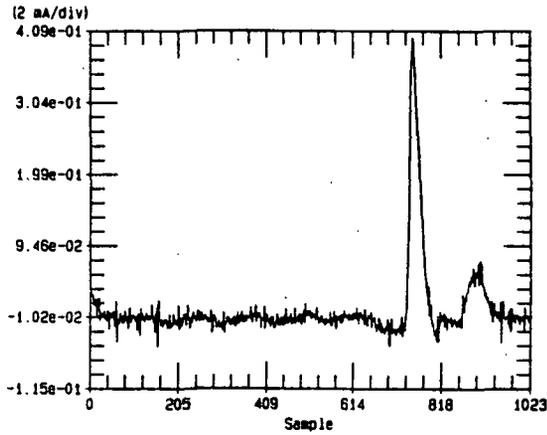


Fault-1

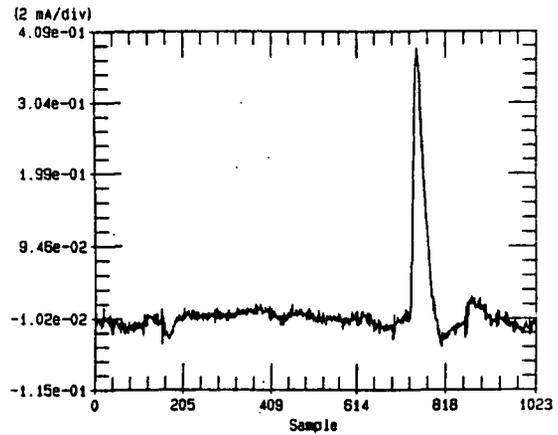


Fault-2

Figure 1: Data Fault Differences

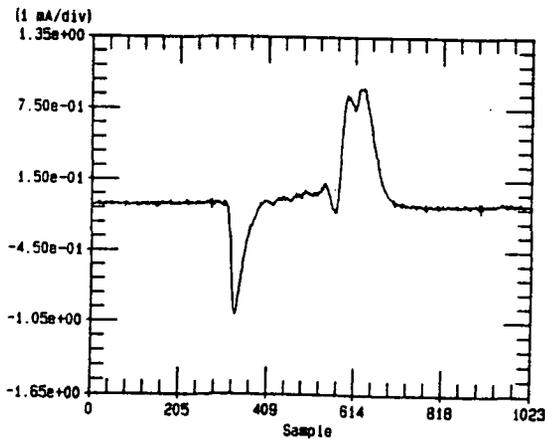


Fault-1

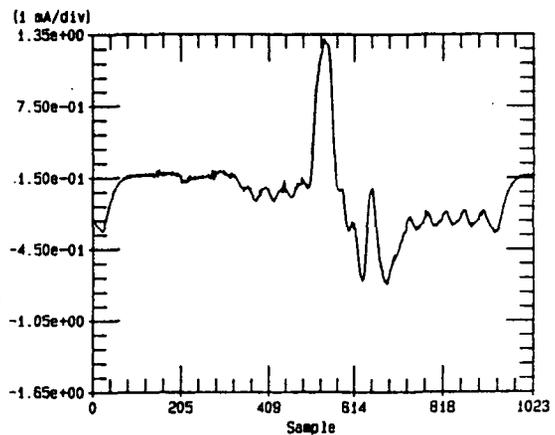


Fault-2

Figure 2: Register Fault Differences



Fault-1



Fault-2

Figure 3: Instruction Fault Differences

the sum of the current drawn by a fault-free device, $w(t)$, any additional current (positive or negative) which is drawn as a consequence of faults, $F(t)$, and random noise, $n(t)$, caused by factors such as thermal effects and sampling error. For the case of a fault-free device $F(t)$ will equal zero; conversely, when multiple faults are present $F(t)$ will be a composite signal, referred to as the "fault component", representing the cumulative effect of the individual faults. The observed current signature may be given as

$$z(t) = w(t) + n(t) + F(t) \quad (1)$$

If we are able to form an estimate of $w(t)$, either through simulation or repeated observations of a "golden device", then this estimate, $\hat{w}(t)$, may be subtracted from the observed cd, leaving

$$z(t) - \hat{w}(t) = n(t) - e(t) + F(t) \quad (2)$$

where the term $e(t)$ represents the error in the estimate, and may be ignored provided enough trials are made to form an accurate estimate of the supply current drawn by a fault-free device. Consequently, the procedure of detecting a fault reduces to that of estimating $w(t)$ and determining whether $F(t)$ is equal to zero. This is equivalent to the classical signal estimation and detection problem involving non-random, unknown signals in noise.

3.2 Maximum Likelihood Estimator and Detector

With a "golden device", Equation 1 will reduce to $z(t) = w(t) + n(t)$, and the problem of estimating $z(t)$ is that of estimating an unknown signal in noise. If the signal is deterministic and the noise has a mean of zero with a Gaussian distribution, then an appropriate estimator is the maximum likelihood estimator (MLE), which is given as

$$\hat{w}_{MLE} = \frac{1}{N} \sum_{i=1}^N z_i t \quad (3)$$

Subtracting this estimate from the observed cd and discarding the error term, leaves $z(t) - \hat{w}_{MLE}(t) = n(t) + F(t)$, and the problem of fault detection becomes equivalent to detecting an unknown signal, $F(t)$, in noise.

Again, if the signal is deterministic and the noise is Gaussian with zero mean, an appropriate detector for $F(t)$ is the likelihood ratio test detector [19], given by

$$\lambda_t = [z(n) - \hat{w}(n)]' R_n^{-1} [z(n) - \hat{w}(n)] \quad (4)$$

with $z(n)$ representing the sampled current during application of the test patterns to an untested device, $\hat{w}(n)$ is the estimate of the current drawn by a fault-free device, and R_n^{-1} is the inverse of the noise covariance matrix. Under multiple observations of the cd, the test statistic becomes

$$\lambda_t = \sum_{i=1}^N [z_i(n) - \hat{w}(n)]' R_n^{-1} \left(\frac{1}{N} \sum_{j=1}^N [z_j(n) - \hat{w}(n)] \right) \quad (5)$$

We can now state a formal description of the test procedure which we define as "the likelihood ratio test".

1. Form the inverse noise covariance matrix, R_n^{-1} , using the acquired noise statistics.
2. Estimate the current drawn by a fault-free device, employing either simulation or a "golden device" and Equation 3.
3. Apply the appropriate test patterns to the device under test and form the test statistic λ_t according to either Equation 4 or Equation 5, depending upon whether multiple observations are available.
4. Compare λ_t to an established threshold; if the threshold is exceeded then the unit under test is classified as "faulty", otherwise it is accepted as having passed the test.

We have introduced a method for analyzing the cd, based upon knowledge derived from classical signal detection theory. We chose to model a faulty device as exhibiting a cd with an additional, but unknown, fault component. This allows greater flexibility in the types of faults which may be detected, as well as the systems to which this procedure may be applied, as no restrictions or assumptions have been made of the *form* of either the cd or the fault component. We will now evaluate the fault detection performance of the likelihood ratio test against the microprocessor faults described in Section 2.

3.3 Performance Evaluation of the Likelihood Ratio Test

It is possible to quantify the separation of the density function obtained under a simulated fault, hypothesis H_1 , from the density function obtained from a fault-free system, hypothesis H_0 , using a detectability index given as

$$d^2 = \frac{(\mu_{H_1} - \mu_{H_0})^2}{\sqrt{\sigma_{H_1}^2 \sigma_{H_0}^2}} \quad (6)$$

where μ and σ represent the parameters of the test statistic density functions. A second method of assessing fault detection capability is to use the empirical probabilities of fault detection (P_D) and false alarm (P_F), where false alarm implies the classification of a fault-free system as faulty. These probabilities are calculated by tallying the instances of correct and incorrect system classification under H_1 and H_0 , respectively. Both of these methods will be used to compare the performance of the likelihood ratio test against the simulated faults under different processing environments.

As was noted in Section 2, the data faults had the smallest impact upon the supply current, while the instruction decoding faults had the largest. This observation would indicate that the more circuitry affected by the fault during execution of the test program, the greater the alteration of the supply current signature, and is supported by the indices of detectability shown in Table 1. It is important to note that it was predicted that fault-1 from the class of data storage and transfer faults would prove difficult to detect using this method, and based upon the results in Table 1, this appears to be correct. However, for all remaining faults the test algorithm yielded *complete* fault detection with no false alarms.

Table 1: Algorithm Performance versus Faults

Signature	d	μ	σ
Data Fault-1	0.07	1848	576
Data Fault-2	12.27	6115	338
Register Fault-1	80.37	1.8×10^5	2.5×10^3
Register Fault-2	57.24	1.9×10^5	5.5×10^3
Instruction Fault-1	1204.40	1.6×10^6	5.4×10^3
Instruction Fault-2	424.24	3.9×10^5	2.6×10^3

Table 2: Detectability under Subsampling

Detectability Index d for Data Fault-2					
Number of Points					
31	62	125	250	500	1000
6.71	8.56	10.58	11.41	11.98	12.27

3.3.1 Data Reduction

The utility of any test method is dependent upon the amount of resources required for implementation. Consequently, an effort was made to evaluate the effects of subsampling upon the fault detection performance.

Subsampling was accomplished by discarding evenly spaced samples from the original data. The effect of subsampling upon the detectability index is shown in Table 2 for data fault-2. As might be expected, the fewer the number of points used to make a decision, the lower the detectability index. However, if we examine the case which yielded the lowest detectability index, based upon 31 data points, we find that there was still a significant amount of separation under the two hypotheses. Assuming equal *a priori* probabilities, if one operates the detector at the threshold which yielded the minimum probability of error, then the probability of detection was 0.98 and the probability of false alarm was zero, based upon histograms of 50 reference signatures and 50 signatures under data fault-2.

3.4 Use of the Likelihood Ratio Test in System Prognosis

The previous sections chronicled the effectiveness of cd analysis for the purpose of detecting system faults. However, the increasing use of electrical systems in "critical mission" applications has created an urgent need for test methods capable of exposing *potential* faults *prior* to actual system failure. Supply current analysis in general, and the likelihood ratio test, in particular, possess several unique attributes which provide for the capability of system prognosis. This section explores the relationship between system failures, their effect upon the supply current, and the potential for system prognosis.

A fault may be defined as the alteration, in electrical behavior, of a circuit component

or signal path. "Hard" failures, such as open and short circuits, may be caused by metal migration, poor bonding, and insulator breakdown, resulting in an alteration of circuit connectivity. "Soft" failures, such as a change in component value or switching speed, may not *immediately* lead to an operational failure, yet over time may deteriorate into a hard fault which does affect the functionality of the system. However, both types of faults cause a change in the electrical current drawn by the affected subnetwork *as a function of time*. Depending upon the amount of circuitry identified with the fault, this deviation may be reflected in the power supply current signature. It is these two attributes, sensitivity to system changes and immediate observation of system behavior, that allow for system prognosis under cd analysis. Because cd analysis removes the requirement for propagating system status to observable outputs, fault prognostication may be accomplished *prior* to experiencing functional failures.

The likelihood ratio test is particularly appropriate for prognostication as it allows for the statistical comparison of present behavior, captured in the test statistic, to past behavior, represented by the distribution of the test statistic under fault-free conditions ($p(\lambda|H_0)$). For purposes of system prognosis, it is possible to quantify the deviation of the present behavior from historical observations by calculating the integral

$$\int_{\lambda_t}^{\infty} p(\lambda|H_0) d\lambda \quad (7)$$

Generally $p(\lambda|H_0)$ will be represented by a histogram; thus, the integral may be calculated by tallying the number of observations for which the calculated test statistic exceeded the present test statistic and normalizing. If the result is greater than one half, then the agreement between the present cd and the reference signature is better than was normally observed. However, as this number approaches zero, indicating a strong deviation from previous system behavior, the probability of falsely classifying the system as faulty approaches zero, and the system should be taken off-line for extensive testing and examination, even if no malfunctions have been detected.

4 Autoregressive Modeling of the Supply Current Signature

It has been suggested that fault detection in digital devices might be realized through autoregressive modeling of the supply current waveform [14]. In this section we will review the theory behind autoregressive (AR) modeling and apply the technique against the simulated faults described in Section 2. Finally, we will compare the effectiveness of AR modeling against the performance of the likelihood ratio test as detailed in Section 3.3.

4.1 The Theory of Autoregressive Modeling

Autoregressive (AR) modeling is an area of time series analysis in which the time series in question is assumed to be the output of a linear system according to the following equation

$$s_n = - \sum_{k=1}^p a_k s_{n-k} + G u_n \quad (8)$$

where G and a_k , $1 \leq k \leq p$, are the parameters of the system, u_n is the present input, and s_n is the present output. This approach has proven useful in exposing the underlying structure of many complicated systems, ranging from the human vocal tract to wind turbulence [3]. In this particular case we intend to explore the use of the AR coefficients, a_k , as a means of compressing the information contained in the supply current signature.

Often the input signal, u_n , is unknown and it is necessary to estimate the present output as a linearly weighted summation of the past outputs

$$\tilde{s}_n = - \sum_{k=1}^p a_k s_{n-k} \quad (9)$$

The error of the estimate, \tilde{s}_n , is given by

$$e_n = s_n - \tilde{s}_n = s_n + \sum_{k=1}^p a_k s_{n-k} \quad (10)$$

and is typically referred to as the residual.

4.1.1 Determining the Model Order

The first step in AR modeling is the determination of the appropriate model order. Two methods are commonly used to arrive at a selection: computation of the residual variance, and analysis of the partial autocorrelation function (PACF) [1]. The former is a straightforward procedure; AR models of increasing order are successively applied against the time series under study until the variance of the resulting residuals reaches a satisfactory threshold.

An alternative method is based upon study of the partial autocorrelation function. The PACF is a plot of the correlation between observations at increasing lags, with the effects of the intervening observations removed. It has been shown, that for an AR process of order p , the PACF will cut off after lag p , where cut off implies that the function truncates abruptly with the remaining values less than twice the standard error of the coefficient estimate [2]. As a result, it is possible, through evaluation of the estimated PACF, to determine the appropriate model orders to select for experimentation.

4.1.2 Reducing Nonstationarity through Differencing

While Equation 8 is effective at modeling a wide class of times series, there are many signals which exhibit some degree of nonstationarity, indicated by a slowly decaying ACF [2].

In order to effectively model these waveforms it is necessary to first reduce the effect of the nonstationarity. This may be accomplished through suitable first-order differencing. A time series which is nonstationary in the mean may be transformed into a stationary process through the application of a single difference operator, whereas a series which is nonstationary in both the mean and the slope will require that the operation be performed twice [2].

4.1.3 Refining the System Model

Rare is the case where the scientist is presented with data which calls for a specific model order. More often, time series analysis is an iterative process, involving many attempts at improving the model performance through the selection of difference operators and model order. An initial model is formed based upon the information presented in the ACF and PACF. After this, it is necessary to analyze the residuals, using the ACF, for any remaining process structure which has not been included in the model. The model is then updated to reflect this additional information and the process repeated until the residuals resemble those of a random process.

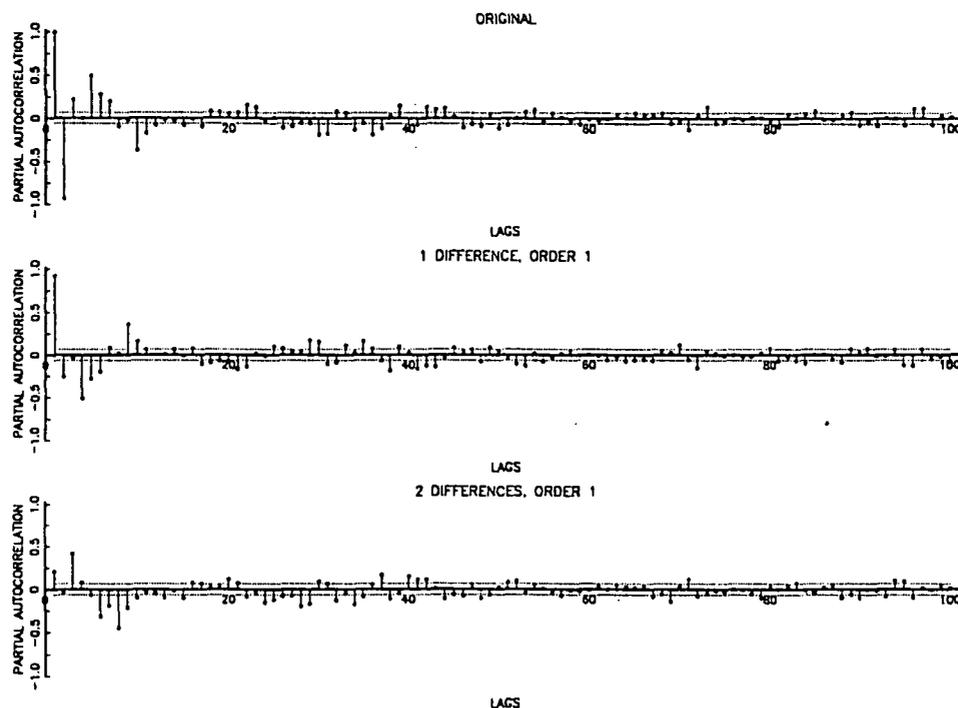


Figure 4: PACF of Original and Differenced Data Reference Signatures

4.2 Application of AR Modeling to Microprocessor Faults

We began our investigation with the cd observed during application of the data storage and transfer test program. Figure 4 shows the PACF for the original time series, as well as the first and second order differenced time series. Bearing in mind that, for an AR process

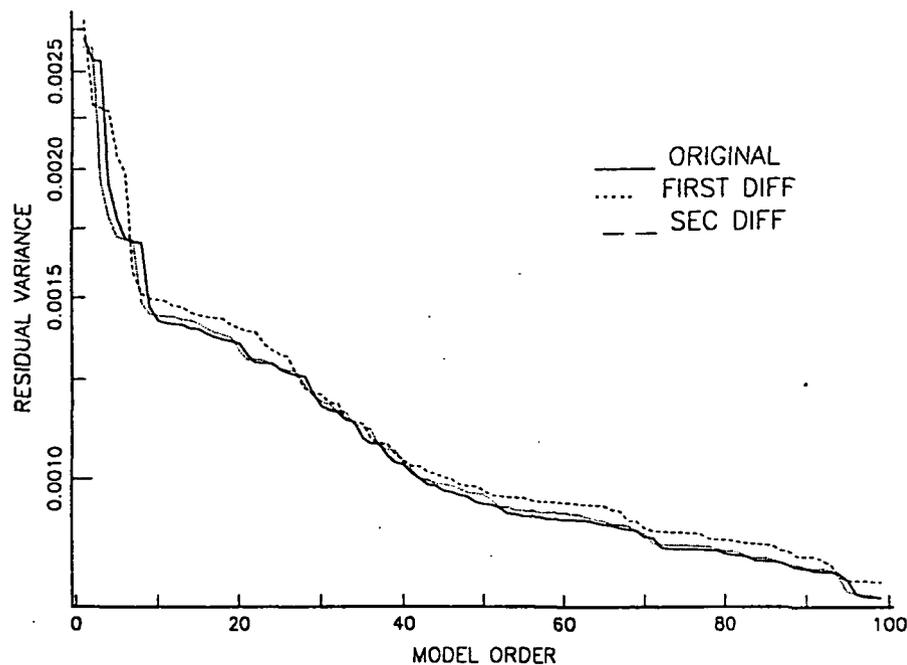


Figure 5: Residual Variance versus Model Order

of order k , the PACF will cut off after k lags, it appears that there is no clear choice of a particular model order for any of the time series. However, we can deduce that the model order must include at least ten terms. This deduction is supported by the information in Figure 5, which is a logarithmic plot of the residual variance versus model order for each of the time series. Based on these observations we selected two AR models for exploration with these time series, one of order 12 and one of order 100.

Figure 6 shows the ACF of the residuals obtained when modeling each of the time series using 12 terms. There are a significant number of coefficients which are greater than twice the standard error of the estimate, the most prominent of which occurs at lag 40. This is supported by Figure 5, where the slope of the plot seems to change slightly in that vicinity. There is also a large coefficient at lag 94. Figure 7 shows the ACF of the residuals obtained using 100 terms and we see that there are no coefficients greater than the margin of error for lags less than 100. From this we would conclude that there is no structure remaining in the process which needs to be incorporated into the model.

4.2.1 Performance Evaluation of AR Modeling

We now turn to the appraisal of autoregressive modeling using the performance metrics introduced in Section 3.3. As the test statistic, we will use the Euclidean distance between a vector containing the AR coefficients of the reference cd and a vector formed from the coefficients corresponding to the signature of the device under test (DUT). This distance

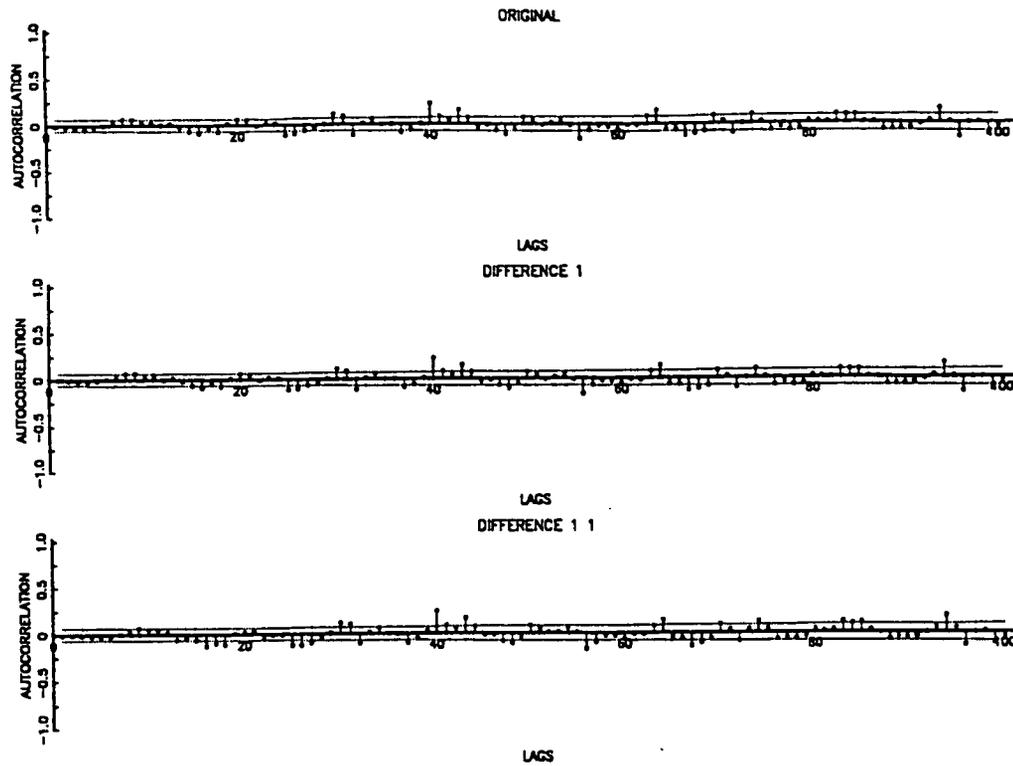


Figure 6: ACF of the Residuals for Model Order of 12

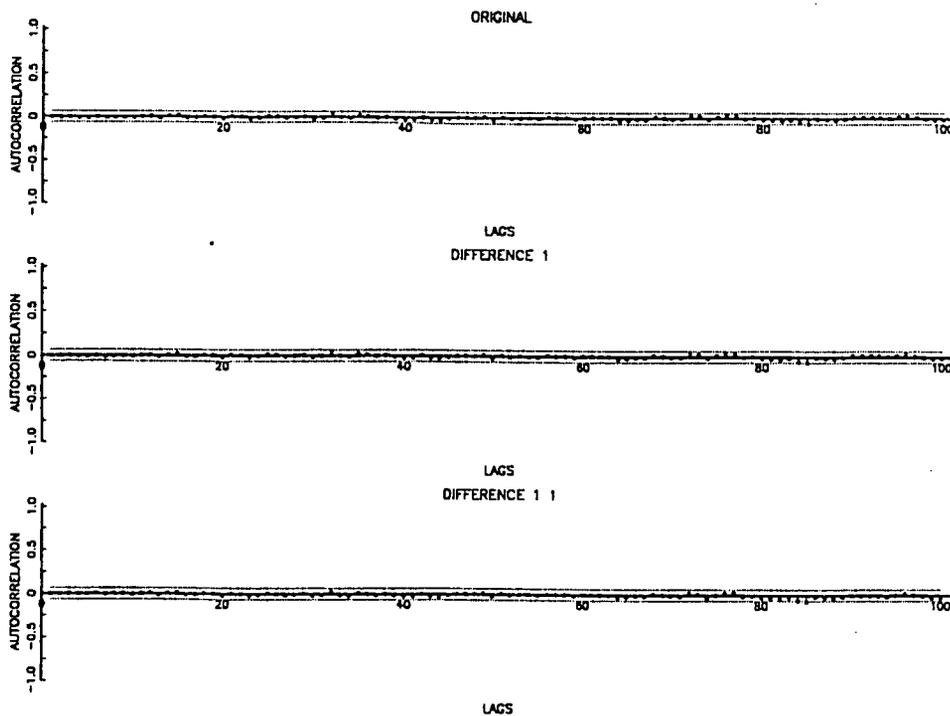


Figure 7: ACF of the Residuals for Model Order of 100

Table 3: Performance of AR modeling against Data Fault-2

Data Fault-2					
Time Series	Order	d	P_D	P_F	MPE
original	12	1.22	0.94	0.46	0.26
	100	1.44	0.72	0.18	0.23
difference (1)	12	1.14	0.86	0.38	0.26
	100	1.24	0.86	0.36	0.25
difference (1 1)	12	0.58	0.86	0.48	0.31
	100	0.44	0.84	0.44	0.30

D_E , in contrast to the λ_t used earlier, is given by

$$D_E = \sum_{k=1}^M (a_{Rk} - a_{Tk})^2, \quad (11)$$

where a_R and a_T correspond to the AR coefficients of the reference signature and the signature from the DUT, respectively.

Table 3 lists the results obtained with each of the time series, using AR models with 12 and 100 terms, against data fault-2. From this we can draw several conclusions: first, the application of the difference operator consistently resulted in poorer performance; second, based upon the minimum probability of error (MPE), the model with 100 terms yielded superior results against the model with 12 terms, although this effect diminished with each application of the difference operator.

Compared to the perfect fault detection demonstrated by the likelihood ratio test in Section 3.3, autoregressive modeling would appear to be a poor candidate for modeling the supply current signature in devices of this complexity. We elected to perform the comparison using data storage fault-2, which affected the cd to a lesser degree than either register or instruction decoding faults. AR modeling is normally used to characterize the spectral density of a process in a general sense, and is insensitive to minor variations. Consequently, for the sake of completeness we applied AR modeling against fault-1 of the register decoding and instruction decoding fault classes.

Table 4 shows the results of applying an AR model with 100 terms against the original supply current signatures obtained under fault-1 of the register and instruction fault classes and compares the performance to that achieved against data fault-2. Contrary to expectation, the performance of AR modeling was poorer against the register and instruction faults than the data fault, even though their effect upon the cd is much greater. One explanation for this phenomenon is that the *magnitude* of the variations is not the dominant factor in AR modeling, as it was in the likelihood ratio test, but rather the *shape* of the supply current deviations. In autoregressive modeling, the transfer function must be represented as an all-pole model, as the signal output is based only upon its previous values. This imposes restrictions upon the types of waveforms which may be accurately modeled. Although instruction fault-1 caused a supply current variation that is roughly

Table 4: Performance of AR modeling against Each Fault Class

Model Order 100				
Time Series	d	P_D	P_F	MPE
Data Fault-2	1.44	0.72	0.18	0.23
Register Fault-1	0.57	0.50	0.26	0.38
Instruction Fault-1	0.25	0.94	0.76	0.41

eight times greater than that experienced under data fault-2, its effect upon the AR coefficients was less, rendering it more difficult to detect. A contributing factor may be the effect of the noise upon the AR coefficients. It was shown in Section 3.3 that the fault detection performance could be greatly enhanced by incorporating the noise covariance matrix into the test statistic. A disadvantage of AR modeling is that there is no method for noise compensation.

4.2.2 Use of the Residual Variance

It has been reported that the residual variance may be used, in conjunction with the AR coefficients, to improve the performance of AR modeling [14], [22]. To evaluate the effectiveness of this technique when applied to cd analysis, we have repeated certain experiments using the residual variance and the AR coefficients to form the comparison vector. It was found that while the use of the residual variance consistently increased the distance between the reference signature and those of simulated faults, the contribution was minor. The maximum increase was on the order of 10^{-3} , with most of the values being on the order of 10^{-8} . Consequently, we conclude that use of the residual variance contributes very little to AR modeling in this particular application.

4.3 Summary

Based upon the experiments reported in this section, one must conclude that autoregressive modeling is not an effective technique for characterizing the information contained within the cd of a device as complex as a microprocessor. The effect of faults upon the signature is too small to be reflected in the AR coefficients to an extent that would exceed the normal deviations due to noise. However, for systems in which failures cause a drastic alteration of the cd spectrum, it is possible that AR modeling might prove useful in the area of fault diagnosis, as the system observables would be captured in a vector, rather than a single test statistic, allowing for the use of a fault dictionary.

5 Conclusions

In this paper we have presented a method of testing, referred to as *Power Supply Current Signature (cd) Analysis*, and demonstrated its potential for purposes of fault detection

using examples of failures in a general purpose microprocessor. A model for the experimental cd was introduced and used to develop a method of signature analysis based upon statistical signal estimation and detection, referred to as the *likelihood ratio test*. The performance of this technique was shown to be excellent at detecting all decoding faults and most data faults, and a methodology for system prognosis using the likelihood ratio was introduced. Finally, performance comparisons between the likelihood ratio test and autoregressive modeling of the cd were presented.

There are two applications for which cd analysis may be an attractive alternative to conventional testing, the first being the production testing of cost-sensitive products. Once a mature manufacturing process has been installed, cd analysis could be used in place of expensive and time-intensive testers. This would apply to both modules and boards, for cd analysis provides for the testing of mounted modules *in situ*, eliminating the need for board partitioning and module isolation. The second application involves the field-test of critical systems. Because cd analysis does not require any external observation points, systems may be tested on-line, with the application which is assigned to that board or subsystem serving as the test patterns during normal operation. Periodically, signatures could be captured and compared, using the likelihood ratio test, to those observed previously. This procedure would allow for on-line monitoring and prognostication, or failure prediction, of critical systems. In such an application, once environmental effects such as temperature fluctuation had been eliminated, any detectable cd perturbations could be directly attributed to a change in the system behavior.

Areas of future research include methods for improving the accuracy of the supply current measurement and refinement of the cd model. The concept of a built-in current sensor as proposed by Maly et al. would provide several advantages. Specifically, conversion of supply current to a voltage for off-chip measurement should provide greater immunity to system noise, thus increasing the signal to noise ratio of the cd. Furthermore, it allows for the partitioning of a VLSI module into smaller sections, providing greater distinguishability than would otherwise be possible. This concept of partitioning would involve designing for testability for cd analysis, and could be applied with similar expectations at the board level. Finally, an on-chip current sensor would provide for the implementation of the likelihood ratio test as a built-in test function.

In this paper we chose to model the cd as an unknown but nonrandom signal which could be estimated through observation of a "golden device". By subtracting this estimate from the cd of the DUT, the problem of detecting a fault reduced to detection of the unknown and nonrandom fault component. The advantage afforded by such a decision was widespread applicability across all levels of integration. However, one could chose to model the fault component as a random signal, with several uncertain parameters, such as phase and amplitude. Another possibility would be to model the cd as one of M possible signals. The task of fault detection would then be to determine which of the M possible signatures the cd of the DUT best resembled. However, this limits the number of detectable faults, or perhaps fault classes, to $(M - 1)$. A more practical solution might involve modeling minor variations in the cd as uncertainties in the noise statistics.

In closing, we have presented the development of a statistical approach to fault detection

and system prognosis which has demonstrated potential at detecting faults in complex digital devices. Furthermore, no restrictions have been placed upon the nature of the system or the possible faults, other than the requirement of access to the supply current for observations. As a result of these precautions, it is hoped that this technique will prove useful in the testing of hybrid, mixed-signal systems.

References

- [1] Gwilym M. Jenkins and Donald G. Watts, *Spectral Analysis and its Applications*, Holden-Day, 1968.
- [2] Douglas C. Montgomery and Lynwood A. Johnson, *Forecasting and Time Series Analysis*, McGraw-Hill, 1976.
- [3] John Makhoul, "Linear Prediction: A Tutorial Review," *Proceedings of the IEEE*, Volume 63, Number 4, April, 1975.
- [4] Charles Crapuchettes, "Testing CMOS I_{DD} on Large Devices," *Proceedings of the International Test Conference*, pp 310-315, 1987.
- [5] J.A. Starzyk and J.W. Bandler, "Nodal Approach to Multiple-fault Location in Analog Circuits," *IEEE Transactions on Circuits and Systems*, pp 1136-1139, 1982.
- [6] James F. Frenzel and Peter N. Marinos, "Functional Testing of Microprocessors in a User Environment," *Proceedings of the Fourteenth International Conference on Fault-tolerant Computing*, pp 219-224, 1984.
- [7] Mike Keating and Dennis Meyer, "A New Approach to Dynamic I_{DD} Testing," *Proceedings of the International Test Conference*, pp 316-321, 1987.
- [8] Mark W. Levi, "CMOS is Most Testable," *Proceedings of the International Test Conference*, pp 217-220, 1981.
- [9] Yashwant K. Malaiya and Stephen Y. H. Su, "A New Fault Model and Testing Technique for CMOS Devices," *Proceedings of the International Test Conference*, pp 25-34, 1982.
- [10] Yashwant K. Malaiya, "Testing Stuck-On Faults in CMOS Integrated Circuits," *Proceedings of the International Conference on Computer-Aided Design*, pp 248-250, 1984.
- [11] Luther K. Horning and others, "Measurements of Quiescent Power Supply Current for CMOS ICs in Production Testing," *Proceedings of the International Test Conference*, pp 300-309, 1987.
- [12] J. M. Soden and C. F. Hawkins, "Test Considerations for Gate Oxide Shorts in CMOS ICs," *IEEE Design and Test*, August, pp 56-64, 1986.

- [13] A. P. Dorey and others, "Reliability Testing by Precise Electrical Measurements," *Proceedings of the International Test Conference*, pp 369–373, 1988.
- [14] Masaki Hashizume and others, "Fault Detection of Combinational Circuits Based on Supply Current," *Proceedings of the International Test Conference*, pp 374–380, 1988.
- [15] L. Richard Carley and Wojciech Maly, "A Circuit Breaker for Redundant IC Systems," *Proceedings of the Custom Integrated Circuits Conference*, pp 27.6.1–27.6.6, 1988.
- [16] Wojciech Maly and Phil Nigh, "Built-In Current Testing — Feasibility Study," *Proceedings of the International Conference on Computer-Aided Design*, pp 340–343, 1988.
- [17] Luther K. Horning and others, "Measurements of Quiescent Power Supply Current for CMOS ICs in Production Testing," *Proceedings of the International Test Conference*, pp 300–309, 1987.
- [18] Derek B. I. Feltham and others, "Current Sensing for Built-In Testing of CMOS Circuits," *Proceedings of the International Conference on Computer Design*, pp 454–457, 1988.
- [19] James F. Frenzel, "Power Supply Current Signature Analysis: A Tool and a Methodology for use in Fault-Detection and System Prognosis," PhD thesis, Duke University, 1989.
- [20] George F. Nelson and William F. Boggs, "Parametric Tests Meet the Challenge of High-Density ICs," *Electronics*, pp 108-111, December, 1975.
- [21] A. J. Melia, "Supply-Current Analysis (SCAN) as a Screen for Bipolar Integrated Circuits," *Electronics Letters*, 1978, volume 14, number 14, pp 434–436.
- [22] Masaki Hashizume and others, "Estimating the Level of Anesthesia by EEG Analysis," *Systems and Computers in Japan*, 1985, volume 16, number 1.
- [23] James F. Frenzel and Peter N. Marinos, "Power Supply Current Signature (cd) Analysis: A New Approach to System Testing," *Proceedings of the International Test Conference*, pp 125–125, 1987.

Session 3
Signal Processing

Chairman: Joseph Feeley

A VLSI Implementation for Synthetic Aperture Radar Image Processing

A. Premkumar and J. Purviance
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - A simple physical model for the Synthetic Aperture Radar (SAR) is presented. This model explains the one dimensional and two dimensional nature of the received SAR signal in the range and azimuth directions. A time domain correlator, its algorithm and features are explained. The correlator is ideally suited for VLSI implementation. A real time SAR architecture using these correlators is proposed. In the proposed architecture, the received SAR data is processed using one dimensional correlators for determining the range while two dimensional correlators are used to determine the azimuth of a target. The architecture uses only three different types of custom VLSI chips and a small amount of memory.

1 Introduction

Modern real time signal and image processing demands high speed computing hardware and large memory. The availability of low cost, high density, high speed, very large scale integrated circuits enables the design of massively parallel processors which can exceed supercomputers in calculation speeds. This makes special purpose parallel processors attractive for real time signal processing.

Synthetic Aperture Radar (SAR) is an imaging system dominated by signal processing algorithms. Signal processing algorithms use techniques like convolution, correlation and filtering. The major processor requirement in these techniques is the need to multiply a data value by a coefficient and add it to a sum. Hence, digital processors implementing these signal processing functions are specialized arithmetic computers optimized to do multiplications, additions and logical operations. Most of the signal processing algorithms are highly repetitive and possess properties such as regularity, recursiveness, etc. These properties make the signal processing architecture less complex to build [1,2].

In this paper, a physical model for SAR is presented. An architecture for the SAR processor is also presented. This architecture for SAR uses a new correlator chip. This paper also describes the correlator, its application to two dimensional imaging and the features of the SAR architecture.

2 SAR Principles

Synthetic Aperture Radar is an imaging system in which a return pulse from a object target is detected and processed using convolution techniques. SAR differs from normal aperture radar in that the beam aperture in the direction of motion of the radar (Azimuth direction) is synthetically increased by processing carrier phase history (Doppler phase history) information in the returned radar signal. In this way, a sharply focussed beam is synthesized from a normal length antenna and a high azimuth resolution is achieved. In a SAR, the area to be mapped (Swath) is in a direction perpendicular to the azimuth direction (Range direction). Processing the received data in a SAR is a 2-D operation. However, with suitable corrections and compensations, this 2-D operation can be considered as two 1-D operations. A simple physical model explaining the principle on which this separation is done is explained in the following subsection.

2.1 A Physical Model for SAR

First, it is helpful to visualize a two dimensional array of all the data collected from the target during the time the target is under the influence of the beam (dwell time). The signal processing that will be developed [3] is easily understood by referring to this data array. This array never really exists as a whole, but is used here as a conceptual aid in the presentation that follows.

The radar signal is a swept linear FM pulse of a fixed duration and the pulses are repeated at the Pulse Repetition Frequency, PRF . Between the pulses, the transmitter is turned off and the antenna and electronics become a receiver system. The reflected signal from a single pulse is collected, digitized and stored as a range line in the memory. The length of the range line is dependent on the width of the area to be imaged.

Let a be the number of samples in each range line. After several range lines, say b , have been received, the signal memory array is formed and the array contains $a \times b$ data points. The coordinates of the signal memory are the slant range and the azimuth. A single sample in the signal memory array contains contributions from a large number of point targets (ground reflectors) in the area to be mapped. By the above principle, the received signal from a single point reflector on the ground will be spread through a large area in the signal memory array. Let p and q be the number of samples along the range direction and azimuth directions over which this spread in the signal memory array takes place. It is valid to assume that the energy contribution from a point target beyond this area is negligible.

Assuming for a moment, that the transmitted pulses are continuous in nature, the received data will exhibit a carrier phase history proportional to the slant range rate. It is this carrier phase history on which the SAR principle is based. An analysis of the satellite and point target motion shows that the slant range rate varies, approximately, linearly with time. This means that the received carrier phase history varies linearly with time or has a linear FM modulation. As the radar transmitted signal is pulsed and itself is a swept linear FM waveform, this carrier phase history is manifested as a changing phase angle in

the received carrier signal. There is a large disparity in range and the azimuth FM rates in the received signal and hence, can be decoupled with correction. Thus the point target response in the signal memory array consists of two linear FM waveforms, one in the range and the other in the azimuth directions.

The function of the SAR signal processor is to focus or compress the energy of each point target which is spread in the signal memory array in the manner described above, into single points in the image. In actuality, a two dimensional compression using a two dimensional reference function is needed to compress the energy into a point target, since the spread is in two dimensions. A two dimensional correlator of $a \times b$ multipliers will image the point targets comprising the swath.

A closer look at the received data reveals that the data collected from point targets from a single transmitted pulse are all the same except that the data from each point target may have a different amplitude and time shift. This time shift is the delay associated with the received signals from point targets at different locations. Since, only p samples contribute to the energy for a point target in the range direction, it is enough if only p multipliers are used in the range direction. It is also sufficient for the range reference function to use only p samples. A similar argument is true in the azimuth direction. The azimuth phase history is similar for all the point targets at different azimuth coordinates, but shifted in time. Hence, only q multipliers are needed in this direction, since the spread in the energy of a point target extends only over q samples in the memory. Figure 1 shows the total image data in the signal memory array and the two dimensional correlator with p and q multipliers in the range and azimuth directions respectively. The other point targets can be imaged by sliding this $p \times q$ correlator along and across the range direction.

The effect of the correlation operation is to slide the two dimensional correlator along the data and whenever a two dimensional match is obtained with the two dimensional reference function, a point is imaged. This, in effect, reduces the number of multipliers to be used in the two dimensional correlator. If no range migration (RM) is assumed, then the two dimensional operation along the range and the azimuth directions can be performed independently, since there is no cross coupling between the data in the two directions. The range compressed data lies parallel to the azimuth axis. Azimuth compression of the range processed data with an azimuth reference function results in the point target being imaged. Because of the forward motion of the radar platform and the Earth's rotation, the range of a point target response, however, varies with each transmitted FM pulse. This is because the instantaneous slant range from the satellite radar antenna to the point target on the ground varies appreciably during the total period over which the pulses are transmitted. Hence, the spread in the energy due to a single point target is nonlinear and is in two dimensions. So, when the data is compressed in the range direction, the path taken by the compressed data in the signal memory array is two dimensional in nature. Hence, if a one dimensional compression in the azimuth direction is to be done, then a correction has to be applied to align the data parallel to the azimuth axis. However, if a two dimensional azimuth compression is done, a two dimensional azimuth reference function is needed. This 2-D azimuth reference function should also take into account the satellite parameters such as pitch, yaw and roll.

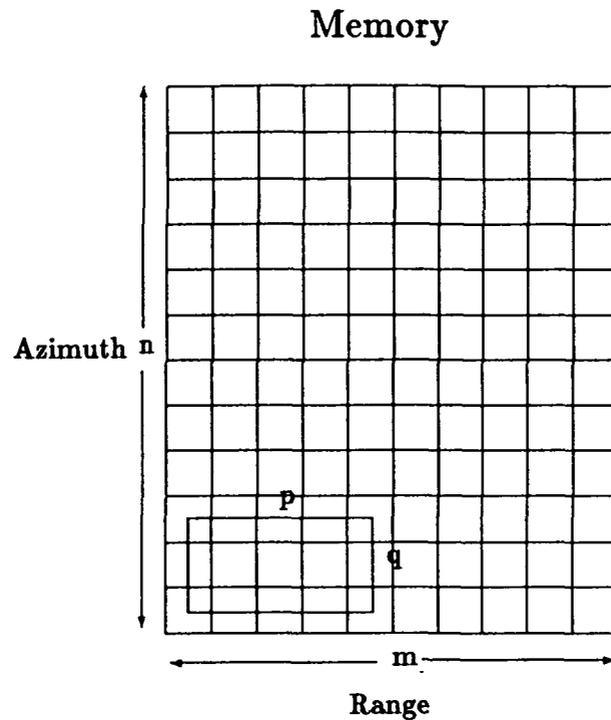


Figure 1: Total Image Data collected and a Two Dimensional Correlator with $p \times q$ Multipliers

3 Correlator

In signal processing, discrete time sequences and systems are extensively encountered. A common practice is to process discrete time sequences using discrete time systems. The discrete time system produces an output given by the convolution operation as:

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) \quad (1)$$

where $x(k)$ and $h(k)$ are the input sequence and the impulse response of the system respectively.

The preceding 1-D signal processing operation can be extended to 2-D applications. In image processing, since the input data is inherently 2-D, the convolution and correlation operation will also be 2-D. The 2-D correlation expression is as follows:

$$y(n_1, n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} a^*(k_1, k_2)b(n+k_1, n+k_2) \quad (2)$$

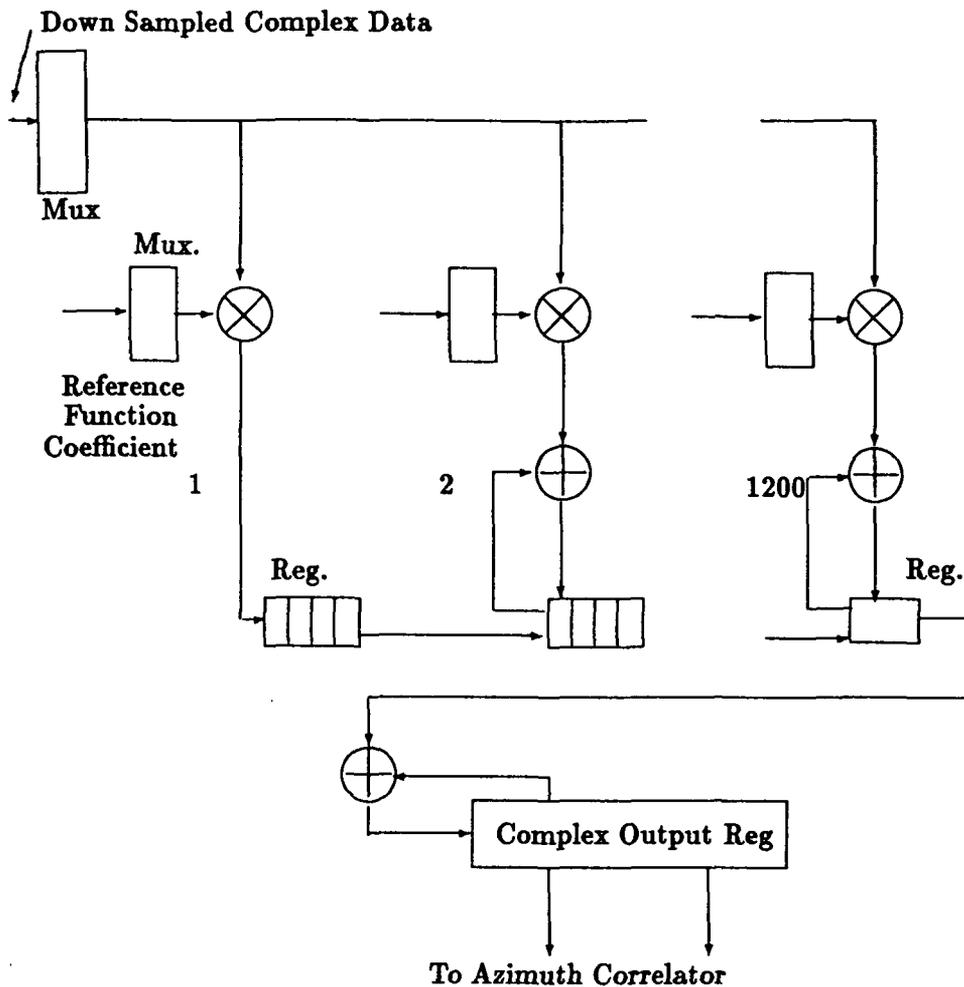


Figure 2: Block Diagram of the Correlator Circuit

3.1 Correlator Architecture

The one dimensional correlator architecture consists of a series of multipliers operating in parallel as shown in Figure 2. The multiplicand and the multiplier are in general both complex sequences. The multiplier operates at four times the speed of the data rate, since one multiplier is used to do the complex multiplication. The one dimensional correlation algorithm is implemented according to Equation 1.

The circuit shown above can be easily extended to the two dimensional correlation operation. For two dimensional correlation, the input data is an $n \times m$ array of complex numbers. Theoretically, the number of multipliers needed is equal to the number of elements in the array, i.e., $n \times m$ multipliers. However, a smaller number of multipliers can be used and correlation can be achieved if the reference function has many zero sample values. Then, the number of multipliers needed is equal to the number of non-zero samples in the reference function.

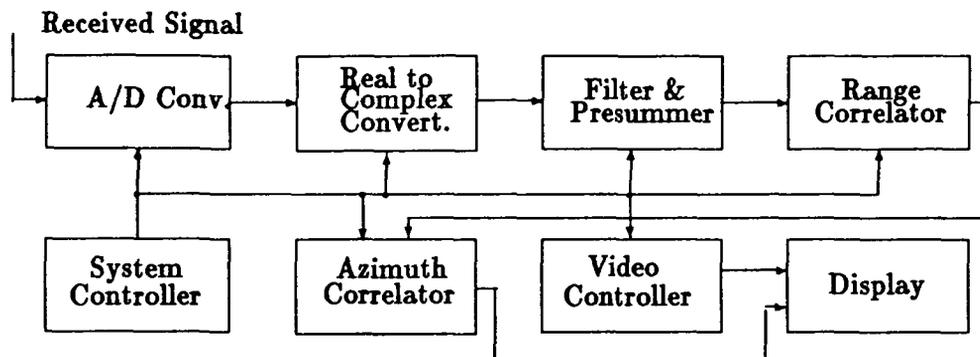


Figure 3: Block Diagram of SAR Processing Architecture

3.2 Significant Features of the Correlator

1. The architecture performs the two dimensional correlation with one data clock period latency.
2. The architecture is ideally suited for two dimensional correlation especially when there are a large number of zero coefficients in one of the functions being correlated.
3. Correlation output appears every data clock period.
4. A CMOS VLSI chip containing 64 multipliers configured as a correlator is possible.

4 A Real Time SAR Architecture

Processing SAR images involves compression techniques using correlation operations. The similarity in both range and azimuth processing enables similar hardware to be used for both operations. Using the correlator architecture described above, it is possible to implement the correlation operation with VLSI chips in the time domain. Hence, a time domain implementation is discussed in the following sections. This example is based on the SEASAT system parameters [4]. A block diagram of the time domain architecture is shown in Figure 3. The data received from the target area is sampled and converted to digital form by an A to D converter. The digitized data is converted to real and imaginary form to extract both the reflectivity and the phase information. The following sections briefly describe the block diagram.

4.1 Filter and Presummer

The azimuth time-bandwidth product required for the desired resolution is much lower than what is available in the received signal. In order to reduce the memory requirements

of the system and the processing data rates, the received data is filtered in the azimuth direction and then down sampled.

4.2 Range Correlator

Range correlation compresses the image in the range direction. The one dimensional range correlator uses the filtered and down sampled data and a reference function with 1200 samples. Each processor chip has 64 multiplier cells. Each multiplier cell has its own complex multiplier operating at 16MHz, an adder and a 4 word register. With 1200 samples in the reference function, 20 such chips must be connected in parallel to perform real time correlation. The reference function samples are stored in buffers.

4.3 Azimuth Correlator

The azimuth correlation is performed on the range compressed data. The reference function is two dimensional in nature for the reasons explained in Section 2. The reference function depends on various parameters of the spacecraft and hence, has to be computed. 64 samples are usually needed in the reference function. Hence, a two dimensional correlator consisting of 64 by 64 multipliers is needed to compress the range correlated data. However, in the array of multipliers, all the coefficients except 64 are zero. Hence, only 64 multipliers are needed in the azimuth processor. The multipliers are loaded with the samples corresponding to a reference function that follows the range migration path of the target under consideration. The partial products obtained after every multiplication are stored in the memory according to the position of the 64 non-zero coefficients. The storage operation is performed by the video controller in conjunction with the video memory. Each location in the video memory corresponds to a pixel in the image. The azimuth two dimensional correlation is achieved by a single multiplier chip and the video memory.

4.4 Video and System Controllers

The video controller is a microcontroller which performs the complex function of retrieving processed data stored in the memory after every time the azimuth processor does a multiplication operation. The video controller also restores the summed data from the azimuth correlator in the original video memory location.

The system controller is a microcomputer coordinating the functions of all the sub-systems of the architecture. It computes the reference functions based on the attitude parameters of the spacecraft and updates the coefficients in the corresponding units. The system controller also supervises the operation of the video control processor and the video output device.

4.5 Video Output

This is the final interface for the image. The data stored in the video memory comprises the final focussed image which can be transmitted or displayed.

4.6 Expected System Performance

The system is efficient, fast and uses no corner turn memory. Also, since a two dimensional azimuth processing is used, no explicit range migration circuitry is needed. The presummer and filter circuit reduce the data rates by a factor of 8. With the high speed multipliers and low data rates, the images, like those from SEASAT, can be processed in real time. Since CMOS VLSI chips are used, the size and power consumption are small and hence, ideally suited for on board space applications. Except for the three custom built ICs, the rest of the chips are available commercially. The memory used in the video processor are either video or BICMOS memories [5].

5 Conclusion

The theory behind the SAR signal processing has been discussed and a physical model is presented. A time domain correlator has been discussed. A real time SAR architecture using these correlators for processing SAR data has been presented. This architecture uses only a small amount of memory and performs no corner turn. The architecture is fast and efficient and uses three custom VLSI chips for its implementation.

Acknowledgements

This work was partially funded by NASA SERC at the Univ. of Idaho. Additionally, we, the authors, want to publically thank and praise the Lord Jesus Christ for influencing our lives and our work through His love, faithfulness, grace, death and resurrection.

References

- [1] S. Y. Kung, *VLSI Array Processors*, Prentice Hall Information and System Science Series, Englewood Cliffs, New Jersey, 1988.
- [2] A. B. Premkumar and J. E. Purviance, "Real Time SAR Processing", *Symposium on VLSI Design*, NASA Space Engineering Center, Univ. of Idaho, Jan 1990, pp. 117-139.
- [3] A. B. Premkumar, "Real Time SAR Imaging: A model and Design using VLSI Correlators", *PhD Dissertation*, Univ. of Idaho, Aug 1990.
- [4] C. Wu, "A Digital Fast Correlation Approach to Produce SEASAT SAR Imagery", *IEEE Int. Radar Conference Record*, Apr 1980, pp. 153-160.
- [5] A. B. Premkumar and J. E. Purviance, "A VLSI Two Dimensional Correlator and its Application to Synthetic Aperture Radar", *Northcon/90 Conference*, Oct 1990.

VLSI Corrector Chip for Space-Borne mm-Wave Radiometer Spectrometers

K. Chandra, J. Tarsala, H. Pickett and W. Wilson
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract - JPL has developed a 52-channel 150 MHz bandwidth autocorrelator spectrometer using specially designed ECL gate array correlator chips. This paper describes the characteristics of the ECL chip and the 52-channel autocorrelator. These autocorrelator spectrometers will be used with space-borne mm-wave radiometers for remote sensing of the Earth's atmosphere and astrophysical observations.

1 Introduction

Radiometer spectrometers using digital correlation techniques are widely used in ground-based radio astronomy observatories. This technique was introduced by S. Weinreb for spectral line radio astronomy in 1961 [1]. Both medium and large scale integrated circuits were used in the earlier digital correlator designs. These correlators required many parts and consumed considerable power. Recent developments in VLSI technology have enabled this important digital processing technique to be implemented as an Application Specific Integrated Circuit (ASIC) and several new digital correlators have been built using ASIC chips. [2] [3] [4]. However, the ASICs made for ground based spectrometer applications were not designed for low DC power consumption.

Stable wideband spectrometers with low DC power consumption are required for space-borne operations. They will be used for remote sensing of the Earth's atmosphere with the Microwave Limb Sounder on the Earth observing system (Eos). There are also requirements for low power spectrometers for future astrophysics space missions such as the Submillimeter Infrared Line Survey (SMILS) and the Large Deployable Reflector (LDR). Correlation spectrometers will also find applications in space-borne thinned antenna arrays and the submillimeter lunar array.

The Jet Propulsion Laboratory is developing digital correlators for space-borne spectrometer applications. This is because of their advantages of high stability, low power, high reliability, small size and low mass. Future developments in both VLSI and material technologies will further reduce the size, DC power requirements and increase the speed for wider signal bandwidths. The digital autocorrelator spectrometer is preferred over other spectrometers, such as the multichannel filterbank and Acousto-Optic Spectrometer (AOS), because of its better stability. Also, the bandwidth and the resolution of an autocorrelator spectrometer can easily be changed by changing the clock frequency and/or by

using additional delay elements. In this paper, the autocorrelator background information will be presented. This is followed by the design of the digital correlator chip and the 52-channel autocorrelator spectrometer.

2 Background

2.1 Autocorrelation Theory

The autocorrelation function of a signal is expressed as follows:

$$R(\tau) = \text{Lim } 1/T \int_0^T f(t) * f(t + \tau) dt \text{ as } T \rightarrow \infty \quad (1)$$

where $f(t)$ is the input signal

τ is the delay time and

T is the integration time.

The incoming signal at microwave frequencies is down converted to baseband frequencies and then divided into two paths. In one path, a delay element with multiple taps at Δt intervals is introduced. The undelayed signal is then multiplied with each output from the tapped delay line and the products are integrated and averaged over the integration time T . The accumulated values represent an estimate of the autocorrelation function of the input signal $f(t)$. A theorem due to Wiener and Khintchine (F. N. H. Robinson, 1974) relates the autocorrelation function, in the time domain, to the power spectrum, in the frequency domain, by the Fourier transform equation:

$$S(f) = \int_0^{\infty} R(\tau) * \cos(2\pi f\tau) dt \quad (2)$$

where $S(f)$ is the power spectrum of the input signal,

$R(\tau)$ is the autocorrelation of the input signal.

The autocorrelation function is even; therefore, only a cosine transform is required.

2.2 Digital Autocorrelator

Figure 1 shows the digital autocorrelator block diagram. In the autocorrelator, the input signal is band limited, sampled at the Nyquist rate and digitized to a few bits. The sampled signal is delayed using shift registers and multiplied with the undelayed sample using simple logic circuits. The multiplied output from each delay stage or channel, is accumulated in a binary ripple counter.

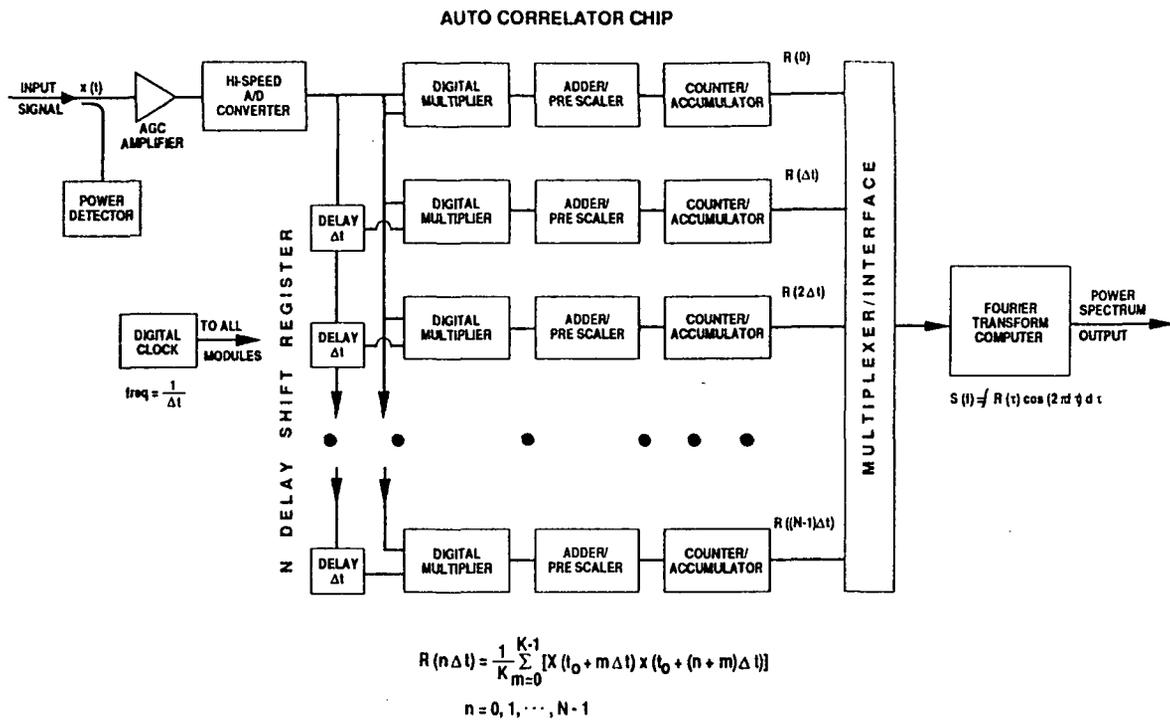


Figure 1: Digital autocorrelator spectrometer

The autocorrelation function for the sampled data can be expressed as:

$$R(n\Delta t) = \frac{1}{K} \sum_{m=0}^{K-1} [X(t_0 + m\Delta t) X(t_0 + (n+m)\Delta t)] \tag{3}$$

where $n = 0, 1, \dots, N-1$ represent the delay in one of the signal paths,

K is the number of products in the integration time T ,

Δt is the delay, usually made equal to the sampling interval.

The power spectrum is calculated by performing a Fourier transform. The N channels (corresponding to the N delay values) in the autocorrelation function are transformed into N points on the frequency domain by using the discrete Fourier transform (DFT) relationship:

$$P \left[\frac{j}{2n\Delta t} \right] = \frac{1}{N} \left[R(0) + 2 \sum_{n=0}^{N-1} R(n\Delta t) * \cos(\pi j/N) \right], \tag{4}$$

where $j = 0, 1, \dots, N-1$,

$P \left[\frac{j}{2n\Delta t} \right]$ represents the power on the j th point on the output spectrum,

$R(0)$ is the correlation coefficient for the zero delay channel ($= 1$ after normalization) and

$R(n\Delta t)$ is the normalized autocorrelation coefficient for delay $n\Delta t$.

The input signal is digitized to only a few bits to permit a higher sampling rate, and thus increase the signal bandwidth. Limiting the number of bits speeds up the multiplication and addition because fewer digital operations are required. However, the Signal to Noise Ratio (SNR) of the correlator is degraded when only a few bits are used. The loss in SNR is 12% when two bit digitization is used [6]. Quantizing schemes representing the input signal by more than two levels to improve the SNR have also been considered by others [7]. However, the size, complexity and power of the digital circuits grow as the number of bits increases. This is of particular concern for space applications, where low DC power is important. The two bit correlator appears to offer the best trade-off between sensitivity, complexity of the hardware, and minimum power.

2.3 Digitizer

The first element in the digital correlator is the analog to digital converter, called the digitizer. The prototype 52-channel autocorrelator spectrometer uses a 2-bit digitizer. One of the bits represents the sign (zero-crossing detector output), and the second bit represents the magnitude. This magnitude bit is assigned a value "one" if the input voltage is outside the pre-determined limits $\pm V_{ref}$. The four states of the 2-bit digitizer and the assigned weighting factors to these states are shown in Table 1.

SIGN	MAGNITUDE	WEIGHT
1	1	-n
1	0	-1
0	0	+1
0	1	+n

Table 1:

Setting the decision level of V_{ref} equal to the RMS voltage of the input signal and $n = 3$, gives an SNR of 88% relative to the continuous correlator [6].

2.4 Correlator Multiplier

The digitized signals, are multiplied after one of the signals is delayed in time using shift registers. Using $n = 3$ for best SNR performance, and normalizing by 3, gives the product table shown in Table 2.

		Undelayed Signal				
		SM	11	10	00	01
Delayed signal	11	+3	1	-1	-1	
	10	1	0	0	-1	
	00	-1	0	0	1	
	01	-3	-1	1	3	

Table 2:

The inner products are deleted to simplify the circuit. This only results in 1% loss to the correlator SNR [7]. A bias of +3 is added to the products, so that only positive numbers need to be added to further simplify the adder circuit. Table 3 shows the final multiplication algorithm used in the hardware realization.

		Undelayed Signal				
		SM	11	10	00	01
Delayed signal	11	6	4	2	0	
	10	4	3	3	2	
	00	2	3	3	4	
	01	0	2	4	6	

Table 3:

2.5 Accumulators

The binary coded outputs from the multiplier are added, using a four bit adder and the carry output from the adder is accumulated using ripple counters. The length of the binary counters is determined by the rate at which the computer reads the counter values – typically a few times a second. The counter length is determined by the number of product terms that can be accumulated during each integration time.

3 52-channel autocorrelator spectrometer hardware

The prototype 52-channel, digital autocorrelator spectrometer (DACS) uses ECL correlator chips. The chip was designed by JPL, and was made by the Raytheon Corporation using 2 micron ECL gate array technology. This chip has 26 channels, requires 120 milliwatts of DC power per delay channel, and can be clocked up to 300 MHz. The power requirements may be considered as moderate to low compared to the other designs that have been implemented. A block diagram of the prototype autocorrelator spectrometer is shown in Figure 2.

3.2.6

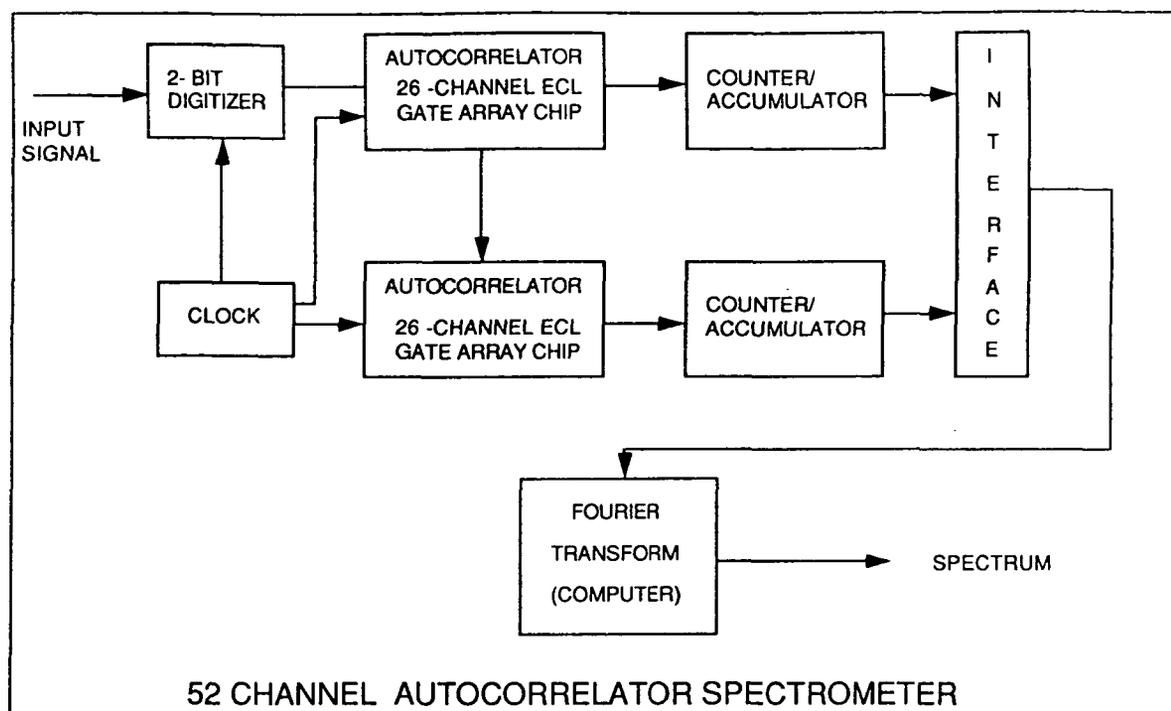


Figure 2: Block Diagram

3.1 2-bit digitizer

The 2-bit digitizer was designed at the California Institute of Technology for mm-wave astronomy (S. Padin and M. Ewing, 1989). The digitizer board uses a Plessey SP 93808 sub-nanosecond octal comparator chip. The digitizer board is made of Duroid-based copper clad material. Microstrip transmission lines are used to provide a 50-ohm characteristic impedance for ECL signals which have edge speeds of 1 nanosecond.

Three comparators are used for digitizing the input analog signal into a 2-bit word. The two comparators which detect the magnitude of the input signal are ORed together. The OR output and the output from a zero-crossing detector provide the simple encoding scheme shown in Table 1 for the two bit output. A stable, low noise, reference voltage generator is used for the comparator decision levels.

3.2 ECL Autocorrelator Integrated Circuit

The digital correlator is an Application Specific Integrated Circuit (ASIC), specially designed for the digital spectrometer development. The chip design was done using the CAD system on a Mentor workstation. The chip was made by Raytheon Corporation using the silicon foundry at Bipolar Integrated Technology Company in Beaverton, Oregon. Their unique wafer fabrication technique reduces transistor and metal capacitances, that reduces the device power, while maintaining ECL speeds.

The autocorrelator design was created using hierarchical design methodology. Pre and post layout simulations were performed on the design to verify the logic function and also to verify the circuit timing for proper operation at clock speeds up to 250 MHz. The design

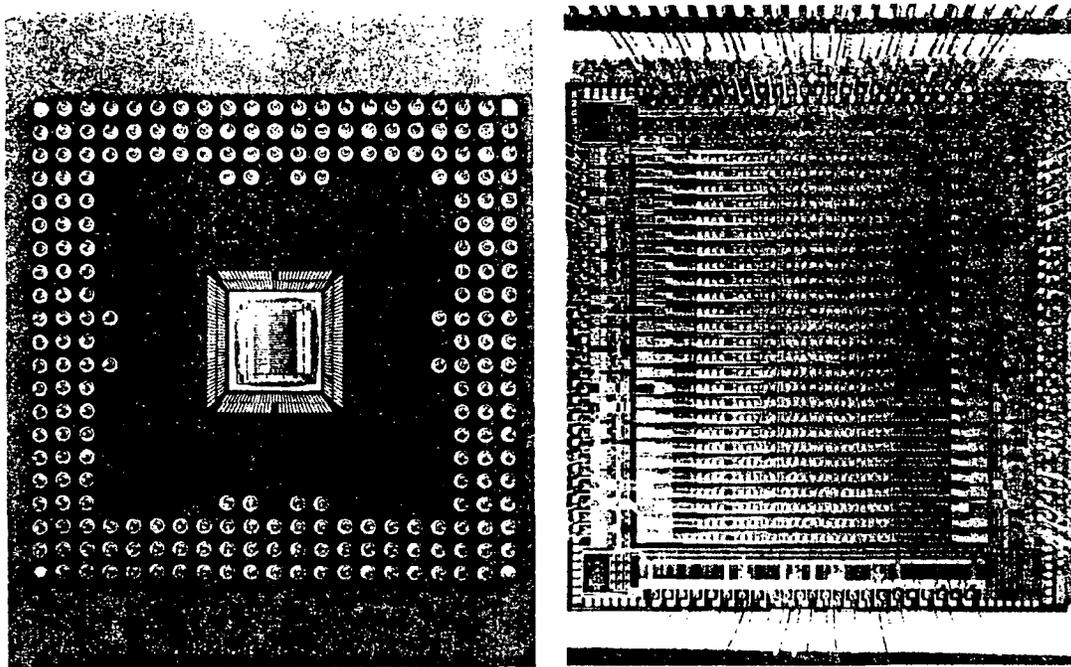


Figure 3:

hierarchy made it easier to run the simulation program on every schematic level and to identify the critical paths. The critical path analysis identified interconnecting lines (nets) in the multiplier/accumulator logic, and also in the pipeline architecture. The propagation delays in these critical paths were reduced by careful layout. A fault grading was done for the design, using the simulation vectors, that revealed a coverage of about 70%. The 30% loss in coverage was because the design utilized macrocell parts from Raytheon's library and some internal nodes in these parts were permanently tied to supply or ground.

The circuit layout was created for Raytheon's CGA 70E18 gate array topology which offers 12,800 gates and 176 input and output connections. The 26 channels of the correlator were laid out in the 26 contiguous rows, as shown in Figure 3. The array utilization for the correlator design is about 61%.

The silicon die, measuring 336×364 mils, with the correlator design, was packaged in a custom Pin Grid Array (PGA) package which was developed by Raytheon for its ECL gate array chips. The PGA is a 229-pin square package measuring 2.1 inches on a side. The pinout configuration for the correlator chip was selected so there was minimum skew in the timing of the signals inside the PGA package. It was also developed to make the signal routing easier on the printed circuit board with minimum number of plated through holes in the signal path.

The analog signal which is digitized and sampled is divided into two signals externally

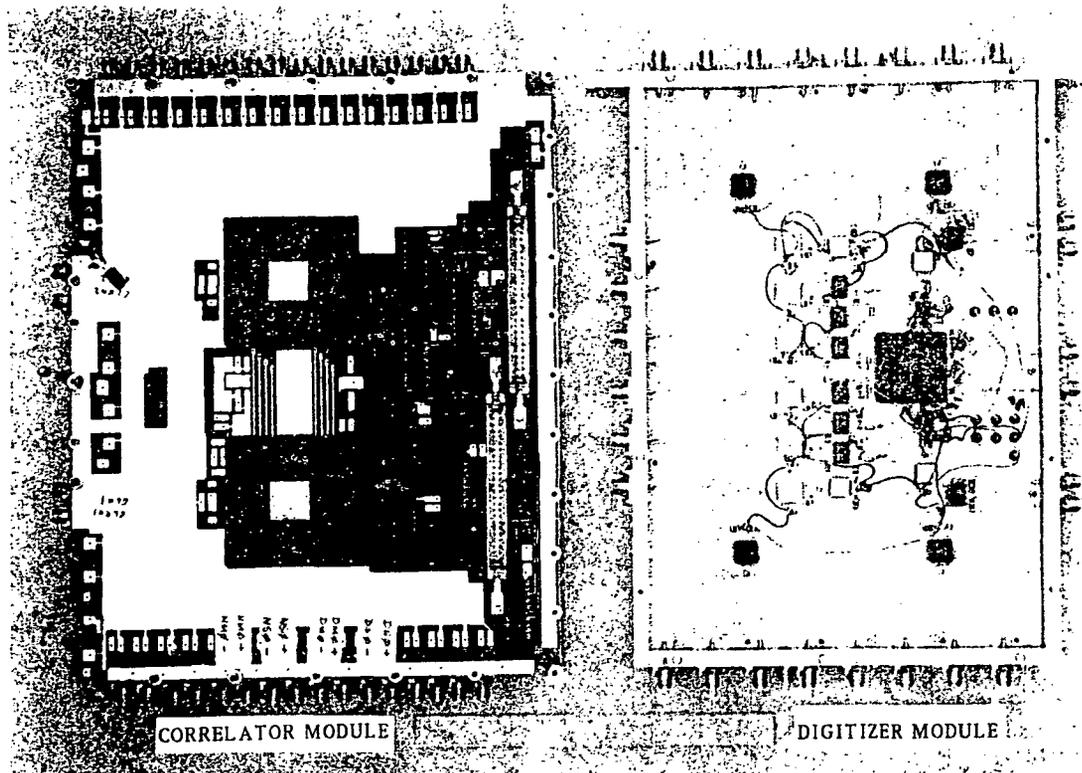


Figure 4:

and then given as inputs to the correlator. There are two signal paths inside the chip. One signal path is the direct or undelayed data, and the second signal path is the delayed data. The data in the second signal path is delayed by a 26 stage shift register inside the chip. The output from each stage of the 26-bit shift register is multiplied with the undelayed sample and the products are accumulated in a binary counter. A pipeline technique is used in the data path to increase the clock speed.

Each channel in the correlator chip has a 4-bit multiplier followed by a 4-bit adder/accumulator stage. The two bit multiplication algorithm, shown in Table 3, is used. The carry output from the most significant bit (MSB) of the adder is scaled by a 3-bit binary counter. The 3-bit prescaler is used in each channel before the correlator data is output to an I/O pin. The prescaler reduces the output data rate allowing the use of low power CMOS counters for further accumulation outside the chip.

Standard 100K ECL logic is used for the signal inputs and outputs. The high speed signals such as clock, data inputs and outputs use differential drive. The correlated outputs use standard TTL logic.

3.3 52-channel Autocorrelator

The autocorrelator board, shown in Figure 3.3, is a four layer printed circuit board with outside layers used for interconnection and inside layers used as ground and VCC planes. The four layers are arranged in such a way that the outside layer and the layer below have 50 Ohm microstrip transmission lines for the high frequency signals. Two correlator chips are cascaded on the board to provide 52 channels. The TTL outputs from the correlator chips are buffered externally with BiCMOS drivers.

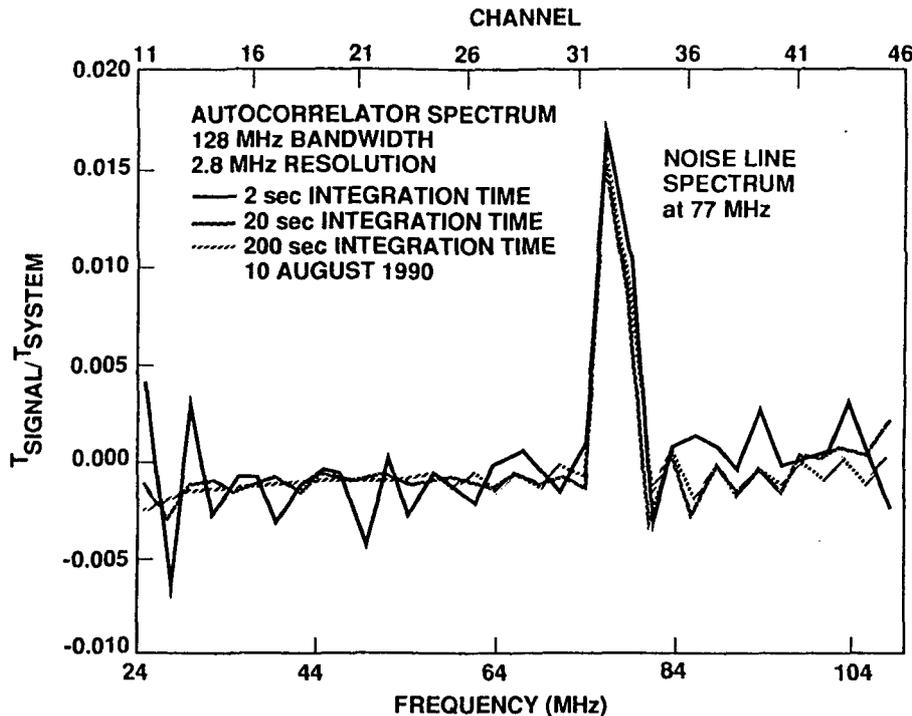


Figure 5: Digital Correlator

The outputs from the 52-channel correlator module are TTL or CMOS compatible. The output from each channel is accumulated using a 24-bit binary counter. Two ASIC CMOS counter chips are used for accumulating the 52-channel outputs. Each counter chip contains 32, 18-bit counters, and accumulate the correlation values for up to a one second integration period.

A PC compatible computer with a digital I/O card are used for the data acquisition and the Fourier transform. A one second pulse interrupts the computer after every integration period. At the interrupt, the computer reads all the 52 counter values in byte mode, and each channel value is arranged as an 18-bit word in memory. The counters are cleared, and the correlation continues for the next integration period.

4 Power spectrum measurement

A wideband noise source, with a simulated spectral line, was used to test the autocorrelator spectrometer. The noise source output is prefiltered with a 100 MHz low pass filter to eliminate aliasing when the digitized signal is sampled at 250 MHz. The simulated line source was generated by passing white noise through a 1 MHz wide bandpass filter centered at 77 MHz and added to the broadband noise. The power spectrum, measured with the simulated line source, for three integration times is shown in Figure 5. These curves show the reduction in spectrum noise as the integration time is increased.

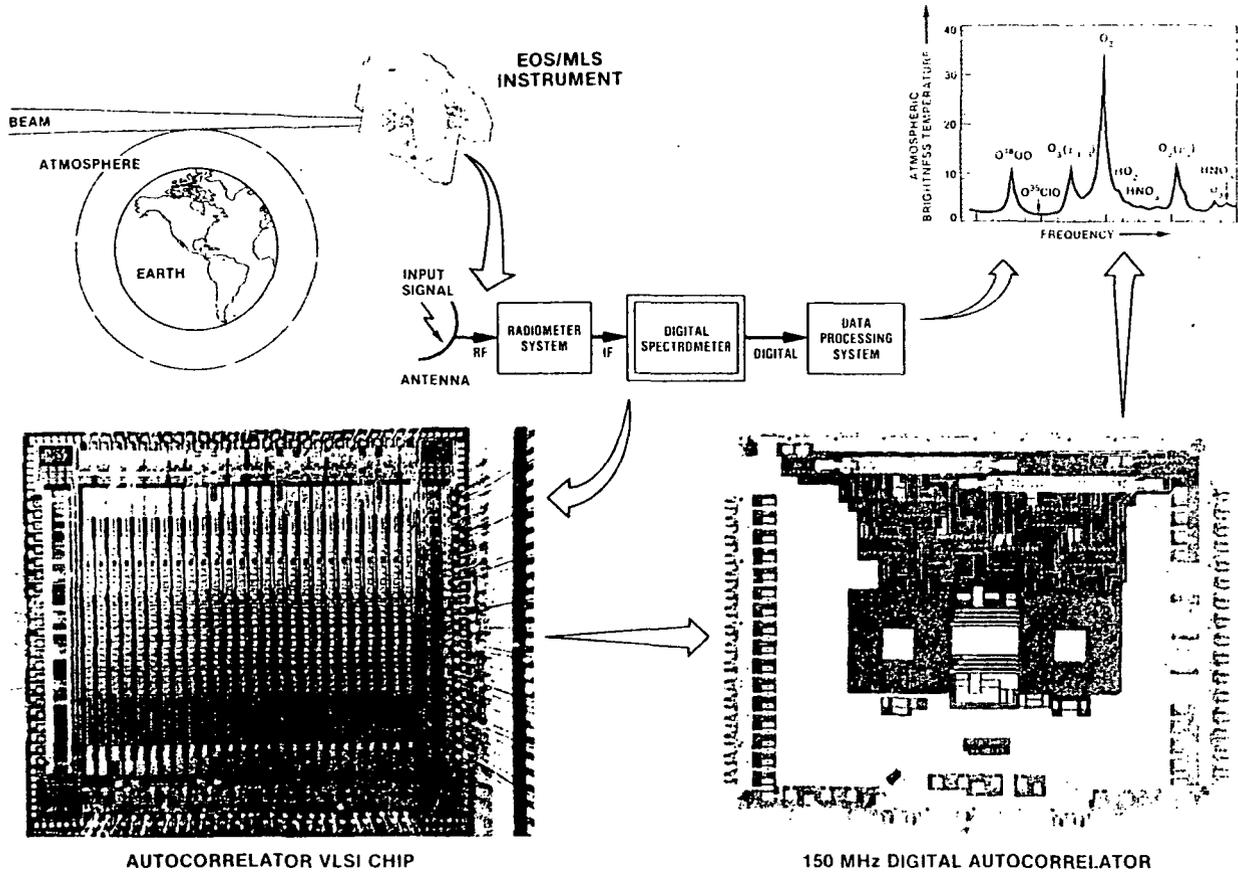


Figure 6:

5 Applications

The digital autocorrelator spectrometer using VLSI chips is an attractive choice for the space-borne applications due to its stability, small size, mass, and low power requirements. Figure 5 shows one of the applications where the digital autocorrelator spectrometer will be used with millimeter-wave radiometers for spectral analysis of molecular emission lines in the Earth's atmosphere. Other space applications, where the digital autocorrelator spectrometers may be used, are shown in Table 4.

6 Conclusion

The digital spectrometer using the autocorrelation technique will replace analog filterbank spectrometers which are large, massive and require a large DC power. The goal of the

Flight Project	Science Objective	Requirement	Launch Date
EOS/Microwave Limb Sounder (EOS/MLS)	Atmospheric Ozone depletion/chemistry	40 Wideband 10 Narrowband low power spectrometers	2000
Submillimeter Moderate Mission (SMMM)	Astrophysics interstellar molecules	5-10 wideband low power spectrometers	2001 cooperative project with France (CNES)
Advanced Microwave Sounding Unit C (AMSU-C)	Earth upper atmosphere temperature sounding	narrowband low power spectrometer	1996
Lunar Submillimeter Interferometer	Astrophysics, high spatial resolution imaging	many wideband low power, cross correlator spectrometers	> 2005
Large Deployable Reflector (LDR)	Astrophysics, interstellar molecules	many wideband low power spectrometers	> 2010

Table 4:

digital correlator development program is to develop spectrometers with bandwidths, to 2 GHz, and low DC power consumption of 5 milliwatts per delay channel. The present ECL correlator chip, operates up to a bandwidth of 150 MHz, and requires about 120 milliwatts of DC power per delay channel. Advances in digital technology and new material processes will increase the bandwidth of the digital technique and reduce the power requirements. These spectrometers will find space applications in remote sensing of the Earth's atmosphere and astrophysical observations.

Acknowledgment:

The digital autocorrelator spectrometer development was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. This project was initially funded by the JPL Director's Discretionary Fund. Later this work was supported by the NASA Civil Space Technology Initiative (CSTI).

References

- [1] S. Weinreb, MIT Research Laboratory of Electronics, Tech. Rep. 412, (1963).

- [2] M. Ewing, High-Speed Correlator Chip, OVRO MM Correlator memo. 22. California Institute of Technology, 1987.
- [3] A. Bos, *Functional design of a wideband digital spectrometer*, Netherlands Foundation for Radio Astronomy, Internal Tech. Rep. 179, 1986.
- [4] R. Escoffier, National Radio Astronomy Observatory, VLBA correlator Memo. 87 and 91, 1987.
- [5] F. N. H. Robinson, *Noise and Fluctuations*, Clarendon Press, Oxford, 1974.
- [6] B. F. C. Cooper, *Correlators with two-bit quantization*, Aust. J. Phys, Vol. 23, 1970.
- [7] B. F. C. Cooper, *Methods of Experimental Physics*, Vol. 12 - Part B, Chapter 3.5, "Autocorrelation Spectrometers", Academic Press, 1976.

A High Speed CMOS Correlator

J. Canaris and S. Whitaker
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - A full custom, 25 MHz, 1.6 μ m CMOS Correlator chip is presented. The 5.15mm by 4.23mm chip performs either autocorrelation or crosscorrelation, consuming less than 10mW per channel. The correlator, designed for a space borne spectrometer, contains 32 channels. The 24 bit accumulator registers can be read independent of the input data path, in either 8 bit bytes, or 16 bit words. The device is cascadable and allows integration periods of up to 1.78 seconds, at 25 Megasamples/second. The controllers, for the input data path and the data output section, are implemented with Sequence Invariant State Machines.

1 Introduction

A high speed, low power CMOS correlator chip is presented in this paper. The correlator, designed for a space borne spectrometer contains 32 time-lag channels, each of which contains a biasing multiplier, a 4 bit accumulator and a 24 bit counter. The sensing instruments provide the chip with two 2 bit input words, which can be either the same signal, for autocorrelation, or different signals, for crosscorrelation. The biasing multiplier does not perform binary multiplication, but implements a special function described in Section 4. External control of the correlator is quite simple, requiring only a reset pin and a pin to signal the end of an integration period. Simple handshaking is provided through a single output pin, which signals when data is available to be read from the output port. Output data can be read while integration is in progress, in either 8 bit bytes or 16 bit words, under the control of a user provided strobe. The correlator is capable of maintaining a 25 Megasample/second input data rate, with integration periods of up to 1.78 seconds. Data can be output from the chip at 10 MHz. The chip consumes less than 10mW/channel of average power. Auxiliary ports are provided for both of the data inputs.

The data path of this chip is extremely regular, the initial layout of the core required only 30 hours to complete. The controllers for the input data path and the data output section are implemented with Sequence Invariant State Machines [1], and were initially layed out with a compiler described in [2]. This chip is amongst the first VLSI designs to utilize Sequence Invariant State Machines.

Some of the main features of the correlator chip are listed below.

3.3.2

- Autocorrelation or Crosscorrelation
- 25 Megasamples/second
- 32 channels
- Up to 1.78 second integration time at 25MHz
- Low Power ($\leq 10mW$ per channel)
- Cascadable
- Selectable auxiliary ports on the data inputs
- Integration can continue while data is output
- CMOS and TTL compatible inputs

2 General Description

The correlator chip accepts two 2 bit data streams clocked at a maximum rate of 25MHz. Delayed versions of one stream are multiplied with the current data of the other stream. Products for each delay (channel) are accumulated and the accumulator overflows are counted. This procedure continues for one integration period, as defined by a control line (INT) held low. Integration is performed continuously until INT is strobed high. At this time the overflow counters from each channel are isolated from their respective accumulators. After the counters have settled DATARDY going high signals that data is available for output. The overflow counters are cleared and are reconnected to the accumulators and a new integration period begins at this time. The contents of the overflow counters are output, under user control, through a 24 bit wide shift register after DATARDY goes high. Data output is either word serial or byte serial, under user control. When word serial mode is selected only the 16 most significant bits of each channel are output. DATARDY will remain high until all of the 32 output registers have been read, regardless of which output mode is selected. A test mode is provided to decrease the time required to test the onboard overflow counters.

3 Functional Description

3.1 Initialization

The chip must be powered up with RN held low for at least 10 clock cycles, while OUTCK and INT are held low. This will bring the chip into a sanity state while guaranteeing that the output pads will be tristated. During this time the overflow counters will be cleared and the control state machine will be prepared for normal operation. Integration will begin on the clock following RN being brought and held high. The delay path shift register and

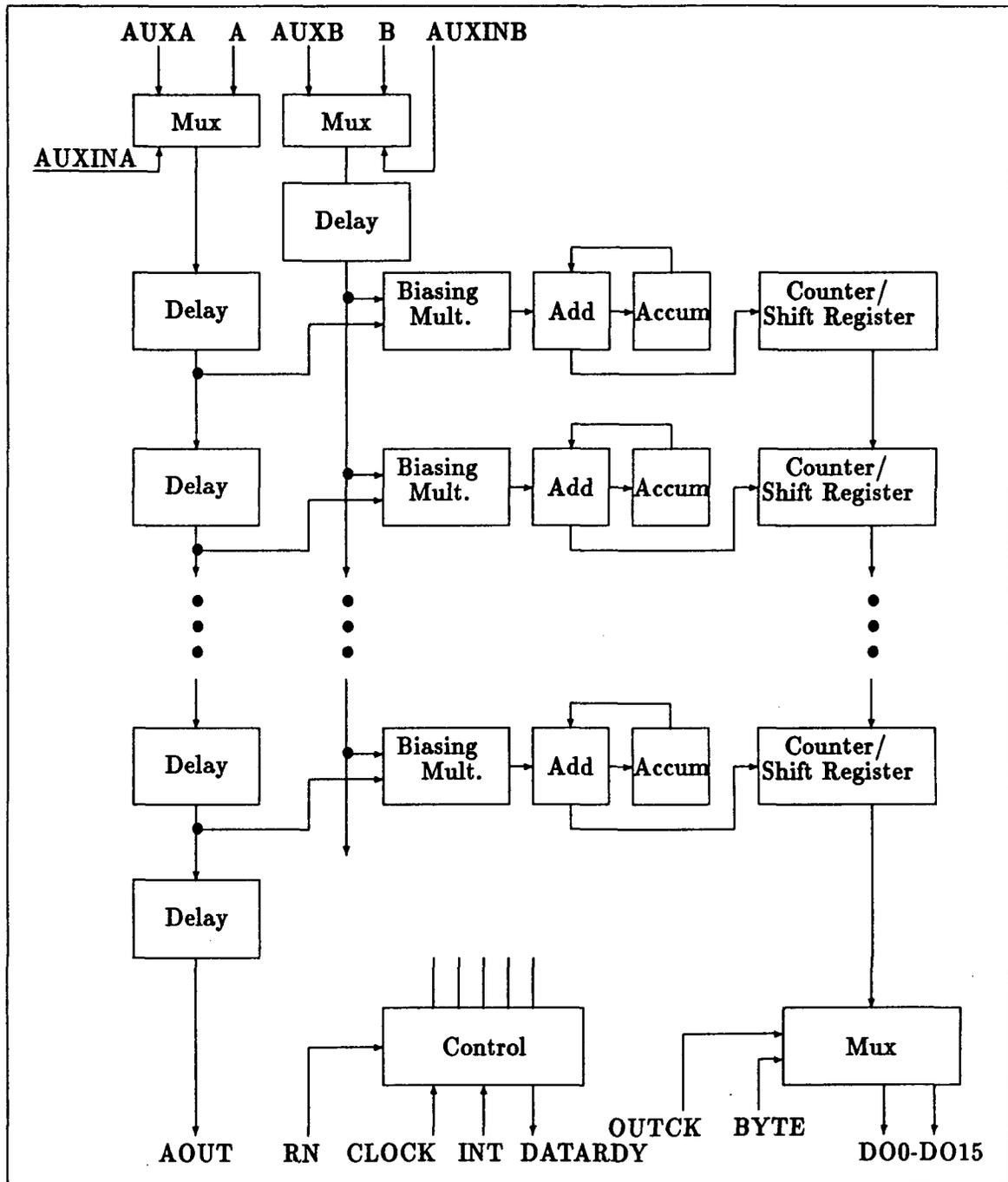


Figure 1: Functional Block Diagram

3.3.4

the channel accumulators can never be cleared. Figure 1 shows a block diagram of the correlator chip.

3.2 Data Input

Data will be input to the chip on the A and B data buses (A0,A1 B0,B1) every clock cycle. Although data will still be clocked into the chip, processing will not occur between INT being strobed high and DATARDY going high. Pins A0 and A1 are the least significant and most significant, respectively, bits of the delay line. B0 and B1 are the least significant and most significant, respectively, bits of the undelayed signal.

Additionally, two auxiliary input ports (AUXA0,AUXA1 AUXB0,AUXB1) are provided. These ports are multiplexed with the A and B buses, respectively, under the control of the AUXINA and AUXINB pins. When AUXINA is held high, the A bus becomes the input port to the chip, when AUXINA is held low the AUXA bus becomes the input port to the chip. When AUXINB is held high, the B bus becomes the input port to the chip, when AUXINB is held low the AUXB bus becomes the input port to the chip. The auxiliary input ports behave identically to the primary ports.

3.3 Correlation

Correlation begins on the clock following RN being brought and held high or when INT is held low and DATARDY goes high (signaling that data is ready from the previous integration period). At that time each channel will multiply the data on the B bus with the output from it's respective delay element. The product will be accumulated with the previous sum for that channel. Any overflow from the accumulator will be counted in the overflow counter of that channel. This process will continue until the INT signal is strobed high for at least 1 clock cycle.

The multiplications are not purely binary in nature. The output of the multiplier is biased in a manner described in Section 4. The accumulator contains a four bit adder and four bit register. The carry out of the adder is the toggle signal into the the overflow ripple counter. The overflow counters are 24 bits wide, allowing for the count of up to $2^{24} - 1$ overflows. The frequency of overflow is a function of sample frequency and the length of the integration period.

3.4 Data Output

At the end of the integration period, signaled by INT being strobed high for at least 1 clock cycle, the overflow counters will be isolated from the processing elements. After the overflow counters have settled (10 clock periods, maximum) the contents of the counters will be dumped into an output shift register and the chip will signal that data is ready (DATARDY) on the output bus. When DATARDY goes high the first data can be read from the output bus on the next rising edge of OUTCK. When DATARDY goes high a

new integration period begins. DATARDY will remain high until all 32 output registers have been read, regardless of which output mode is selected.

At all times, after the clock starts, data will be output from the end of the delay line at pins AOUT0 and AOUT1. AOUT0 is the delayed version of A0 and AOUT1 is the delayed version of A1. These pins may be used in a cascaded system by connecting them to the A0 and A1 pins of the next chip downstream.

Data output is, optionally, word serial or byte serial. Holding the output control signal (BYTE) high during the output phase will output the 24 bits of the counters in 8 bit bytes, most significant byte first. Pins DO0-DO7 will be used (DO0 being the least significant bit). Pins DO8-DO15 will be tristated. Holding BYTE low will cause the 16 most significant bits of the counter to appear on pins DO0-DO15.

The output pins DO0-DO15 will be tristated whenever DATARDY is low or when DATARDY is high and OUTCK is low.

3.4.1 Word Serial Mode

When BYTE is held low (word serial mode), successive output words will be clocked out by the rising edge of OUTCK. During the low half of the OUTCK cycle the output bus will be tristated. OUTCK has a minimum frequency of 0Hz and a maximum frequency of 10MHz. OUTCK periods do not have to be of equal length and the duty cycle need not be 50%, but a minimum pulse width of 44ns is required.

During the output phase the output data shift register will be cleared. The output clock must be strobed 32 times to unload the output shift registers. Data on the A and B buses continues to be input to the chip during the output phase.

Data output is terminated by bringing and holding OUTCK low after reading out all 32 channels.

3.4.2 Byte Serial Mode

When BYTE is held high (byte serial mode), successive output bytes will be clocked out by the rising edge of OUTCK. During the low half of the OUTCK cycle the output bus will be tristated. OUTCK has a minimum frequency of 0Hz and a maximum frequency of 10MHz. OUTCK periods do not have to be of equal length and the duty cycle need not be 50%, but a minimum pulse width of 44ns is required.

During the output phase the output data shift register will be cleared. The output clock must be strobed 96 times to unload all of the output registers. This option allows access to the 8 least significant bits of the overflow counters, as well as providing an 8 bit data bus.

At the end of the output phase the chip will be ready to begin a new correlation. Data on the A and B buses continues to be input to the chip during the output phase.

Data output is terminated by bringing and holding OUTCK low after reading out all 32 channels.

3.5 Test Mode

Test mode provides a method for checking the operation of the 24 bit overflow counters. Test mode is entered by bringing and holding TEST high, while in an integration period. Test mode breaks the overflow counters into three 8 bit counters, the inputs to which are the overflow bit of the adder in each channel. An appropriate input pattern must be applied to the A and B buses during test mode operation. Access to the counters is the same as during normal operation.

4 Biasing Multiplication

The multiplication to be performed takes two 2 bit input words and forms a 3 bit product. The mapping of data is described in Figure 2. B0 and B1 are the real-time inputs. A0 and A1 are the delay line inputs. P0, P1 and P2 are the product outputs.

B1	B0	A1	A0	P2	P1	P0
0	0	0	0	0	1	1
0	0	0	1	1	0	0
0	0	1	0	0	1	1
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	1	1	0
0	1	1	0	0	1	0
0	1	1	1	0	0	0
1	0	0	0	0	1	1
1	0	0	1	0	1	0
1	0	1	0	0	1	1
1	0	1	1	1	0	0
1	1	0	0	0	1	0
1	1	0	1	0	0	0
1	1	1	0	1	0	0
1	1	1	1	1	1	0

Figure 2: Biasing Multiplication Truth Table

5 Data Path Design

Each of the 32 channels integrated on the correlator chip are identical. A single channel consists of 2 delay elements for the time-lag input signal, one biasing multiplier, one 4 bit adder with a 4 bit accumulator register and a 24 bit counter.

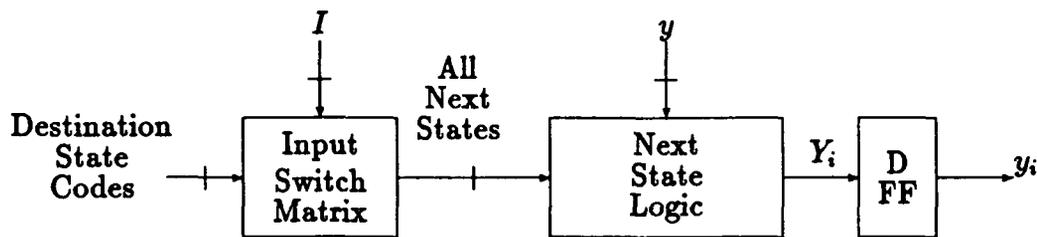


Figure 3: General block diagram.

The overflow counter, which stores correlator integration values is an asynchronous ripple counter. The asynchronous design reduces the power requirements of the chip. The biasing multiplier is implemented as n-transistor pass network [3,4,5,6], which yields a very dense, regular, combinational logic network, while providing the operational speed required by the 25 MHz clock frequency. The four bit adder is implemented as a Transmission Gate Conditional Sum Adder [7,8]. This adder provides the performance needed while minimizing both capacitive loads and silicon area required to implement this function. In general, the use of pass transistor networks reduces nodal capacitance, which is an important consideration in low power applications. Each channel also contains two pipe registers, between the carry out of the adder and the input to the overflow counter. These registers serve two purposes. First, the propagation path would be too long for a 40ns clock period and second the registers provide a means for isolating the ripple counters from the input data path. At the end of an integration period, the counters must be allowed to settle, before output data is ready for reading. All registers are static.

6 Controller Design

The correlator chip requires two controllers. The input data path controller maintains the state information required by the 32 channels. The input data path has two main states. The first state is normal integration and the second is the preparation of integrated data for output. The output controller provides the control of the output shift register and the formatting of the data sent to the output port.

Both controllers were designed using Sequence Invariant State Machines [1]. The logic of such state machines is invariant with respect to the actual sequence required. Only the number of states and inputs need to be known to specify the logic.

A general block diagram of a Sequence Invariant State Machine is shown in Figure 3. The logic that forms each next state equation, Y_i , consists of a storage device (a D flip-flop), next state excitation circuitry, a Binary Tree Structured (BTS) network, which generates the next state values to the flip-flop, and input logic consisting of a pass transistor matrix. Present state information is fed back to the next state logic and input information drives the input switch matrix.

A general BTS network is employed to formulate the next state equations for sequential circuits. This general BTS network represents a complete decoding of an input space and

hence only constants are input to the network. Any specific function can be realized by simply changing the pass variable constants, $1(0)$, at the input to the appropriate branch. The input matrix is programmed with appropriate connections to $1(0)$ to produce the desired state transitions. Changing the sequence of operation merely requires a reprogramming of these connections. For the correlator state machines, the input switch matrix was eliminated by applying the inputs, I , as pass variables to the BTS network. Work to produce a general theory for this logic reduction is currently under way at the UI NASA SERC.

6.1 Input Controller

During integration, the input state machine connects the 24 bit overflow counters to the calculation section, while disconnecting the output shift register. At the end of an integration period, signaled by INT going active, the controller moves through a fixed set of states. The state machine first isolates the overflow counters from the calculator. The machine then steps through a number of states while the ripple counters settle. At that time the counters are first parallel loaded into the output shift register and then reset. The state machine then reconnects the counters to the calculator, beginning a new integration period. This controller also sets a latch which signals that data is ready for reading.

The use of Sequence Invariant State Machines proved invaluable in this application. At the time of initial logic design, it was unknown how long it would take for the ripple counters to settle. It was desirable to minimize the length of time that the counters were disconnected, as this time shortens the integration period. The state machine was initially designed with a number of wait states which was deemed sufficient. After circuit design was finished on the counters, several states could be removed. The redesign of the state machine required about 10 minutes of engineering time and about 10 minutes of layout time. This is a significant improvement over traditional state machine designs. The output equations of the signals required to control the data path are formed by logical blocks which are identical to those in the state machine itself, as described in [9]. Again as the number of states changed, the output equations were also easily modified.

The input state machine requires only two external signals, INT and RESET, for proper operation.

6.2 Output Controller

The output state machine provides signals which control the output and formatting of correlated data. This chip has two modes for data output, byte serial mode and word serial mode, which are selected with the BYTE pin. When the circuit is in byte mode the 24 bit counters are read out in 8 bit bytes. Word mode sends only the upper 16 bits of each register. Output is under user control. New data appears on the output pins on the rising edge of OUTCK. When OUTCK is low the output pins are tristated. Additionally, when in byte mode, the upper 8 bits of the output port are tristated. When all data has been read the DATARDY flag is reset.

The output controller, therefore, must control several operations. Byte mode requires that the three 8 bit portions of the overflow counters be multiplexed onto the lower 8 bits of the output port. The mux control signals are formed by the output controller. Additionally, the tristate signal, for the upper 8 bits of the output port, is provided by this machine. A seven bit counter is decoded for either byte or word mode. This counter maintains a count of the amount of data which has been read out. When the output shift register is empty, the DATARDY flag is reset. This controller also provides the clock for the output shift register itself.

Again, as in the input controller, Sequence Invariant State Machines were utilized in this controller. As logic design progressed, inevitable changes were easy to implement with these functional blocks.

This controller requires three external signals. BYTE to signal which output mode is active, DRI which is the data ready signal from the input controller and RESET.

7 Layout

The mask design of the correlator chip was straight forward as the structure is extremely regular. The base cells and the layout of the correlator core required only 30 hours of layout time. The core of the chip, the 32 correlator channels, contains 31,948 transistors. The n-transistor to p-transistor ratio is 1.77, which reflects the extensive user of n-transistor pass networks. The silicon area consumed by the core is 3.49mm by 2.52mm , which yields a density of $275.3\mu\text{m}^2/\text{device}$.

The layout of the Sequence Invariant State Machines was done with a pre-released version of the silicon compiler described in [2]. The correlator chip, as a whole contains no more than 120 random devices. Figure 4 is a plot of the correlator chip.

8 Summary

A 25 MHz CMOS correlator chip has been described. The chip provides either crosscorrelation or autocorrelation of 2 bit input signals at a data rate of 25 Megasamples/second. The 32 channel chip, designed for space applications, consumes no more than 10mW/channel. The VLSI circuit has two options for data output and provides a simple handshaking scheme. The layout of the correlator is highly regular and has taken advantage of Sequence Invariant State Machines, in the controller design.

9 Acknowledgements

The authors wish to acknowledge the NASA Space Engineering Research Center program, whose support made this design possible. The staff of the Microelectronics Research Center also deserves credit for all of the assistance they have provided. The support of Dr. Gary Maki, director of the MRC, was also invaluable.

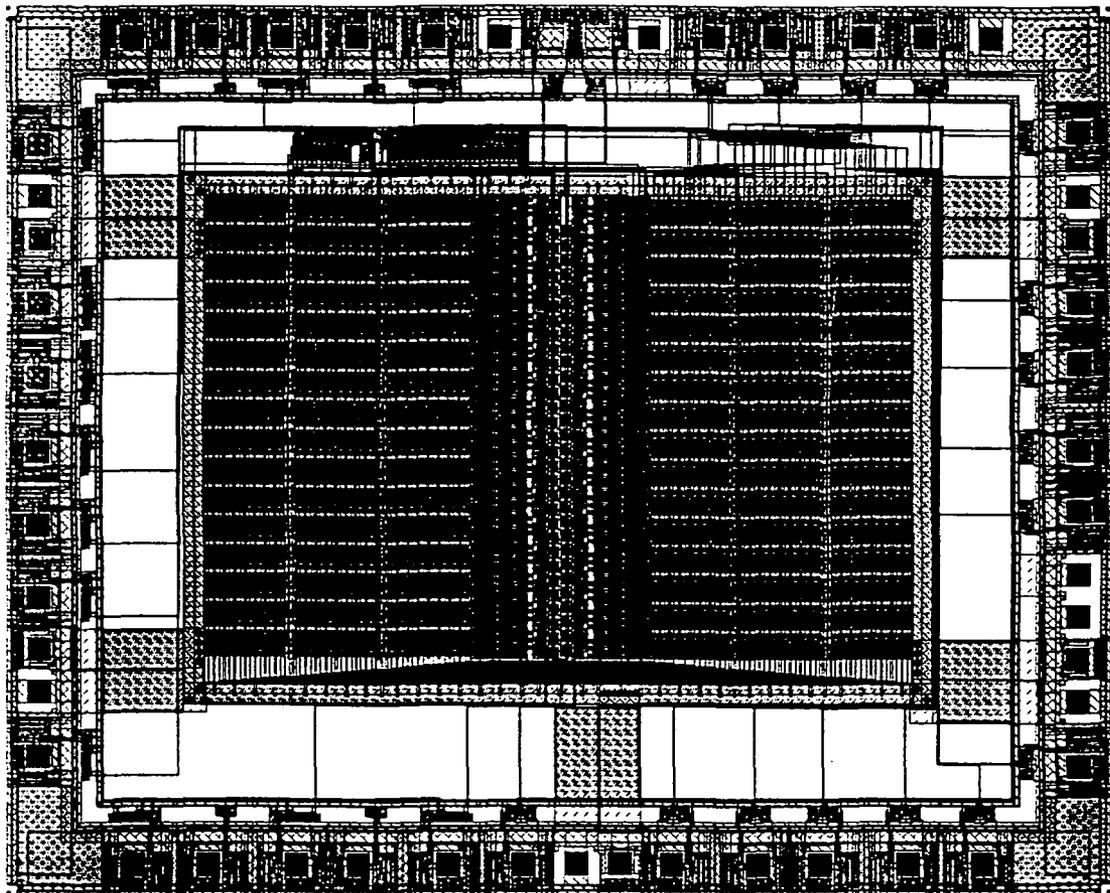


Figure 4: Correlator Chip Plot

References

- [1] S. Whitaker and S. Manjunath, "Sequence Invariant State Machines," Proceedings of the NASA SERC 1990 Symposium on VLSI Design, Moscow, Idaho, January 1990, pp. 241-252.
- [2] D. Buehler, S. Whitaker and J. Canaris, "Automated Synthesis of Sequence Invariant State Machines," Paper to be presented at the 2nd NASA SERC Symposium on VLSI Design 1990, Moscow, Idaho, November 1990.
- [3] D. Radhakrishnan, S. Whitaker and G. Maki, "Formal Design Procedures for Pass Transistor Switching Circuits," IEEE JSSC, Vol. SC-20, April, 1985, pp. 531-536
- [4] G. Peterson and G. Maki, "Binary Tree Structured Logic Circuits: Design and Fault Detection," Proceedings of IEEE International Conference on Computer Design: VLSI in Computers, Port Chester, NY, Oct. 1984, pp. 671-676.
- [5] C. Pedron and A. Stauffer, "Analysis and Synthesis of Combinational Pass Transistor Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 7, July 1988, pp. 775-786.
- [6] J. Canaris, "A High Speed Fixed Point Binary Divider," Proceedings ICASSP-89, Glasgow, Scotland, May 1989, pp. 2393-2396.
- [7] A. Rothermel *et al.*, "Realization of Transmission-Gate Conditional-Sum (TGCS) Adders with Low Latency Time," IEEE JSSC, Vol. 24, June 1989, pp. 558-561.
- [8] J. Canaris and K. Cameron, "A Comparison of Two Fast Binary Adder Configurations," Proceedings of the NASA SERC 1990 Symposium on VLSI Design, Moscow, Idaho, January 1990, pp. 78-86.
- [9] S. Whitaker, G. Maki and M. Canaris, "A Programmable Architecture for CMOS Sequential Circuits," Proceedings of the NASA SERC 1990 Symposium on VLSI Design, Moscow, Idaho, January 1990, pp. 223-230.

Frequency Domain FIR and IIR Adaptive Filters

D. W. Lynn
Department of Electrical Engineering
University of Idaho
Moscow, Idaho

Abstract

A discussion of the LMS adaptive filter relating to its convergence characteristics and the problems associated with disparate eigenvalues is presented. This is used to introduce the concept of proportional convergence. A novel approach is used to analyze the convergence characteristics of block frequency-domain adaptive filters. This leads to a development showing how the frequency-domain FIR adaptive filter is easily modified to provide proportional convergence. These ideas are extended to a block frequency-domain IIR adaptive filter and the idea of proportional convergence is applied. Experimental results illustrating proportional convergence in both FIR and IIR frequency-domain block adaptive filters is presented.

1 The LMS Adaptive Filter

We first present Widrow's LMS adaptive filter, analyzing it's mean convergence following his approach in [5]. The equations describing an FIR filter in both scalar and vector notation are

$$y_j = \sum_{i=0}^{N-1} w_i x_{j-i} = W^T X_j = X_j^T W$$

with $X_j^T = [x_j, x_{j-1}, x_{j-2}, \dots, x_{j-(N-1)}]$ and $W^T = [w_0, w_1, \dots, w_{N-1}]$.

The goal of an adaptive filter is to automatically adjust the weights W so that the difference between the output $\{y_n\}$ and some desired signal $\{d_n\}$ is a minimum in a mean square sense. That is, with $e_j = d_j - y_j = d_j - W^T X_j = d_j - X_j^T W$

$$E \langle (e_j)^2 \rangle = E \langle d_j^2 \rangle - 2E \langle d_j X_j^T \rangle W + W^T E \langle X_j X_j^T \rangle W$$

where $E \langle \rangle$ represents the expectation operator. With $P_{xd} = E \langle d_j X_j \rangle$ (the cross-correlation between $\{d_n\}$ and $\{x_n\}$) and $R_{xx} = E \langle X_j^T X_j \rangle$ (the autocorrelation of $\{x_n\}$), the mean square error ξ can be written

$$\xi = E \langle d_j^2 \rangle - 2P_{xd}^T W + W^T R_{xx} W \quad (1)$$

Since the MSE (mean square error) is a hyperquadratic function of the weights, in essence a bowl shaped surface with a single global minimum, the minimum can be sought using simple gradient search techniques. That is

$$W_{j+1} = W_j - \mu \nabla_j$$

where μ is a scalar constant that controls the rate of adaptation, and ∇_j is the gradient of the error surface with respect to the weights evaluated at time j .

$$\nabla_j = \frac{\partial E \langle e_j^2 \rangle}{\partial W} = -2P_{xd} + 2R_{xx}W \quad (2)$$

Setting gradient to 0 yields the optimum choice of W for minimum mse

$$W^* = R_{xx}^{-1}P_{xd} \quad (3)$$

Since R_{xx} is positive definite, it is non-singular and its inverse exists. This equation is the Wiener-Hopf equation written in matrix form, thus W^* represents the optimum Wiener filter.

Obviously, if R_{xx} and P_{xd} are known exactly (3) gives the optimum weights and requires the inversion of the autocorrelation matrix, but this need only be done once. In one approach to adaptive filtering, a large number of data samples are processed to obtain an accurate estimate of R_{xx} and P_{xd} , then R_{xx}^{-1} is found and W^* is computed and the data are filtered using W^* . This approach requires a large amount of data storage, imposes a significant processing delay and works well only for stationary signals. The LMS approach uses the gradient method presented above but to minimize the storage requirement, the gradient is estimated on the basis of no more data than are present in the filter itself. That is, at iteration j , we estimate $R_{xx} \approx X_j X_j^T$ and $P_{xd} \approx d_j X_j$. With this, the estimated gradient at iteration j becomes $\hat{\nabla}_j = -2e_j X_j$ and the weight update becomes

$$W_{j+1} = W_j + 2\mu e_j X_j \quad (4)$$

Admittedly, these estimates of the gradient are noisy, but if μ is chosen to be small, the error in each estimate will contribute little to the final solution.

To analyze the convergence characteristics of the LMS adaptive filter, consider an ensemble of adaptive processes that all begin with the same initial weight vector W_0 . Also, the inputs to each adaptive filter are drawn from the same statistical populations. If we take the expected value over the ensemble of the weight update (4), we have

$$E \langle W_{j+1} \rangle = E \langle W_j \rangle + 2\mu (P_{xd} - R_{xx} E \langle W_j \rangle) \quad (5)$$

provided that X_j and W_j for each adaptive process are uncorrelated. We first note, that *in the mean*, the gradient estimate equals the true gradient, so, if the adaptive filter converges, the weight vector converges to the Wiener solution. We can simplify (5) if we translate the weight vector coordinates so that the optimum weight vector in the new coordinate system is 0. That is, let the new weight coordinates be represented by the vector V where $V_j = W_j - W^*$. Letting W_j be $V_j + W^*$ and recognizing that $P_{xd} = R_{xx}W^*$, the above expression becomes

$$\begin{aligned} E \langle V_{j+1} \rangle &= (I - 2\mu R_{xx}) E \langle V_j \rangle \\ &= (I - 2\mu R_{xx})^{j+1} V_0 \end{aligned} \quad (6)$$

This is a geometric equation whose convergence depends on R_{xx} and μ . If we now rotate the weight space so that the axes fall along the principle axes of R_{xx} , the above equation will be transformed into a set of uncoupled scalar equations. Since R_{xx}

is symmetric, it can be orthogonally diagonalized as in $\Lambda = Q^{-1}R_{xx}Q$ where Λ is a diagonal matrix composed of the eigenvalues of R_{xx} , that is $\Lambda = \text{diag}[\lambda_0, \lambda_1, \dots, \lambda_{N-1}]$ and Q is the orthonormal modal matrix of R_{xx} , that is the columns of Q are the normalized eigenvectors of R , each of which is distinct and orthogonal to the other eigenvectors. Thus $QQ^T = I$ and $Q^{-1} = Q^T$. Now, substituting $R_{xx} = Q\Lambda Q^T$ into (6) and premultiplying both sides of the equation by Q^T gives

$$E \langle V'_j \rangle = (I - 2\mu\Lambda)^j V'_0 \quad (7)$$

where $V' = Q^T V$ is a rotation of the translated weight coordinates into the principle axes of R_{xx} . Now, because Λ is diagonal, the above equation decouples into a set of scalar equations

$$v'_{pj} = (1 - 2\mu\lambda_p)^j v'_{p0}$$

with v'_p representing the p th element of V' . We will refer to $\{v_p\}$ as *modes* of the adaptive filter. It is evident that for $E \langle V'_j \rangle$ and hence $E \langle W_j \rangle$ to converge, we must have

$$|1 - 2\mu\lambda_p| < 1 \quad \forall p$$

which will be satisfied if $0 < \mu < 1/\lambda_{max}$ since $0 < \lambda_p \quad \forall p$ because R_{xx} is positive definite. It is important to note that this only guarantees convergence *in the mean* over an ensemble of adaptive processes. Note here that choosing $\mu = 1/2\lambda_p$ will give the fastest convergence for mode p . In fact it should converge *in the mean* in one step. but unless $\lambda_p = \lambda_{max}$, other modes will not converge. μ must be typically be chosen $< 1/2\lambda_{max}$ because the gradient estimate is itself noisy. Often, the eigenvalues are not directly available. Since R_{xx} is positive definite, $\lambda_{max} < (\sum_{i=0}^{N-1} \lambda_i = \text{tr}R_{xx})$ and since $\text{tr}R_{xx}$ is just the average power in $\{x_n\}$, μ is conveniently chosen as $1/\text{tr}R_{xx}$. It should be clear that λ_{max} limits rate at which the filter can converge. The mode associated with λ_{max} converges the fastest and that the mode associated with λ_{min} will be the last to converge. That is, the overall convergence of the LMS algorithm is controlled by the spread in the eigenvalues of R_{xx} . This aspect has attracted a lot of attention among researchers attempting to speed up LMS convergence. If all the eigenvalues were equal, the LMS algorithm could be made to converge at the fastest possible rate, taking into account that the overall rate must still be relatively slow to compensate for the fact that we have only estimated the gradient.

Another related problem we encountered [13] has to do with the non-proportional convergence of the modes of the adaptive filter, particularly in non-stationary environments. The most highly correlated modes of R_{xx} will have the largest eigenvalues and these modes will be resolved first. Since the eigenvalues of R_{xx} are related to the power spectral density of X , this essentially means that the spectral components of X that have the most power will be resolved first. If the problem is non-stationary, the lower power components may never be resolved. This effectively alters the power spectral density of the output process in a way that may be undesirable. For example, consider processing speech for noise cancellation using the line enhancer configuration of the adaptive filter. The spectral components of speech having the highest power are those in the low frequencies, so these will be resolved first. The high frequencies may never be resolved. In this case the non-proportional convergence has effectively added a low-pass filter. This is very undesirable as a large percentage of the intelligibility of speech is carried in the higher frequencies.

One way to achieve both goals of faster and more proportional convergence is to normalize (7) by replacing μ with $\mu\Lambda^{-1}$. This amounts to replacing a scalar- μ by a vector or a matrix μ . If we do this, and follow our development back through the rotation and translation steps, we arrive at the update equation

$$W_{j+1} = W_j + 2\mu R_{xx}^{-1} e_j X_j \quad (8)$$

A number of algorithms have been developed using this approach which is essentially Newton's method for the minimization of a quadratic surface. Most of these are referred to as "self-orthogonalizing" adaptive filters and they attempt to estimate R_{xx}^{-1} at each iteration. [26,6][5,20] This results in a much larger computational load than might be desired. Not only is there the additional load of estimating R_{xx}^{-1} , but also an extra matrix multiply is required between it and the gradient estimate.

Another way to do the same thing, and the approach we pursue in this paper, is to transform the input data into the rotated space, perform the update and output computations in the rotated space and then inverse transform the output data. To see how this might work, let us premultiply both sides of (8) with Q^T . With $R_{xx}^{-1} = Q\Lambda^{-1}Q^T$, this becomes

$$W'_{j+1} = W'_j + 2\mu\Lambda^{-1} e_j X'_j \quad (9)$$

where $W'_j = Q^T W_j$ and $X'_j = Q^T X_j$ represent vectors in the rotated space. Because Λ is diagonal, we shall often refer to $\mu\Lambda^{-1}$ as a *vector- μ* . Note that each of the modes of the adaptive process decouple and we end up with N one-weight adaptive filters with the adaptive gain equal to μ/λ_p for the p th mode. This can also be seen in the fact that $\Lambda = E \langle X'_j X_j'^T \rangle$, so λ_p is just the power in the p th component of X'_j averaged over the iterations j . This approach is very attractive in that it requires no more computation than the usual LMS time-domain approach, yet it promises improved convergence rates as well as more proportional convergence. The only obstacle to be overcome is the problem of transforming X into the rotated space. We could attempt to estimate Q , but this is essentially the same as the Newton's methods discussed above. What is needed is an orthogonal transform that decouples the spectral components of X . Several candidates have been studied, the most prominent among them being the Discrete Fourier Transform (DFT). Efforts to improve convergence in this way developed synergistically with efforts to reduce the number of computations required in the adaptive filter by using the Fast Fourier Transform (FFT) to implement high speed convolution and correlation.

Yet another recent approach [30] uses a DCT to estimate $\{\lambda_p\}$, orders the set by magnitude and then assigns to μ a sequence of values related to the reciprocals of the ordered set $\{\lambda_p\}$. Effectively, once the mode associated with λ_{max} is converged, μ can be increased to speed the convergence of the mode associated with the next largest eigenvalue until it too is converged and so on.

2 The Block Adaptive Filter

A block adaptive filter using FFTs to perform fast convolution and correlation was proposed nearly simultaneously in 1980 by Clark et. al. [9,14,18] and Ferrara [10]. Clark also presented the effect of block processing on the convergence and misadjustment of the adaptive filter as opposed to point by point processing. Waltzman and

Schwartz [1,3] had earlier applied the use of the FFT to the automatic channel equalizer. Not only did they show that the filter could be run with fewer computations, but by adapting the weights in the frequency domain they were able to obtain tighter bounds on λ_{max} and λ_{min} and hence a more accurate setting for μ . The following discussion is based largely on the approach used by Clark et. al.

We note that the convolution $y_j = W^T X_j = X_j^T W$ can be written in matrix form as

$$\begin{bmatrix} y_j \\ y_{j+1} \\ y_{j+2} \\ y_{j+3} \\ \vdots \end{bmatrix} = \begin{bmatrix} X_j^T \\ X_{j+1}^T \\ X_{j+2}^T \\ X_{j+3}^T \\ \vdots \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \end{bmatrix} \quad (10)$$

or as

$$Y_l = X_l W \quad (11)$$

where Y_l is an L element vector $[y_{lL}, y_{(l+1)L}, \dots, y_{(l+1)L-1}]^T$ and X_l is an L by N matrix whose rows are the transposes of the vectors $X_j = [x_j, x_{j-1}, \dots, x_{j-(N-1)}]^T$ for $j = (lL, (lL+1), \dots, (l+1)L-1)$. We also similarly define desired and error vectors D_l and E_l so that $E_l = D_l - Y_l = D_l - X_l W$. Thus, instead of computing the error for every input point, we only do so once every L points. For this approach, we are interested in minimizing the block mean square error ξ_B .

$$\xi_B = E \left\langle \frac{1}{L} E_l^T E_l \right\rangle$$

Under the assumption that the inputs are stationary, it is easy to show that the block mean square error will be the same as the mean square error in the unblocked filter. Further, the Weiner optimum weights will be the same in both cases.

We also only update the weights once per block. Following the same approach as for the unblocked case, we approximate the the gradient of the ξ_B using only the information available at that iteration. That is $\xi_B \approx 1/L E_l^T E_l$. (We note that this is L times the information that was available to the unblocked filter so we suspect that this gradient estimate is not as noisy as the estimates generated by an unblocked filter.) This leads to a block weight update equation

$$W_{l+1} = W_l + \frac{2\mu_B}{L} X_l^T E_l \quad (12)$$

where μ_B is the block adaptive gain constant.

By a similar analysis as applied to the unblocked filter, it can be shown that convergence is guaranteed *in the mean* if $0 < \mu_B < 1/\lambda_{max}$ which is the same condition as for the unblocked case. However, we must set $\mu_B = L\mu$ for the blocked and unblocked filters to converge at the same rate. Depending on L and how much smaller than $1/2\lambda_{max}$ μ is chosen to compensate for the larger gradient estimate noise in the unblocked filter, this may or may not be possible. So the blocked filter may be constrained to converge more slowly. This is particularly true in the case of highly disparate eigenvalues. We need to remember here that this discussion applies to a scalar μ_B and that when we eventually introduce a vector- μ , the convergence of the blocked algorithm can be made faster than that of the unblocked filter. There is, however, another factor that may

limit how large L can be. In applications where $\{x_n\}$ is slowly non-stationary, L must be small enough so that over several successive L -point blocks (the number depending on the convergence rate) $\{x_n\}$ is approximately stationary. Otherwise, the filter would, at best, not properly track the non-stationarity or, at worst, become unstable.

Let us next consider how to apply the FFT to reduce the number of computations required in the block adaptive filter.

The N -point DFT of a sequence $x(n)$ is computed

$$\mathcal{X}(k) = \sum_{n=0}^{N-1} x(n)W_N^{nk}$$

where $W_N = e^{-j\frac{2\pi}{N}}$ and W_N^n for $n = 0, 1, 2, \dots, N-1$ are the N roots of unity. Similarly, the inverse transform is given

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} \mathcal{X}(k)W_N^{-nk}$$

These transformations can be written in matrix form

$$\mathcal{X} = \sqrt{N}\mathcal{F}X \quad \text{and} \quad X = \frac{1}{\sqrt{N}}\mathcal{F}^{*T}\mathcal{X}$$

where $\mathcal{X} = [\mathcal{X}(0), \mathcal{X}(1), \dots, \mathcal{X}(N-1)]^T$, $X = [x(0), x(1), \dots, x(N-1)]^T$, \mathcal{F} is a matrix whose (k, n) th element is W_N^{kn}/\sqrt{N} , and $*$ represents the complex conjugate and T represents the transpose operation. We note first of all that \mathcal{F} is symmetric and that $\mathcal{F}^* = \mathcal{F}^{-1}$ so that \mathcal{F} is a unitary transform. Also, it can be shown [25,2,21] that if C is a circulant matrix, \mathcal{F} will diagonalize it. Further, if the diagonalization is expressed

$$\Lambda_C = \mathcal{F}C\mathcal{F}^* \quad (13)$$

then the eigenvalues of C and the elements of Λ_C will be given by the DFT of the first column of C .

With this background, let us now illustrate how the DFT can be used to implement the convolution in (11). First we rewrite (11) as follows

$$\begin{bmatrix} \cdot \\ Y_l \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ X_l & \cdot \end{bmatrix} \begin{bmatrix} W_l \\ 0 \end{bmatrix} \quad (14)$$

where the \cdot 's represent arbitrary elements which do not affect the equation. More simply,

$$Y_l^a = X_l^a W^a$$

If X_l^a can be made circulant, then the DFT can be used to diagonalize it. This can be done by defining X_l^a to be an $M \times M$ circulant matrix whose first column is the M -point vector $[x_{lL-(N-1)}, \dots, x_{lL-1}, x_{lL}, x_{lL+1}, \dots, x_{(l+1)L-1}]$. Each successive column is formed by circularly down shifting the previous column by one sample. The M point augmented weight vector W^a formed by padding the N -point W with $N - M$ zeros. The result will be another augmented vector Y_l^a whose last L points is the vector Y_l . With $I = \mathcal{F}^*\mathcal{F}$, we can write

$$Y_l^a = \frac{1}{\sqrt{M}}\mathcal{F}^*\mathcal{F}X_l^a\mathcal{F}^*\sqrt{M}\mathcal{F}W^a$$

which, with the circularity of X_l^a , can be written

$$Y_l^a = DFT^{-1} \left\{ DFT([x_{lL-(N-1)}, \dots, x_{lL}, \dots, x_{(l+1)L-1}]) \otimes DFT(W^a) \right\} \quad (15)$$

where \otimes is a point-by-point multiply of two vectors. Consider the following example with $l = 0$.

$$\begin{bmatrix} \hat{y}_{1-N} \\ \hat{y}_{2-N} \\ \vdots \\ y_0 \\ y_1 \\ \vdots \\ y_{L-2} \\ y_{L-1} \end{bmatrix} = \begin{bmatrix} x_{1-N} & x_{L-1} & \cdots & x_{L-N+1} & x_{L-N} & \cdots & x_{3-N} & x_{2-N} \\ x_{2-N} & x_{1-N} & \cdots & x_{L-N+2} & x_{L-N+1} & \cdots & x_{4-N} & x_{3-N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_0 & x_{-1} & \cdots & x_{1-N} & x_{L-1} & \cdots & x_2 & x_1 \\ x_1 & x_0 & \cdots & x_{2-N} & x_{1-N} & \cdots & x_3 & x_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{L-2} & x_{L-3} & \cdots & x_{L-N-1} & x_{L-N-2} & \cdots & x_{1-N} & x_{L-1} \\ x_{L-1} & x_{L-2} & \cdots & x_{L-N} & x_{L-N-1} & \cdots & x_{2-N} & x_{1-N} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{N-1} \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \quad (16)$$

From the above, we can see that the last $L = M - (N - 1)$ points y_0 to y_{L-1} represent points from the linear convolution of $x(n)$ with W . The previous $N - 1$ points are incorrect and are discarded. The next segment of $x(n)$ to be processed will overlap the previous by $N - 1$ points, i.e., $[x_{L-N+1}, \dots, x_{L-1}, x_L, x_{L+1}, \dots, x_{2L-1}]$. This will produce another valid L points which are abutted with the previous set. This is the standard "overlap and save" approach to discrete convolution using the DFT.

Waltzman [1,3] and, later, Ferrara [10], observed that the update (12) involved a cross-correlation between X_l and E_l and that this also could be sped up by applying FFTs. To see how this is done we first observe that by taking the complex conjugate transpose of (13), we have

$$\Lambda_C^* = \mathcal{F} C^T \mathcal{F}^*$$

Now, as before, we augment the vectors in (12). We use the same circulant matrix X_l^a and augmented vector W^a . This requires the use of an augmented vector E_l^a , whose first $N - 1$ points are 0 and whose last L points form E_l . That is

$$\begin{bmatrix} W_{l+1} \\ \cdot \end{bmatrix} = \begin{bmatrix} W_l \\ \cdot \end{bmatrix} + 2 \frac{\mu_B}{L} \begin{bmatrix} \cdot & X_l^T \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} 0 \\ E_l \end{bmatrix} \quad (17)$$

or, simply,

$$W_{l+1}^a = W_l^a + 2 \frac{\mu_B}{L} X_l^{aT} E_l^a$$

This is fortunate because it gives us a consistent set of augmented vectors. W_l^a and X_l^a are the same for both the update and convolution equations. Also, $E_l^a = D_l^a - Y_l^a$, if we define an appropriate augmented vector for D_l . Next, with the fact that $\mathcal{F}^* \mathcal{F} = I$, the augmented update equation becomes

$$W_{l+1}^a = W_l^a + 2 \frac{\mu_B}{L} \frac{1}{\sqrt{M}} \mathcal{F}^* \mathcal{F} X_l^{aT} \mathcal{F}^* \sqrt{M} \mathcal{F} E_l^a$$

Since $\mathcal{F} X_l^{aT} \mathcal{F}^*$ is given by the complex conjugate of the DFT of the first column of X_l^a , the update equation ultimately becomes

$$W_{l+1}^a = W_l^a + 2 \frac{\mu_B}{L} DFT^{-1} \left\{ \left(DFT([x_{lL-(N-1)}, \dots, x_{lL}, \dots, x_{(l+1)L-1}]) \right)^* \otimes DFT(E_l^a) \right\} \quad (18)$$

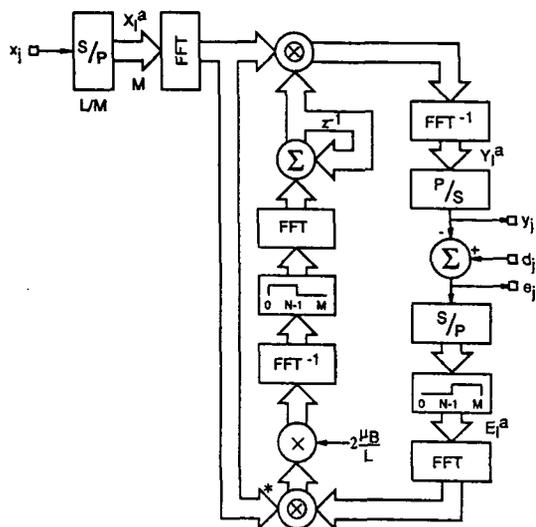


Figure 2: Another Version of The Block Frequency Domain Filter

N	Multiplies	Additions
8	0.967	3.313
16	0.820	2.250
32	0.580	1.391
64	0.374	0.820
128	0.228	0.471
256	0.134	0.265
512	0.077	0.147
1024	0.043	0.081

Table 1: Ratio of Frequency Domain to Time Domain Calculations

quires fewer approximations.

In order to formalize the sectioning and zero-padding operations we define two projection operators Π_N and Γ_L . Γ_L is constructed from an $M \times M$ identity matrix whose first $N - 1$ diagonal elements have been set to 0. Thus $\Gamma_L = \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix}$ where I is $L \times L$. Π_N is similarly constructed but in this case the first N diagonal elements are left at 1 with the remaining $L - 1$ elements set to 0. When Γ_L premultiplies a vector, it projects its first $N - 1$ points to 0. When premultiplying a matrix, it sets the first $N - 1$ rows of the matrix to 0. Postmultiplying a matrix by Γ_L is equivalent to setting the first $N - 1$ columns to zero. Π_N has similar properties but applies to the last $L - 1$ points of a vector or rows or columns of a matrix. With this (17) can be restated

$$W_{l+1}^a = W_l + 2\frac{\mu_B}{L}\Pi_N X_l^{aT} E_l^a \quad (19)$$

which formalizes the fact that the weight vector must be updated in such a way so that last $M - N$ elements of W_l^a are always 0.

Now, to consider the convergence of the augmented blocked filter, we take the expected value of (19) over an ensemble of adaptive processes just as we did in for the unblocked filter. This gives

$$E \langle W_{l+1}^a \rangle = E \langle W_l \rangle + 2\frac{\mu_B}{L}\Pi_N E \langle X_l^{aT} E_l^a \rangle$$

With $E_l^a = D_l^a - \Gamma_L X_l^a W_l^a$ and with the assumption of independence between X_l^a and W_l the above expands to

$$E \langle W_{l+1}^a \rangle = E \langle W_l \rangle + 2\frac{\mu_B}{L}\Pi_N \left(E \langle X_l^{aT} D_l^a \rangle - E \langle X_l^{aT} \Gamma_L X_l^a \rangle E \langle W_l \rangle \right)$$

Observing that

$$E \langle X_l^{aT} D_l^a \rangle = E \left\langle \begin{bmatrix} \cdot & X_l^T \\ \cdot & X_c^T \end{bmatrix} \begin{bmatrix} 0 \\ D_l \end{bmatrix} \right\rangle = L \begin{bmatrix} P_{xd} \\ 1/LE \langle X_c^T D_l \rangle \end{bmatrix} = LP_{xd}^a$$

and that

$$E \left\langle \begin{bmatrix} \cdot & X_l^T \\ \cdot & X_c^T \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \cdot & \cdot \\ X_l & X_c \end{bmatrix} \right\rangle = L \begin{bmatrix} R_{xx} & 1/LE \langle X_l^T X_c \rangle \\ 1/LE \langle X_c^T X_l \rangle & 1/LE \langle X_c^T X_c \rangle \end{bmatrix}$$

or, in augmented matrix notation,

$$E \langle X_l^{aT} \Gamma_L X_l^a \rangle = LR_{xx}^a$$

where we have chosen X_c to represent a part of the circulant matrix X_l^a . We can now write

$$E \langle W_{l+1}^a \rangle = E \langle W_l^a \rangle - 2\mu_B \Pi_N (R_{xx}^a E \langle W_l^a \rangle - P_{xd}^a) \quad (20)$$

This is just the blocked and augmented analog of (5) and the projection operator and the quantity in parentheses is the augmented form of the gradient (2). To find the optimum weight vector, we set the gradient to 0, but we note that this imposes no constraint on the augmented components of the gradient since Π_N already forces them

to zero. All that is required is $P_{xd} = R_{xx}W^*$. That is, the projection operator Π_N guarantees that W^{*a} is W^* (the Weiner optimum weight vector) padded with $L - 1$ zeros, which is precisely what we desire. If Π_N were not present, as is done in Mansour and Gray's unconstrained filter [15,16], setting the gradient to zero would also require that

$$E \langle X_c^T X_l \rangle W^* = E \langle X_c^T D_l \rangle \quad (21)$$

which is only true if $D_l = X_l W^* + \Psi$ where Ψ represents samples of a white gaussian noise sequence. This will be true if $\{x_j\}$ and $\{d_j\}$ are related by a simple FIR transfer function whose order G is less than or equal to N . Provided this is the case, we can delete Π_N which also allows us to drop two FFTs. The only drawback is that while the last $L - 1$ points of W^a start out as 0, gradient noise will allow small non-zero quantities to enter in which will lead to a slightly higher mean square error (MSE). When $M \geq G > N$, $N - G$ of the weights will tend toward non-zero values and the filter will be able to achieve a lower mean-square error than a constrained N weight filter. The lower MSE comes at the expense of introducing circular correlation products; however, if $N - G$ is small with respect to L this may be tolerable. The amount of error introduced has yet to be studied extensively. The same is true when the transfer function between $\{d_n\}$ and $\{x_n\}$ is IIR where the unconstrained filter will attempt to use all M weights. The unconstrained filter also fails when it is configured as a line enhancer where $x_j = z^{-\delta} d_j$. Because the filter is unconstrained it can find the non-causal solution for which the mean square error is zero - unfortunately, this is a useless solution.

While we could continue to pursue the mean convergence of augmented blocked algorithm, it is exactly the same as that of the blocked algorithm. What is important to note here is that, although the above expectations have explicitly determined the augmented portions of P_{xd}^a and R_{xx}^a , these portions contribute nothing to $E \langle W_{l+1}^a \rangle$ as a result of the projection Π_N and the zero padding of W_l^a (also due to Π_N). This will later allow us to augment R_{xx} in such a way that R_{xx}^a is easily invertible.

As we saw in Section 1, we can both speed up and orthogonalize the convergence of the LMS adaptive filter by replacing μ with μR_{xx}^{-1} . Writing (8) using augmented vectors consistent with those we have already defined gives

$$W_{l+1}^a = W_l^a + 2 \frac{\mu_B}{L} \Pi_N (R_{xx}^{-1})^a \Pi_N X_l^{aT} E_l^a$$

where $(R_{xx}^{-1})^a$ is an $M \times M$ augmented matrix whose upper left corner is composed of the $N \times N$ matrix R_{xx}^{-1} . The remaining elements of the matrix are arbitrary. Pre- and post-multiplying $(R_{xx}^{-1})^a$ by Π_N guarantees that they do not enter into the computation. Now, if we again take expectations over an ensemble, we will have

$$E \langle W_{l+1}^a \rangle = E \langle W_l \rangle - 2\mu_B \Pi_N (R_{xx}^{-1})^a \Pi_N (R_{xx}^a E \langle W_l \rangle - P_{xd}^a) \quad (22)$$

But,

$$\begin{aligned} \Pi_N (R_{xx}^{-1})^a \Pi_N R_{xx}^a &= \Pi_N \\ &= \Pi_N (R_{xx}^a)^{-1} R_{xx}^a \end{aligned}$$

and, while we have $\Pi_N P_{xd}^a = \Pi_N R_{xx}^a W^{*a}$, since the augmented portion of P_{xd}^a is not

constrained, we can let $P_{xd}^a = R_{xx}^a W^{*a}$. With this we have

$$\begin{aligned}\Pi_N(R_{xx}^{-1})^a \Pi_N P_{xd}^a &= \Pi_N(R_{xx}^{-1})^a \Pi_N R_{xx}^a W^{*a} \\ &= \Pi_N W^{*a} \\ &= \Pi_N(R_{xx}^a)^{-1} R_{xx}^a W^{*a} \\ &= \Pi_N(R_{xx}^a)^{-1} P_{xd}^a\end{aligned}$$

Thus, *in the mean*, $\Pi_N(R_{xx}^{-1})^a \Pi_N$ can be replaced by $\Pi_N(R_{xx}^a)^{-1}$ and the update equation becomes

$$W_{i+1}^a = W_i^a + 2 \frac{\mu B}{L} \Pi_N(R_{xx}^a)^{-1} X_i^{aT} E_i^a \quad (23)$$

Now, we pre-multiply with $\sqrt{M} \mathcal{F}$ and insert $\mathcal{F}^* \mathcal{F} = I$ at appropriate points. Thus,

$$W_{i+1}^a = W_i^a + \mathcal{F} \Pi_N \mathcal{F}^* 2 \frac{\mu B}{L} \mathcal{F}(R_{xx}^a)^{-1} \mathcal{F}^* \mathcal{F} X_i^{aT} \mathcal{F}^* E_i^a$$

where W and \mathcal{E} represent DFTs of W^a and E^a respectively. Next, if R_{xx}^a can be made circulant, \mathcal{F} will diagonalize it. That is $\Lambda^a = \mathcal{F} R_{xx}^a \mathcal{F}^*$ and $\Lambda^{a-1} = \mathcal{F}(R_{xx}^a)^{-1} \mathcal{F}^*$. With this the update equation becomes

$$W_{i+1}^a = W_i^a + \mathcal{F} \Pi_N \mathcal{F}^* 2 \frac{\mu B}{L} \Lambda^{a-1} \mathcal{F} X_i^{aT} \mathcal{F}^* E_i^a \quad (24)$$

This results in the structure shown in Figure 3. This filter [28] is exactly the same as the fast implementation of the block LMS adaptive filter shown in Figure 2, except μ

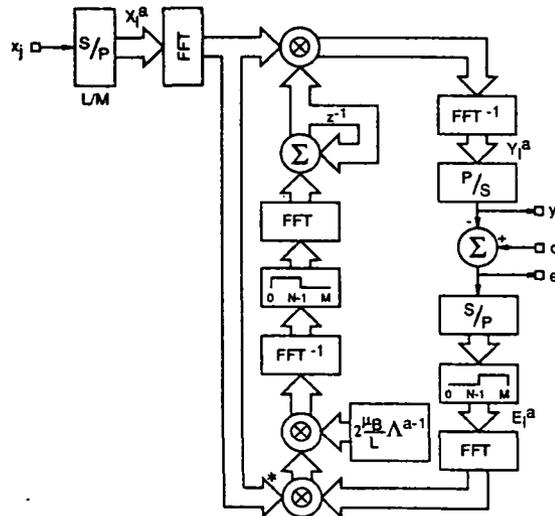


Figure 3: The Block Frequency Domain LMS Algorithm with a Vector μ

has been replaced by $\mu(\Lambda^a)^{-1}$. Since this simply represents a point-by-point multiply of two vectors, we see that this algorithm requires no more multiplies nor adds to implement than does Figure 2¹. The only additional computational load arises in estimating $(\Lambda^a)^{-1}$.

¹We need to note that Figure 2 requires N more real multiplies than did Figure 1.

As we indicated above, we must augment R_{xx} in such a way as to make R_{xx}^a circulant. If we let $r_n = r_{-n} = E\langle x_k x_{k+n} \rangle$ represent samples of the autocorrelation of $x(n)$, then

$$R_{xx} = \begin{bmatrix} r_0 & r_1 & r_2 & \cdots & r_{N-1} \\ r_1 & r_0 & r_1 & \cdots & r_{N-2} \\ r_2 & r_1 & r_0 & \cdots & r_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{N-1} & r_{N-2} & r_{N-3} & \cdots & r_0 \end{bmatrix}$$

Next, following the approach used by Picchi and Prati in [20], we define R_{xx}^a as a circulant matrix whose first column is given as

$$[r_0, r_1, r_2, \dots, r_{N-1}, r_{N-1}, r_{N-2}, \dots, r_2, r_1]$$

and whose successive columns are given by circularly down-shifting each preceding column by one sample. It is easy to see that the resulting $2N - 1 \times 2N - 1$ matrix contains the $N \times N$ matrix R_{xx} in its upper left corner. Defining R_{xx}^a this way imposes restrictions on L , namely we require $L = N$. L can be chosen greater than N by defining the first column of R_{xx}^a and inserting $L - N$ zeros as follows:

$$[r_0, r_1, r_2, \dots, r_{N-1}, 0, \dots, 0, r_{N-1}, r_{N-2}, \dots, r_2, r_1]$$

However, the only way L can be smaller than N is if $r_n = 0 \forall n \geq G$, for some $G \leq N$. In this case the first column is given as

$$[r_0, r_1, r_2, \dots, r_{G-1}, 0, \dots, 0, r_{G-1}, r_{G-2}, \dots, r_2, r_1]$$

In the above there are $G - N$ zeros, so the overall length is $M = N + G - 1$, and thus $L = G$.

Provided we can satisfy these restrictions on N and L we can easily diagonalize R_{xx}^a using the DFT. That is,

$$\lambda_i^a = \sum_{n=-\frac{M}{2}}^{\frac{M}{2}} r_n e^{-j\frac{2\pi ni}{M}} \quad (25)$$

where λ_i^a are just the diagonal elements of Λ^a . From this expression we see that λ_i^a are just samples of the power spectral density of $\{x_j\}$, $P_{xx}(\omega)$ convolved with the Fourier transform of a rectangular window of length M . Thus, the problem of estimating Λ^a is simply one of spectrum estimation.

A well known method for spectrum estimation is Bartlett's procedure [23]. Using this approach gives

$$\hat{\Lambda}^a = \text{diag} \left[\frac{1}{K+1} \sum_{l=0}^K \mathcal{X}_l^a \otimes \mathcal{X}_l^{a*} \right] \quad (26)$$

where \mathcal{X}_l^a is the DFT of the first column of X_l^a . That is, at block K , $\hat{\lambda}_i^a$ is just the average over $K + 1$ blocks of the power in the i th bin of the DFT of the reference input. $E\langle \hat{\lambda}_i^a \rangle$ is a sample of $P_{xx}(\omega)$ convolved with the Fourier transform of a triangular window of length M evaluated at $\omega = \frac{2\pi i}{M}$. Although this is not exactly equivalent to

(25), it provides a simple estimator. The estimator given in (26) was first proposed by Ogue et. al. in [19] and later by Picchi and Prati in [20] for use in an adaptive channel equalizer.

For a general adaptive filter, especially in the case of slowly non-stationary inputs, an exponential average might be more appropriate. That is,

$$\hat{\Lambda}_{l+1}^{\alpha} = \frac{1}{1+\beta} \hat{\Lambda}_l^{\alpha} + \frac{\beta}{1+\beta} \text{diag} [\mathcal{X}_l^{\alpha} \otimes \mathcal{X}_l^{\alpha*}] \quad (27)$$

where β is chosen to reflect the rate of non-stationarity. We need to be careful that any of the $\hat{\Lambda}_l^{\alpha}$ do not become too small or they will cause the digital filter to overflow when inverted. One way to prevent this is to constrain them to be larger than some lower limit. This is equivalent to adding a noise floor to x before the FFT is taken leading to the $\hat{\Lambda}$ estimate but it doesn't require as much computation.

In summary, we have found a way to use the FFT to easily estimate the eigenvalues of R_{xx} and used this to orthogonalize the adaptive process. If we had taken a purely heuristic approach, viewing the frequency-domain filter as a set of M independent one-weight adaptive filters operating on the bins of M -point DFTs of $x(n)$ and $d(n)$ we would have arrived at the same estimates as (26) and (27). That is, the adaptive gain of each filter is determined only by the power in the associated bin. Of course, adjacent bins of the DFT are not necessarily independent of each other. Hodgkiss and Nolte [4] argued that the covariance between two Fourier series coefficients of a process is approximately zero if the power spectral density of the process is relatively smooth. This argument can be extended to the variance between DFT coefficients, since these are just aliased Fourier series coefficients [24]. In this context, adjacent DFT coefficients are approximately independent of each other if the power spectral density changes slowly over the bandwidth of the DFT bins which will be the case if the process contains no isolated periodic components and we choose M large enough.

These arguments correspond to arguments made above. If N , and hence M , is greater than G , that is, if $r_n = 0 \forall n \geq G$, then the rectangular window disappears and the $\hat{\Lambda}_l^{\alpha}$ in (25) are exactly samples of $P_{xx}(\omega)$. Further, the difference between using a rectangular window and a triangular one diminishes as M increases and the estimate in (26) approaches that in (25).

We also note at this point that the filter in Figure 3 degenerates to Narayan and Peterson's Transform Domain filter (TDF) [11,17] if the overlap is set so that at each iteration only one output point is produced. That is $L = 1$ and $M = N$. In this case, the inverse DFT used to generate Y_l^{α} only produces one usable point. Since this one point is the first point in the vector, its computation only involves the average of the points in \mathcal{Y}_l , by definition of the DFT. Although the TDF requires more computation than the LMS adaptive filter, it has the advantage of proportional convergence. Narayan [17] also proposed other transforms including the DCT to be used in place of the DFT.

Another frequency-domain adaptive filter is the Dentino filter [8]. The Dentino filter simply implements (8) using the DFT to orthogonalize the input data. It ignores the fact that the DFT actually implements a circular convolution, so none of the vectors are augmented and there are no projection operations. Though the filter output suffers from circular convolution products, occasionally the meaningful information can be extracted from the converged weight vector. Maiwald et. al. [7] observed that for the

case where the input is truly cyclic with period equal to the length of the filter, as in an adaptive equalizer operating on a repeated training pulse, the frequency-domain weights of an M -point Dentino filter converge to the DFT of the optimal M -point transversal filter. After the weights have converged, they may be used to process data using standard overlap and-save-techniques. This works well in an equalizer where no output is generated during the training sequence. Another example of the Dentino filter used simply for the information in the weights is given by Reed et. al. [22], who use the filter to perform bearing estimation.

3 Proportional Convergence

In this section we present the results of two experiments that illustrate the proportional convergence of the general frequency-domain adaptive filter. In the first experiment we present a signal consisting of two sinusoids to a general 32-weight general frequency domain adaptive filter configured as an adaptive line enhancer. In this configuration, $x_j = z^{-\delta}d_j$. This configuration is commonly used to improve signal to noise ratios in noisy "single microphone" data. The filter will enhance input signals that have a strong autocorrelation, while rejecting those whose autocorrelation is narrower than δ . The first sinusoid has a frequency 0.25 times the sample frequency and the second has a frequency of 0.3125 times the sample frequency. This puts the first sinusoid in the 17th bin of a 64 point FFT, and the second in the 21st bin. The first sinusoid has an amplitude of 2.5 and the second 1.25 which gives a two to one relationship in amplitude. We also set the first 128 samples of the signal to zero. Thus we will have some zero output points before the sinusoids reach the filter. Finally, we analyze the output data by looking at the average magnitude in the two processed sinusoids as a function of output points. This is done for two cases. The first is the scalar- μ case, which is implemented with a vector- μ containing all ones. The second uses a vector- μ tailored to the power in the two sinusoids. It has the 17th and 21st points (as well as the symmetric points) in the vector- μ set to 0.000156 and 0.000626 respectively which is based on the inverse of the average power in these bins. All the other points in the vector- μ are set to 0.002. The power in these bins is actually very close to zero, which would ordinarily result in a much larger μ at these points, but this is undesirable since it may cause the filter to overflow. Instead we simply limit the minimum bin power. The result of these two experiments are shown in Figure 4. The two thin lines represent the scalar- μ case, while the two thicker ones represent the output of the vector- μ case. The μ was chosen for the vector- μ case so that the higher-power sinusoid was resolved at the same rate as in the scalar- μ case. We can easily see that in the scalar- μ case the lower-powered frequency was not resolved at the same rate as the higher-powered one. However, the vector- μ allows the filter to follow a much more proportional convergence.

For the second demonstration we use a frequency domain filter configured as a canceler. We apply speech spectrum noise to the reference (x_j) input and the same noise filtered through a 32 point FIR filter applied to the desired (d_j) input. This is a standard system identification problem. To minimize the mean square error, the adaptive filter weights will converge to the same values as the system filter. Using speech spectrum noise will give a disparate set of eigenvalues in R_{xx} . In this case we analyze both the average spectral energy in the error output as well as the mean square

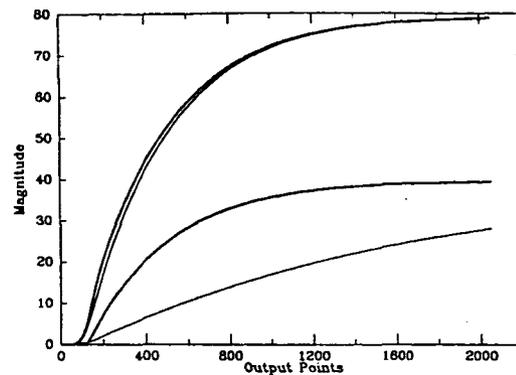
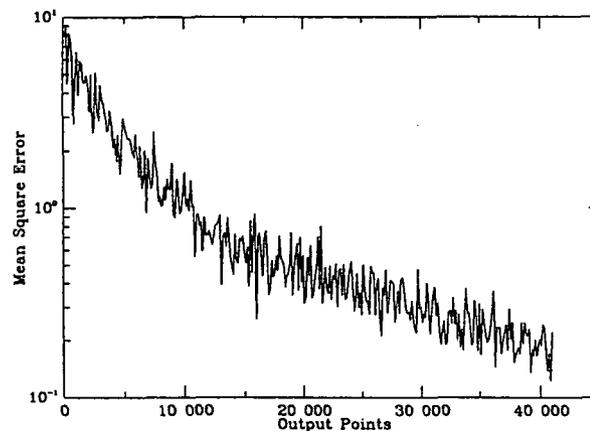


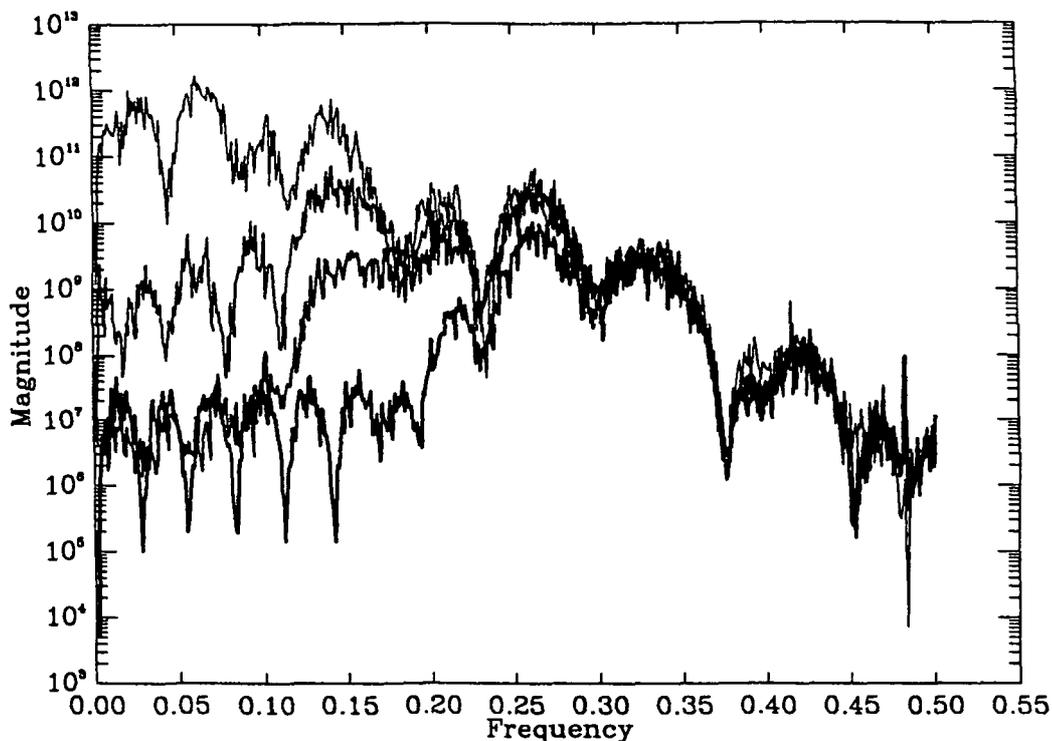
Figure 4: Output of a Two Sinusoid Enhancer

error. As before, we do this for the case of a scalar- μ , as well as a vector- μ tailored to the spectral energy in the reference input. This latter was chosen by inverting the average power in the bins of a 64-point FFT over several overlapping 32 point sections of the reference input. Analyzing the mean squared error in this case of a scalar- μ gives the result shown in Figure 5. This is probably more accurately described as a 'smoothed'

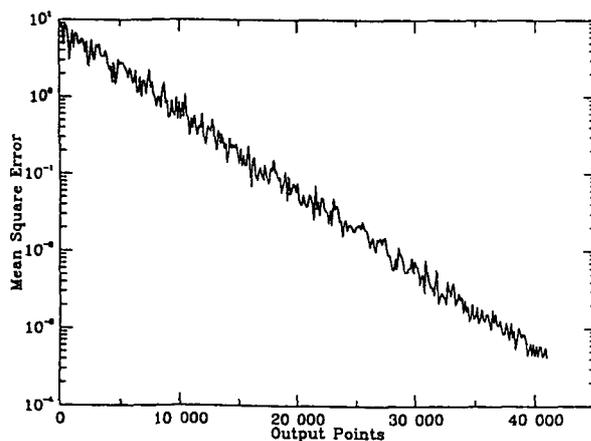
Figure 5: Mean Squared Error for a Scalar μ

square error curve since it is generated by squaring the error and then smoothing it by averaging each point with a few neighboring points. By plotting this against a logarithmic scale, we see at least two different rate of convergence. This illustrates the fact that the filter resolves the weights associated with the larger eigenvalues of R_{xx} first. Having only a scalar- μ limits the convergence rates of the weights associated with the smaller eigenvalues.

Figure 6 shows a series of 4096 point spectral averages of the error output of the filter. Each average is based on 5 4096 point FFTs. The thin line represents the spectral energy in the reference input and each successively thicker line represents averages starting at 12800, 44800, and 172800 points into the output file respectively. We should point out that the mean square error plot stops about where the third trace begins. If we were to continue the mean square error curve we should be able to identify several more progressively decreasing rates of convergence.

Figure 6: Scalar μ Processing

By comparison, the mean square error and spectral average plots for a vector- μ are given in Figures 7 and 8 respectively. In Figure 8 the averages are taken beginning

Figure 7: Mean Squared Error for a Vector μ

at 12800, 38400, and 128000 points into the output file respectively. We see from both these figures that the vector- μ gives us a much more proportional convergence.

Both these demonstrations dealt with stationary problems, and used pre-determined vector- μ s. For non-stationary problems the vector- μ could be estimated using an exponential average as in (27). Again, as we pointed out in the first demonstration,

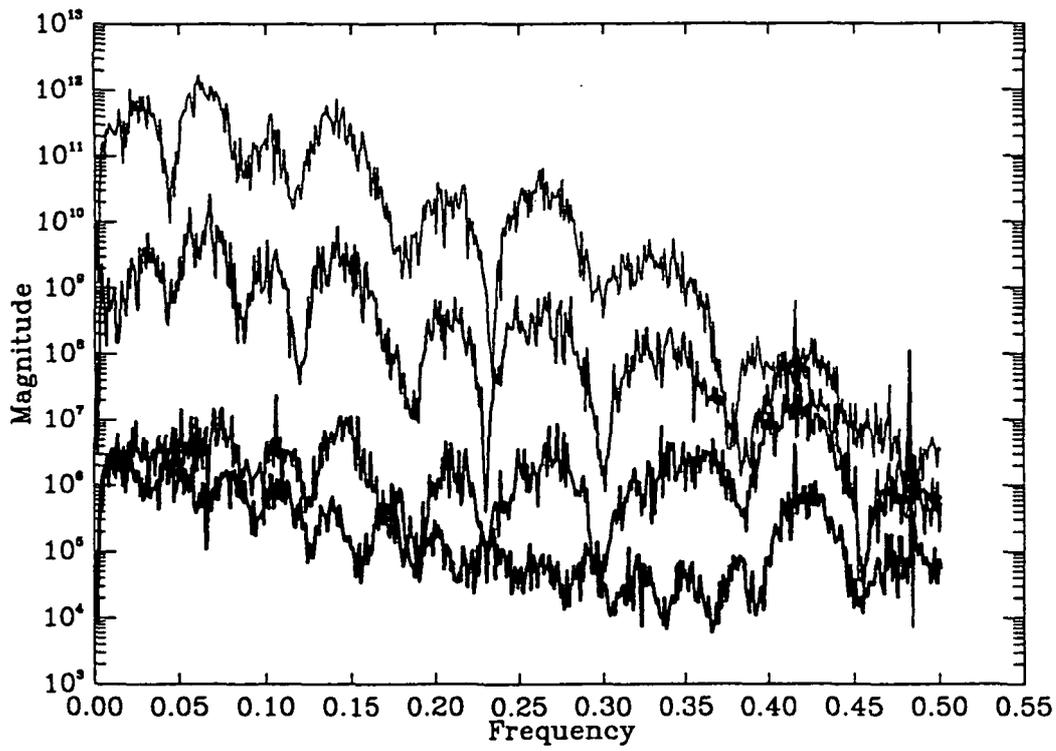


Figure 8: Vector μ Processing

occasionally we have to limit the largest values in the vector- μ , because even though the *average* power in those bins will be very small, a very large corresponding vector- μ element may cause the filter to overflow.

We also need to guard against choosing the block length too large since, as explained in Section 2, this may slow the overall convergence of the filter. Also, the filter will be less able to track any non-stationarities in the input. On the other hand, choosing longer FFTs allows us to better estimate Λ^a and provide more proportional convergence. One way to balance these two requirements is to increase the number of points overlapped in each section. This decreases the number of points output at each iteration and increases the computational overhead but provides the advantage of more proportional convergence.

4 A Frequency-Domain IIR Adaptive Filter

Next, we consider extending the frequency-domain techniques developed for the transversal adaptive filter to the recursive (or IIR) filter. This will include both the application of the DFT to perform fast block IIR convolution and the use of a vector- μ to make the adaptive process converge proportionally. First we present the LMS IIR adaptive filter as it was first developed in 1975 by White [31]. We will not concern ourselves here with questions of stability beyond those addressed by White.

In the recursive filter, the output is a weighted sum not only of the current and past inputs but also of past outputs.

$$y_j = \sum_{i=0}^{N_a-1} a_i x_{j-i} - \sum_{i=0}^{N_b-1} b_i y_{j-i-1} \quad (28)$$

The z -transform of the transfer function of the IIR filter is simply

$$\frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N_a-1} a_i z^{-i}}{1 + \sum_{i=0}^{N_b-1} b_i z^{-i-1}} \quad (29)$$

As with the transversal LMS adaptive filter, we form an error sequence $\{e_j\}$ using the difference between the output $\{y_j\}$ and a desired signal $\{d_j\}$. Then we use the expected value of the square of the error as a performance criteria upon which to base decisions about how to adjust the forward and backward weights, a_i and b_i respectively. Unfortunately, the recursive nature of the equations make analyzing $E\langle e_j^2 \rangle$ the way we did with the FIR filter nearly impossible. Instead we simply proceed with the implementation. As before, we approximate $E\langle e_j^2 \rangle$ with e_j^2 , then we find the gradient of the squared error with respect to the weights and use this to adjust the weights. First, we observe $\nabla(e_j^2) = 2e_j \nabla e_j$ and

$$\nabla e_j = \nabla(d_j - y_j) = - \left[\frac{\partial y_j}{\partial a_0}, \frac{\partial y_j}{\partial a_1}, \dots, \frac{\partial y_j}{\partial a_{N_a-1}}, \frac{\partial y_j}{\partial b_0}, \dots, \frac{\partial y_j}{\partial b_{N_b-1}} \right]^T$$

Now we need to find $\frac{\partial y_j}{\partial a_i}$ and $\frac{\partial y_j}{\partial b_i}$. But

$$\begin{aligned} \frac{\partial y_j}{\partial a_i} &= \frac{\partial}{\partial a_i} \left(\sum_{k=0}^{N_a-1} a_k x_{j-k} - \sum_{k=0}^{N_b-1} b_k y_{j-k-1} \right) \\ &= x_{j-i} - \sum_{k=0}^{N_b-1} b_k \frac{\partial y_{j-k-1}}{\partial a_i} \end{aligned} \quad (30)$$

and similarly,

$$\frac{\partial y_j}{\partial b_i} = -y_{j-i} - \sum_{k=0}^{N_b-1} b_k \frac{\partial y_{j-k-1}}{\partial b_i} \quad (31)$$

Thus the gradient estimate may itself be computed recursively. With suitable adaptive gains μ_f and μ_b , the weight update equations are written

$$a_{i,j+1} = a_{i,j} + 2\mu_f e_j \frac{\partial y_j}{\partial a_i} \quad (32)$$

$$b_{i,j+1} = b_{i,j} + 2\mu_b e_j \frac{\partial y_j}{\partial b_i} \quad (33)$$

We observe that we can reduce the number of computations required to implement a recursive filter simply by performing FFT implemented block convolution on the forward part of the filter and then computing the feedback contribution a sample at a time for each sample in the block. However, our object here is to find a way to bring to bear the power of the FFT in order to reduce the number of computations in the feedback portion of the filter as well as the feedforward. Several ways to perform recursive convolution in a block fashion have been developed [36,38,39,40,41]. We will use an approach first proposed by Voelcker and Hartquist [37], but we will follow a simpler development and make an additional observation. If we consider (28) it first appears that there is no obvious way to perform block recursive convolution when the block length L is greater than 1, since the feedback portion of the filter will require inputs before they have been computed. This is true unless the backward weights b_0, b_1, \dots, b_{L-1} (with $L < N_b$) are identically zero. It would seem that this requirement would limit the types of problems to which such a filter structure could be applied. However, we will show that it is possible to add extra poles to the transfer function in such a way as to set b_0 through b_{L-1} to zero. The effect of the extra poles can be compensated by adding equivalent zeros to the transfer function. To formalize these ideas, let us adopt a slightly different notation. First, we rewrite the filter equation as

$$y_j = \sum_{i=0}^{N_a-1} a_i x_{j-i} - \sum_{i=0}^{N_b-1} b_i y_{j-i-L} \quad (34)$$

where we now represent the first non-zero backward weight as b_0 . Then, the transfer function becomes

$$\frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{N_a-1} a_i z^{-i}}{1 + z^{-L} \sum_{i=0}^{N_b-1} b_i z^{-i}} \quad (35)$$

To implement the feedback portion of this filter using block methods, the largest block that can be computed at a time will be L points long. Suppose we want to replace a recursive filter with N_p forward weights ($p_0, p_1, \dots, p_{N_p-1}$) and N_q backward weights ($q_0, q_1, \dots, q_{N_q-1}$) with a block filter that has a feedback block length L . Both systems will have to be equivalent, that is

$$\frac{\sum_{i=0}^{N_p-1} p_i z^{-i}}{1 + z^{-1} \sum_{i=0}^{N_q-1} q_i z^{-i}} = \frac{\sum_{i=0}^{N_a-1} a_i z^{-i}}{1 + z^{-L} \sum_{i=0}^{N_b-1} b_i z^{-i}} \quad (36)$$

The order of the blocked denominator is $N_b + L - 1$, while that of the unblocked denominator is N_q , so we will have to add N_e poles and zeros to the unblocked system where

$$N_e = N_b + L - N_q - 1 \quad (37)$$

That is, we need to find ($e_0, e_1, \dots, e_{N_e-1}$) and ($b_0, b_1, \dots, b_{N_b-1}$) such that

$$\left(1 + z^{-1} \sum_{i=0}^{N_q-1} q_i z^{-i}\right) \left(1 + z^{-1} \sum_{i=0}^{N_e-1} e_i z^{-i}\right) = 1 + z^{-L} \sum_{i=0}^{N_b-1} b_i z^{-i} \quad (38)$$

Let us rewrite this equation as

$$(1 + z^{-1} Q(z)) E(z) = 1 + z^{-L} B(z) \quad (39)$$

where $Q(z) = \sum_{i=0}^{N_q-1} q_i z^{-i}$, $B(z) = \sum_{i=0}^{N_b-1} b_i z^{-i}$ and $E(z) = 1 + z^{-1} \sum_{i=0}^{N_e-1} e_i z^{-i}$. Then we write

$$E(z) = 1 + z^{-L} B(z) - z^{-1} Q(z) E(z) \quad (40)$$

If we recognize that (40) describes $E(z)$ as the output sequence of an all-pole recursive filter with input sequence $1 + z^{-L} B(z)$, we can see that the first L points of the sequence ($1, e_0, \dots, e_{N_e-1}$) correspond to the first L points of the impulse response of an all-pole filter with transfer function $H(z) = 1/(1 + z^{-1} Q(z))$, which is just the feedback portion of the filter we are trying to model. Since $Q(z)$ is general, we see that $N_e + 1 \geq L$. To keep $E(z)$ finite, we must choose $B(z)$ to clear the filter. Conceptually we can do this by setting b_j to the negative of the output of the filter e_{L+1+j} for $0 \leq j \leq N_q - 1$. This will introduce N_q zeros into the filter after which, provided there is no further input (ie: $b_j = 0 \forall j > N_q - 1$), the filter produces no further output. This would lead to $N_b = N_q$ and $N_e = L - 1$. We could let $N_e > L - 1$ by not choosing the $B(z)$ to immediately clear the filter. This gives us some flexibility in choosing the initial terms of $B(z)$.

We can also make these same observations by expanding the left side of (38) and

equate the coefficients of the two polynomials. In matrix notation this is

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ q_0 & 1 & 0 & \cdots & 0 \\ q_1 & q_0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{N_e-1} & q_{N_e-2} & q_{N_e-3} & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ q_{N_q-1} & q_{N_q-2} & q_{N_q-3} & \cdots & q_{N_q-N_e-2} \\ 0 & q_{N_q-1} & q_{N_q-2} & \cdots & q_{N_q-N_e-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & q_{N_q-1} \end{bmatrix} \begin{bmatrix} 1 \\ e_0 \\ e_1 \\ \vdots \\ e_{N_e-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ b_0 \\ b_1 \\ \vdots \\ b_{N_b-1} \end{bmatrix} \quad (41)$$

We note that the matrix is $N_q + N_e + 1$ by $N_e + 1$ and that the rightmost vector has $N_b + L$ elements which equals $N_q + N_e + 1$ using (37). Next, we observe that only the first $N_e + 1$ rows of the equation are needed to solve for $(e_0, e_1, \dots, e_{N_e-1})$. Also, since the determinant of the submatrix formed from the first $N_e + 1$ rows of the matrix is 1, a unique solution exists. If $L > N_e + 1$ however, rows $N_e + 2$ through row L of the matrix equation will not, in general, satisfy the equality. By using (37), this constraint can be written as $N_q > N_b$, that is, the equation is guaranteed to have a solution if $N_b \geq N_q$. If $N_b = N_q$, $(b_0, b_1, \dots, b_{N_b-1})$ are explicitly determined, but in the case that $N_b > N_q$, $(b_0, \dots, b_{N_b-N_q-1})$ can be chosen arbitrarily². Also notice from (41) that L can be chosen larger than N_q .

Thus we can determine the e_j either by using matrix methods on the first $N_e + 1$ rows of (41) or, as we observed above, the first N_e points of the impulse response of a recursive system whose feedforward weights are given by the first $N_e + 1$ elements of the solution vector and whose feedback weights are given by the first $N_e + 1$ elements of the first column of the matrix excluding the leading 1.³ The undetermined b terms are then found by performing the matrix operations indicated in the last N_q rows of (41).

Once the terms $(e_0, e_1, \dots, e_{N_e-1})$ have been determined, the block forward weights $(a_0, a_1, \dots, a_{N_a-1})$ can be found by adding an identical set of zeros to the system. That is,

$$\sum_{i=0}^{N_a-1} a_i z^{-i} = \left(\sum_{i=0}^{N_p-1} p_i z^{-i} \right) \left(1 + z^{-1} \sum_{i=0}^{N_e-1} e_i z^{-i} \right) \quad (42)$$

Hereafter, we will refer to $1 + z^{-1} \sum_{i=0}^{N_e-1} e_i z^{-i}$ as the auxiliary equation and its roots as auxiliaries. From this we see that the order of the feedforward portion of the block filter will have order $N_a - 1$ where $N_a = N_p + N_e$. With (37) we have $N_a = N_p + N_b - N_q + L - 1$. Although we can choose L to be smaller than N_b we will achieve greatest efficiency when $L \approx N_b$. In such a case, $N_a \approx 2N_b + N_p - N_q - 1$ and, with the constraint $N_b \geq N_q$, we require $N_a \approx \geq N_b + N_q - 1$. In most cases of interest, therefore, N_a will be larger than N_b . In the case that $N_p \approx N_q$ and, with the fact that implementing the convolution with FFTs requires that N_a and N_b must be powers of 2, then most likely

²Other considerations may determine how we select values for these terms.

³We note that these two approaches are mathematically equivalent.

we will need $N_a = 2N_b$. Often, it is undesirable to implement a filter that requires two different length FFTs. One approach is to use the larger length FFT for both convolutions but to use twice the amount of overlap in the feedback convolution as in the feedforward. A more efficient approach is based on the observation that a $2N$ -point convolution can be performed using two N -point convolutions. That is,

$$y_j = \sum_{i=0}^{N_a-1} a_i x_{j-i} = \sum_{i=0}^{N_a/2-1} a_i x_{j-i} + \sum_{i=N_a/2}^{N_a-1} a_i x_{j-i}$$

Notice that each convolution involves half of the weights and the input to the second convolution is just the input to the first convolution delayed by $N_a/2$. If $N_a/2$ is chosen to be equivalent to the block length L , then the input to the second convolution is the same as the input to the first except that it is delayed by one block. Thus, one FFT can be used for the input and, excluding the initial FFTs required for the weights, only three FFTs per block are required to implement a fixed weight recursive filter. The choice of $L = N_a/2$ forces us to use 50% overlap but, by saving an extra FFT, it is likely the most efficient approach. We also have assumed in the preceding argument that we want to use the same block length for both of the forward and the backward block convolutions. Although it is possible to have a different block length for each convolution, the use of two different block lengths increases the storage requirements of the filter as well as its complexity.

Presuming that we have chosen a suitable block length L which is larger than or equal to the larger of the required number of backward weights and half the required number of augmented forward weights, that is $L \geq N_b$ and $2L \geq N_a$, then we pad N_a to $2L$ and N_b to L by adding zeros to each. From these we form three L -point weight vectors

$$Aa_l = [a_0, a_1, \dots, a_{L-1}]^T, \quad Ab_l = [a_L, a_{L+1}, \dots, a_{2L-1}]^T, \quad \text{and} \quad B_l = [b_0, b_1, \dots, b_{L-1}]^T$$

Now defining the matrix X_l as we did for (11) (but with $N = L$) and the matrix Y_l the same way, we can write the block recursion as

$$\vec{Y}_l = X_l Aa_l + X_{l-1} Ab_l - Y_{l-1} B_l$$

where \vec{Y}_l is the l th L point output vector (defined as Y_l in (11)). Following the procedure developed in Section 2 we augment these vectors and matrices making Y_{l-1}^a and X_l^a circulant. This gives

$$\begin{bmatrix} \cdot \\ \vec{Y}_l \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ X_l & \cdot \end{bmatrix} \begin{bmatrix} Aa_l \\ 0 \end{bmatrix} + \begin{bmatrix} \cdot & \cdot \\ X_{l-1} & \cdot \end{bmatrix} \begin{bmatrix} Ab_l \\ 0 \end{bmatrix} - \begin{bmatrix} \cdot & \cdot \\ Y_{l-1} & \cdot \end{bmatrix} \begin{bmatrix} B_l \\ 0 \end{bmatrix} \quad (43)$$

or, simply,

$$\vec{Y}_l^a = X_l^a Aa_l^a + X_{l-1}^a Ab_l^a - Y_{l-1}^a B_l^a$$

Then, applying DFTs we have

$$\vec{Y}_l^a = DFT^{-1} \left\{ \begin{array}{l} DFT([x_{(l-1)L}, \dots, x_{(l+1)L-1}]) \otimes DFT(Aa_l^a) \\ + DFT([x_{(l-2)L}, \dots, x_{lL-1}]) \otimes DFT(Ab_l^a) \\ - DFT([y_{(l-2)L}, \dots, y_{lL-1}]) \otimes DFT(B_l^a) \end{array} \right\} \quad (44)$$

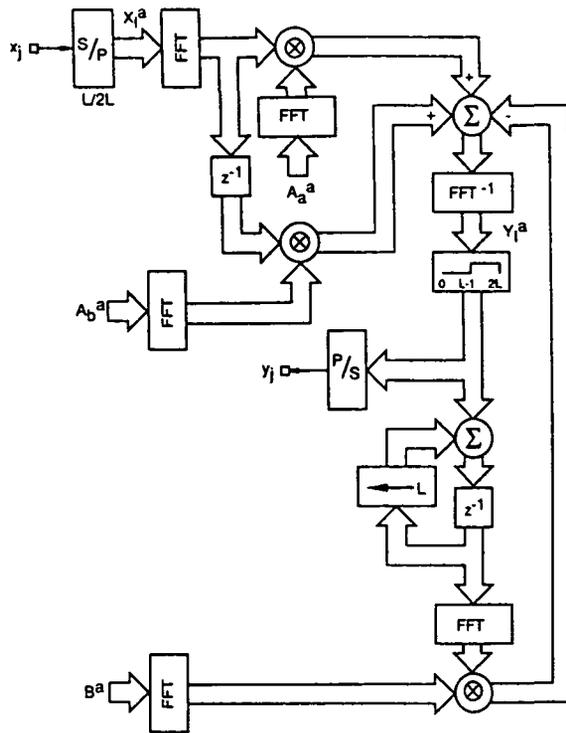


Figure 9: The Block Recursive Filter

which results in the structure diagrammed in Figure 9

Next, we need a method to adaptively update the coefficients of the block filter.

As we did for the block FIR filter, we first form an L -point error vector E_l from the difference of the l th output block \tilde{Y}_l and the l th set of L points from the desired input, D_l . Then we form the block mean square error and estimate it using only the information available at the l th iteration, yielding

$$\xi_B = E \left\langle \frac{1}{L} E_l^T E_l \right\rangle \approx \frac{1}{L} E_l^T E_l$$

and the block gradient estimate becomes

$$\hat{\nabla}_l = \frac{1}{L} \nabla E_l^T E_l = \frac{-2}{L} (\nabla \tilde{Y}_l)^T E_l \quad (45)$$

where

$$(\nabla \vec{Y}_i)^T = \begin{bmatrix} \left(\frac{\partial \vec{Y}_i}{\partial a_0}\right)^T \\ \left(\frac{\partial \vec{Y}_i}{\partial a_1}\right)^T \\ \vdots \\ \left(\frac{\partial \vec{Y}_i}{\partial a_{(i+2)L-1}}\right)^T \\ \left(\frac{\partial \vec{Y}_i}{\partial b_0}\right)^T \\ \left(\frac{\partial \vec{Y}_i}{\partial b_1}\right)^T \\ \vdots \\ \left(\frac{\partial \vec{Y}_i}{\partial b_{(i+1)L-1}}\right)^T \end{bmatrix} = \begin{bmatrix} \frac{\partial y_{iL}}{\partial a_0} & \frac{\partial y_{iL+1}}{\partial a_0} & \dots & \frac{\partial y_{(i+1)L-1}}{\partial a_0} \\ \frac{\partial y_{iL}}{\partial a_1} & \frac{\partial y_{iL+1}}{\partial a_1} & \dots & \frac{\partial y_{(i+1)L-1}}{\partial a_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{iL}}{\partial a_{(i+2)L-1}} & \frac{\partial y_{iL+1}}{\partial a_{(i+2)L-1}} & \dots & \frac{\partial y_{(i+1)L-1}}{\partial a_{(i+2)L-1}} \\ \frac{\partial y_{iL}}{\partial b_0} & \frac{\partial y_{iL+1}}{\partial b_0} & \dots & \frac{\partial y_{(i+1)L-1}}{\partial b_0} \\ \frac{\partial y_{iL}}{\partial b_1} & \frac{\partial y_{iL+1}}{\partial b_1} & \dots & \frac{\partial y_{(i+1)L-1}}{\partial b_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{iL}}{\partial b_{(i+1)L-1}} & \frac{\partial y_{iL+1}}{\partial b_{(i+1)L-1}} & \dots & \frac{\partial y_{(i+1)L-1}}{\partial b_{(i+1)L-1}} \end{bmatrix} \quad (46)$$

From (34) we can find

$$\frac{\partial y_j}{\partial a_i} = x_{j-i} - \sum_{k=0}^{L-1} b_k \frac{\partial y_{j-k-L}}{\partial a_i} \quad \text{and} \quad \frac{\partial y_j}{\partial b_i} = -y_{j-i-L} - \sum_{k=0}^{L-1} b_k \frac{\partial y_{j-k-L}}{\partial b_i} \quad (47)$$

As in the scalar case, (47) can be solved recursively. By comparing these equations to (34) and (35), we note that the recursions can be computed using block methods directly as diagrammed in Figure 10. Then we compute the gradient estimate and update

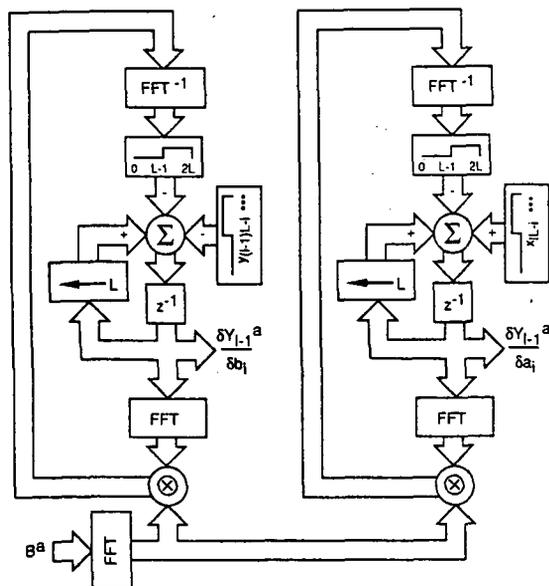


Figure 10: The Block Recursive Filter Weight Partials

the weights

$$a_{i+1} = a_i + \frac{2\mu_f}{L} \sum_{k=0}^{L-1} \frac{\partial y_{iL+k}}{\partial a_i} e_{iL+k} \quad b_{i+1} = b_i + \frac{2\mu_b}{L} \sum_{k=0}^{L-1} \frac{\partial y_{iL+k}}{\partial b_i} e_{iL+k} \quad (48)$$

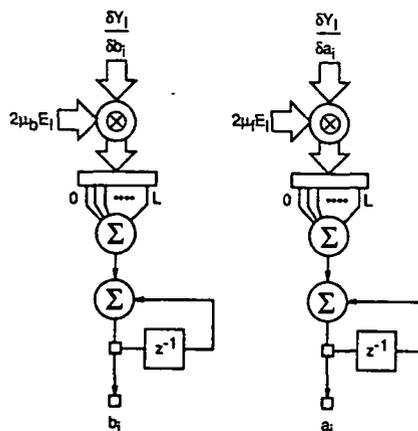


Figure 11: The Block Recursive Filter Weight Update

where μ is the adaptive gain. Thus, the weight update can be performed as shown in Figure 11.

At long last, we have a complete block IIR filter implemented using the FFT to perform fast convolution. To summarize, we present the complete algorithm. To do so, we define the vectors:

$$\begin{aligned}
 X_l^T &= [x_{lL}, x_{lL+1}, \dots, x_{(l+1)L-1}] \\
 D_l^T &= [d_{lL}, d_{lL+1}, \dots, d_{(l+1)L-1}] \\
 Y_l^T &= [y_{lL}, y_{lL+1}, \dots, y_{(l+1)L-1}] \\
 E_l^T &= [e_{lL}, e_{lL+1}, \dots, e_{(l+1)L-1}] \\
 Aa^T &= [a_0, a_1, \dots, a_{L-1}] \\
 Ab^T &= [a_L, a_{L+1}, \dots, a_{2L-1}] \\
 B^T &= [b_0, b_1, \dots, b_{L-1}]
 \end{aligned} \tag{49}$$

Also, we define Γ to be a projection operator that sets the first L points of a vector to zero. With the FFT and its inverse represented by \mathcal{F} and \mathcal{F}^{-1} , we first compute the feedforward contribution and subtract from that the feedback contribution computed at the previous iteration to generate L points of output.

$$\begin{aligned}
 \mathcal{X}_l^a &= \mathcal{F}[X_{l-1}^T \| X_l^T]^T \\
 \mathcal{Y}_l^a &= \mathcal{X}_l^a \otimes Aa^a + \mathcal{X}_{l-1}^a \otimes Ab^a - \mathcal{Y}_{l-1}^a \\
 [0^T \| Z_{a_i}^T]^T &= \Gamma \mathcal{F}^{-1} \mathcal{Y}_l^a \\
 E_l &= D_l - Y_l
 \end{aligned} \tag{50}$$

Next we update the forward weights

$$\left. \begin{aligned}
 G_{a_i, l}^a &= \mathcal{F}[G_{a_i, (l-1)}^T \| G_{a_i, l}^T]^T \\
 Z_{a_i}^a &= G_{a_i, l}^a \otimes B \\
 [0^T \| Z_{a_i}^T]^T &= \Gamma \mathcal{F}^{-1} Z_{a_i}^a \\
 G_{a_i, (l+1)} &= [x_{lL-i}, \dots, x_{(l+1)L-1-i}]^T - Z_{a_i} \\
 a_i &= a_i + \frac{2\mu_f}{L} E_l^T G_{a_i, (l+1)}
 \end{aligned} \right\} \forall i; 0 \leq i \leq 2L-1 \tag{51}$$

where $G_{a,i} \cong \frac{\partial Y_i}{\partial a_i}$ at each iteration and \mathcal{Z} and Z are intermediate results. We then update the backward weights

$$\left. \begin{aligned} G_{b,i}^a &= \mathcal{F}[G_{b_i(l-1)}^T \| G_{b_i}^T]^T \\ Z_{b_i}^a &= G_{b_i}^a \otimes B \\ [0^T \| Z_{b_i}^T]^T &= \Gamma \mathcal{F}^{-1} Z_{b_i}^a \\ G_{b_i(l+1)} &= -[y_{(l-1)L-i}, \dots, y_{lL-1-i}]^T - Z_{b_i} \\ b_i &= b_i + \frac{2\mu_b}{L} E_i^T G_{b_i(l+1)} \end{aligned} \right\} \forall i; 0 \leq i \leq L-1 \quad (52)$$

where $G_{b,i} \cong \frac{\partial Y_i}{\partial b_i}$ at each iteration. Finally, we transform the weights back into the frequency domain and compute the feedback contribution for the next iteration:

$$\left. \begin{aligned} Aa &= \mathcal{F}[Aa^T \| 0^T]^T \\ Ab &= \mathcal{F}[Ab^T \| 0^T]^T \\ B &= \mathcal{F}[B^T \| 0^T]^T \\ \mathcal{Y}_i^a &= \mathcal{F}[Y_{i-1}^T \| Y_i^T]^T \\ \mathcal{V}_i &= \mathcal{Y}_i^a \otimes B \end{aligned} \right\} \quad (53)$$

Equations (50), (51), (52) and (53) constitute a frequency domain IIR adaptive filter.

Now we turn to the question of efficiency. As with the FIR adaptive filter, we restrict our focus to filters whose inputs are real sequences.

With N_p forward and N_q backward weights, a time-domain IIR filter requires $N_p + N_q$ real multiplies and real additions including the error calculation. There are $N_p + N_q$ recursions required to compute the weight updates and each requires $N_q + 1$ multiplies and $N_q + 1$ additions including the sum for the actual weight update. Apart from the recursions we also need two multiplies to form $\mu_f e_j$ and $\mu_b e_j$. Altogether, the time-domain IIR filter requires $(N_p + N_q)(N_q + 2) + 2$ real multiplies and $(N_p + N_q)(N_q + 2)$ real additions for each output point. If we use $2L$ point FFTs in the frequency domain filter, the feedback filter actually implements L feedback weights and $2L$ feedforward weights. An equivalent time-domain filter would use the same number of feedback weights. From (37) using $N_q = L$ gives $N_p = L + 1$. We also need to consider that the frequency domain filter produces L output points at each iteration. To produce L output points the equivalent time-domain filter requires $2L^3 + 5L^2 + 4L$ real multiplies and $2L^3 + 5L^2 + 2L$ real additions.

The filter section of the frequency-domain filter (excluding the weight FFTs) uses 3 FFTs, 3 \otimes operations and one three input \sum operation. Using the same FFT complexity estimates as we used for the frequency-domain FIR filter, the IIR filter section uses $6L \log_2(L) - 2$ real multiplies and $9L \log_2(L) + 18L - 42$ real additions. There are $3L$ recursions of the type in Figure 10. Each uses two FFTs, one \otimes operation and one two input \sum operation. Here, the sum occurs in the time domain and only requires L real additions. So weight partials require $12L^2 \log_2(L) - 12L^2 - 6L$ real multiplies and $18L^2 \log_2(L) + 21L^2 - 78L$ real additions. There are also $3L$ weight recursions of the type shown in Figure 11. The \otimes operations involve L point real vectors in addition to the multiply by μ_b or μ_f , which can be performed after the \sum . Each recursion requires $L + 1$ multiplies and L additions for each of the $3L$ weights. Adding in the cost of the three weight FFTs gives the cost of the weight updates as $3L^2 + 6L \log_2(L) - 9L$ real multiplies and $3L^2 + 9L \log_2(L) + 6L - 36$ real additions. Thus, the overall cost of computing L output points using the block recursive LMS

adaptive filter is $12(L + L^2) \log_2(L) - 9L^2 - 15L - 2$ real multiplies and $24L^2 + 18(L + L^2) \log_2(L) + 90L - 78$ real additions.

Table 2 gives the ratio of frequency-domain to time-domain operations for several choices of L . By comparison, the ratios are a little worse than but still of the same

L	Multiplies	Additions
8	1.376	4.460
16	1.102	2.850
32	0.758	1.731
64	0.480	1.015
128	0.289	0.581
256	0.169	0.327
512	0.096	0.181
1024	0.054	0.099

Table 2: Ratio of Frequency Domain to Time Domain Calculations for the IIR Filters

order as those in Table 1. Here, we see that we need a filter length ≥ 64 before we realize a computational savings. While this filter length may be reasonable for a FIR adaptive filter, one of the reasons for adopting an IIR approach is the hope of shorter filters. However, the use of block methods may allow faster and more proportional convergence even in smaller filters, justifying the added computational expense.

4.1 Proportional Convergence

The forward and backward filters are not strictly independent from each other and we cannot find a simple non-recursive expression for the optimum weights as we did for the FIR adaptive filter. If we take a somewhat heuristic approach and treat the filters as independent from each other, we can apply some of the same ideas that lead to the use of a vector- μ . Following the same development that lead to (24)

$$\begin{aligned}
 Aa_{l+1}^a &= Aa_l^a + \mathcal{F}\Pi\mathcal{F}^* \frac{2\mu_f}{L} \Lambda_x^{a-1} \mathcal{F} \left(E_l^T \nabla_{Aa} \vec{Y}_l \right)^a \\
 Ab_{l+1}^a &= Ab_l^a + \mathcal{F}\Pi\mathcal{F}^* \frac{2\mu_f}{L} \Lambda_x^{a-1} \mathcal{F} \left(E_l^T \nabla_{Ab} \vec{Y}_l \right)^a \\
 B_{l+1}^a &= B_l^a + \mathcal{F}\Pi\mathcal{F}^* \frac{2\mu_b}{L} \Lambda_y^{a-1} \mathcal{F} \left(E_l^T \nabla_B \vec{Y}_l \right)^a
 \end{aligned} \tag{54}$$

where we have used $\nabla_{Aa} \vec{Y}_l$ to be the first L columns of $\nabla \vec{Y}_l$ and similarly for ∇_{Ab} and ∇_B . The notation ∇_W indicates the vector gradient with respect to each of the components of W . Π represents a projection operator that selects the first L points of a vector and the the final term in each of the above equations is augmented by padding each with L zeroes. Noting that $\nabla_{Aa} \equiv [G_{a_0(l+1)}, G_{a_1(l+1)}, \dots, G_{a_{(L-1)}(l+1)}]^T$, $\nabla_{Ab} \equiv [G_{a_L(l+1)}, G_{a_{(L+1)}(l+1)}, \dots, G_{a_{(2L-1)}(l+1)}]^T$, and $\nabla_B \equiv [G_{b_0(l+1)}, G_{b_1(l+1)}, \dots, G_{b_{(L-1)}(l+1)}]^T$,

we adjust the weight updates in (51) and (52) as follows

$$\left. \begin{aligned}
 h_{B_i} &= \frac{2\mu_b}{L} E_i^T G_{b_i(l+1)} \\
 h_{A_{\alpha_i}} &= \frac{2\mu_f}{L} E_i^T G_{a_i(l+1)} \\
 h_{A_{\beta_i}} &= \frac{2\mu_f}{L} E_i^T G_{a_{(i+L)}(l+1)}
 \end{aligned} \right\} \forall i; 0 \leq i \leq L-1$$

$$\begin{aligned}
 A_{\alpha_{i+1}}^a &= A_{\alpha_i}^a + \Pi \mathcal{F}^{-1} \Lambda_x^{\alpha^{-1}} \mathcal{F} [H_{A_{\alpha}}^T \| 0^T]^T \\
 A_{\beta_{i+1}}^a &= A_{\beta_i}^a + \Pi \mathcal{F}^{-1} \Lambda_x^{\alpha^{-1}} \mathcal{F} [H_{A_{\beta}}^T \| 0^T]^T \\
 B_{i+1}^a &= B_i^a + \Pi \mathcal{F}^{-1} \Lambda_y^{\alpha^{-1}} \mathcal{F} [H_B^T \| 0^T]^T
 \end{aligned} \tag{55}$$

where $H_{A_{\alpha}} = [h_{A_{\alpha_0}}, h_{A_{\alpha_1}}, \dots, h_{A_{\alpha_{L-1}}}]$ and $H_{A_{\beta}}$ and H_B are defined similarly. The Λ 's can be estimated as they were in either (26) or (27). An exponential average is preferred for Λ_y^{α} since it is difficult to predict beforehand the inputs to the backward filter. If we use exponential averages, then

$$\begin{aligned}
 \hat{\Lambda}_{x_{i+1}}^{\alpha} &= \frac{1}{1 + \beta_f} \hat{\Lambda}_{x_i}^{\alpha} + \frac{\beta_f}{1 + \beta_f} \text{diag} [\mathcal{X}_i^{\alpha} \otimes \mathcal{X}_i^{\alpha^*}] \\
 \hat{\Lambda}_{y_{i+1}}^{\alpha} &= \frac{1}{1 + \beta_b} \hat{\Lambda}_{y_i}^{\alpha} + \frac{\beta_b}{1 + \beta_b} \text{diag} [\mathcal{Y}_i^{\alpha} \otimes \mathcal{Y}_i^{\alpha^*}]
 \end{aligned} \tag{56}$$

Since the input to the backward filter is less predictable, one would usually choose β_b to be larger than β_f , and thus provide for a shorter window average. Notice that despite the fact that we have broken the forward convolution into two parts, we still use only one Λ_x^{α} . Even if we used separate estimators for both forward filters, they would be very similar since, except for a block delay, the inputs are the same to both forward filters. In Section 5 we present the results of a frequency domain IIR filter using both forward and backward vector- μ s. Using a vector- μ for the forward filter provides dramatic improvement in the case where the input to the filter is characterized by a large eigenvalue spread. The improvement resulting from using a vector- μ in the backward filter is more difficult to assess but it appears to be useful.

4.2 Feasibility

Although adding auxiliary poles and zeroes allowed us to do the block recursion, and although we showed that we could always find the auxiliary roots, we cannot guarantee that the auxiliary poles and zeros will always lie on top of each other since we separately adjust the forward and backward weights⁴. The plots in Section 5 indicate in fact that they do not begin to coincide until the filter is almost converged. If the auxiliary root happens to lie outside the unit circle and if the auxiliary pole follows a path (in time or space) different from the one the auxiliary zero takes when outside the unit circle, then the filter will become unstable. Another possibility is that the adaptive filter may prevent the pole from moving outside the unit circle if the error surface gradient estimate is steep enough in the direction associated with this motion. Though this avoids instability, it prevents the filter from converging to the optimal solution. We say a solution is *feasible* if the all the poles, including the auxiliaries lie inside the unit circle. Voelcker and Hartquist [37] showed that if L is chosen large enough, all the auxiliaries will fall inside the unit circle. We have noted above that choosing N_b larger than

⁴Even in a fixed filter, roundoff errors will cause the auxiliary poles and zeros not to coincide.

N_q allows $(b_0, \dots, b_{N_b - N_q - 1})$ to be chosen arbitrarily. We give an example to illustrate that this choice can also affect whether or not a solution is feasible. Suppose we choose $N_b = N_q = L = 3$, in particular a system with poles at $(-0.7, 0)$ and $(-0.5, \pm 0.5)$. The denominator of the system function is in this case $1 + 1.7z^{-1} + 1.2z^{-2} + .35z^{-3}$. Setting up (41) for this case

$$\begin{bmatrix} 1 & 0 \\ 1.7 & 1 \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \end{bmatrix} = \begin{bmatrix} -1.7 \\ -1.2 \end{bmatrix} \quad (57)$$

and solving for e_0 and e_1 gives that the auxiliary equation is $1 - 1.7z^{-1} + 1.69z^{-2}$ which has roots outside the unit circle at $(0.85, \pm 0.983616)$. If we set up the same problem, but with $N_b = 4$ we have

$$\begin{bmatrix} 1 & 0 & 0 \\ 1.7 & 1 & 0 \\ 1.2 & 1.7 & 1 \end{bmatrix} \begin{bmatrix} e_0 \\ e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} -1.7 \\ -1.2 \\ b_0 - .35 \end{bmatrix} \quad (58)$$

where we are free to choose b_0 . We notice that the choice of b_0 will only affect e_2 . e_0 and e_1 will always be -1.7 and 1.69 respectively. Choosing $e_2 = -0.75$ gives the auxiliary equation $1 - 1.7z^{-1} + 1.69z^{-2} - .75z^{-3}$ which has roots just inside the unit circle at $(0.464943, \pm 0.870465)$ and $(0.770114, 0)$. To get this value for e_2 requires that we set $b_0 = .433$.

Choosing N_b still larger the N_q gives more degrees of freedom (and also introduces more auxiliaries). Whether this can be done in such a way as to guarantee that all solutions for problems of a particular order will be feasible, or whether an adaptive filter left to choose these "free" feedback weights will always choose a feasible solution if one is available, are questions that require further research.

5 Frequency-Domain IIR Experiments

Here we present the results of an experiment to demonstrate both the frequency-domain IIR filter itself, and the application of proportional convergence to the filter.

In this set of experiments, we color a white noise source using a first order Butterworth filter designed with a cutoff of $0.1f_{sample}$ and implemented with an impulse invariant transform. This colored noise is then applied to a time-domain IIR filter. The output of this filter is applied to the desired input of a frequency-domain block IIR adaptive filter implemented with 8 forward and 4 backward weights using 16 point FFTs. The colored noise is also applied to the reference input of the frequency-domain filter making this a system identification problem and the time-domain IIR filter the system model. A random number generator forms the white noise source. The coloring filter has one forward and two backward weights which are $[0.253195]$ and $[-1.158046, 0.411241]$, respectively. The model filter has four forward and three backward weights which are $[1, -1, 1, -1]$ and $[-0.5, 0.25]$ respectively. In each experiment we have used $\mu = .002$ for both the forward and backward sections of the filter, and at each time we process 2000 eight point blocks, saving the weights every 40th block.

The filter will attempt to converge to the blocked version of the model weights. Here we have $N_p = 4$, $N_q = 3$, and $N_b = L = 4$. By (37), we have $N_e = 4$, so there will be 4 auxiliaries. Also, $N_b - N_q = 1$, so we are free to choose one of the auxiliaries. One

possible set is $\{(0.271844, 0), (-.385923, \pm.557571), (0, 0)\}$. The poles should converge to $\{(.5, \pm.5), (-.5, 0)\}$, while the zeros should converge to $\{(0, \pm 1), (1, 0)\}$.

There are three separate experiments, each using a different set of values for β_f and β_b , which are used to estimate the forward and backward vector- μ s as given in (56). If the $\hat{\Lambda}^a$ s are initialized to the identity matrix, setting $\beta = 0$ is equivalent to using a constant scalar- μ . In Case A, both β_f and β_b are set to 0. In Case B, we set β_f to 0.03 which gives an exponential average of about 17 blocks. Finally in Case C, we set both β_f and β_b to 0.03. Plots of the pole-zero tracks and final pole-zero positions after 2000 blocks for each case are shown in the following figures. We also plot the forward and backward weights after each 40 output blocks.

We see that Case C, with both a forward and backward vector- μ , converges in 2000 blocks. Case B, with only a forward vector- μ , also has the poles and zeroes converged, but the auxiliaries are still moving. Case A has one zero converged and some other poles and zeroes near convergence. The difference in the forward weight plots between Cases A and B illustrates how the vector- μ improves the convergence, making it more proportional. Comparing the backward weight plots for Cases B and C shows that a backward vector- μ may have caused some improvement, but it is difficult to judge. Still, it appears that the backward vector- μ improved the overall rate of convergence. This is an area where more experimentation needs to be done. As we indicated in Section 4, a larger β_b might be expected to perform better.

6 Summary

We have presented a general frequency-domain FIR adaptive filter that not only requires fewer computations than the time-domain LMS adaptive filter, but also provides faster and more proportional convergence which preserves the signal's spectral structure. We have also shown that the frequency-domain adaptive filters previously proposed by Dentino [8], Clark [9], Ferrara [10], Narayan and Peterson [11,17] and Mansour and Gray [15,16] are special cases of this general frequency-domain adaptive filter. We have extended these ideas to the recursive LMS adaptive filter and shown that it is possible to reduce the number of computations from order L^3 to order $L^2 \log_2(L)$. At the same time, we have presented a way to use the FFT to perform fast IIR convolution through the addition of auxiliary poles and zeros. We have shown that the convergence properties of the recursive LMS adaptive filter can be improved by using an appropriately chosen vector- μ . It is possible to show [29] that we can use these fast convolution techniques to reduce the number of computations in the CHARF [33], SHARF [34], and Feintuch's recursive LMS filter [32]. Both the general frequency-domain adaptive filter and the frequency-domain LMS recursive filter were implemented and tested. The results presented in this paper show the advantage of using a vector- μ .

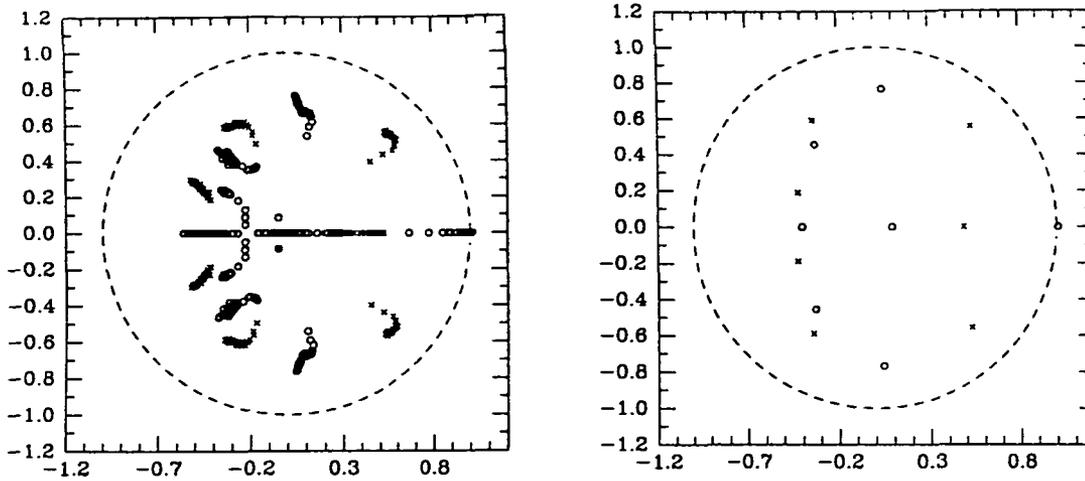


Figure 12: Pole Zero Tracks and Final Pole Zero Plot for Case A

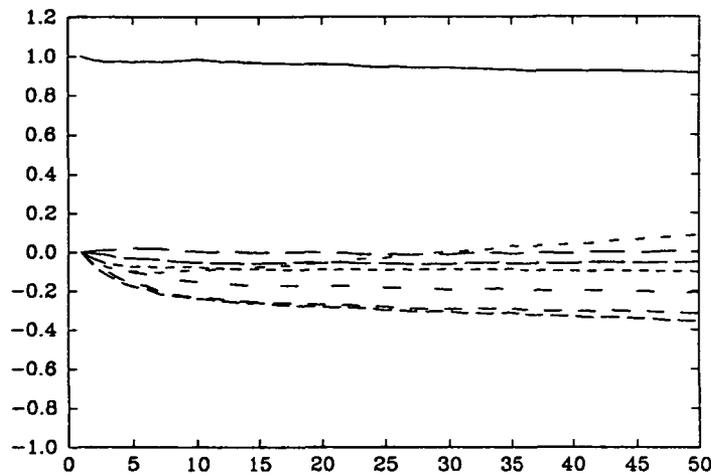


Figure 13: Forward weights for Case A

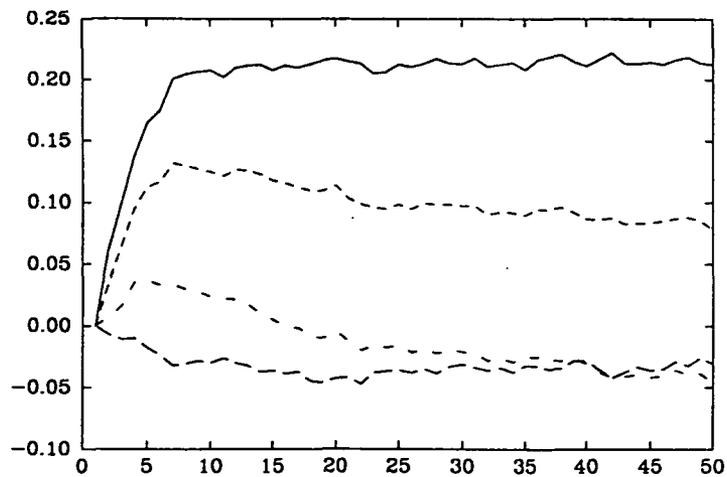


Figure 14: Backward Weights for Case A

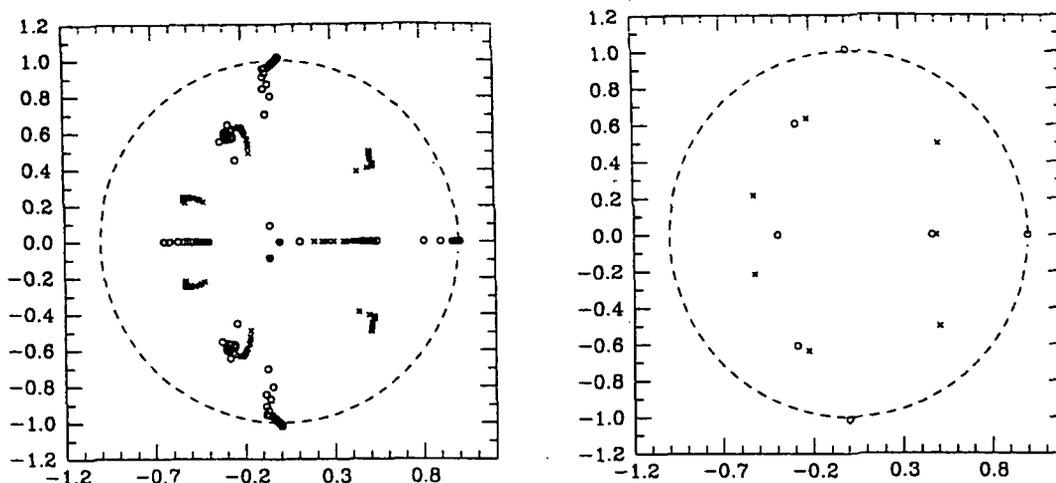


Figure 15: Pole Zero Tracks and Final Pole Zero Plot for Case B

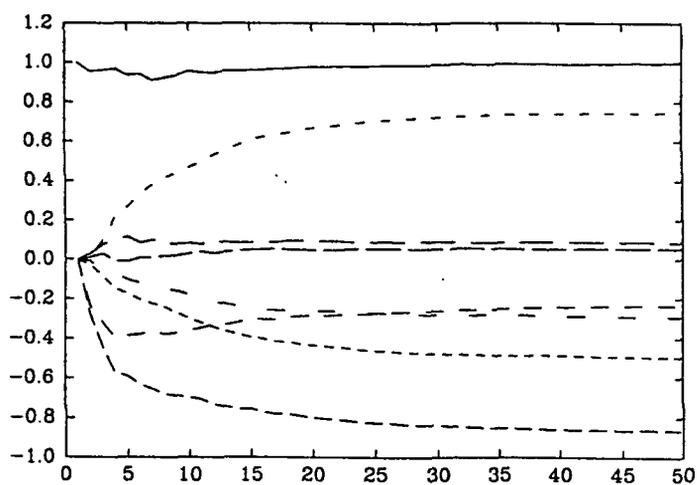


Figure 16: Forward weights for Case B

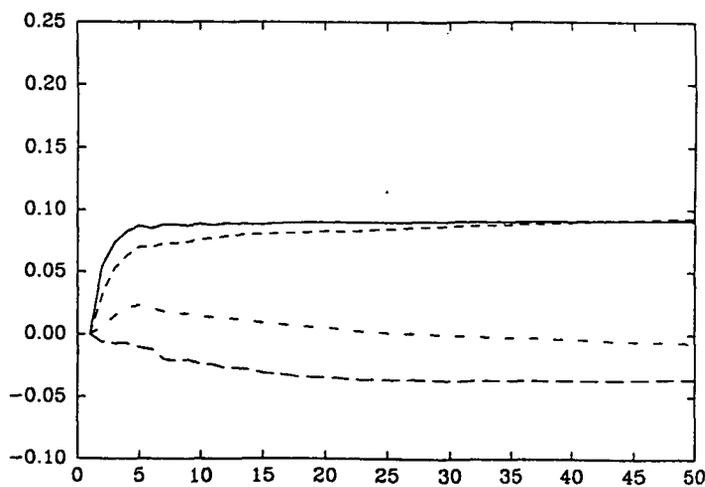


Figure 17: Backward Weights for Case B

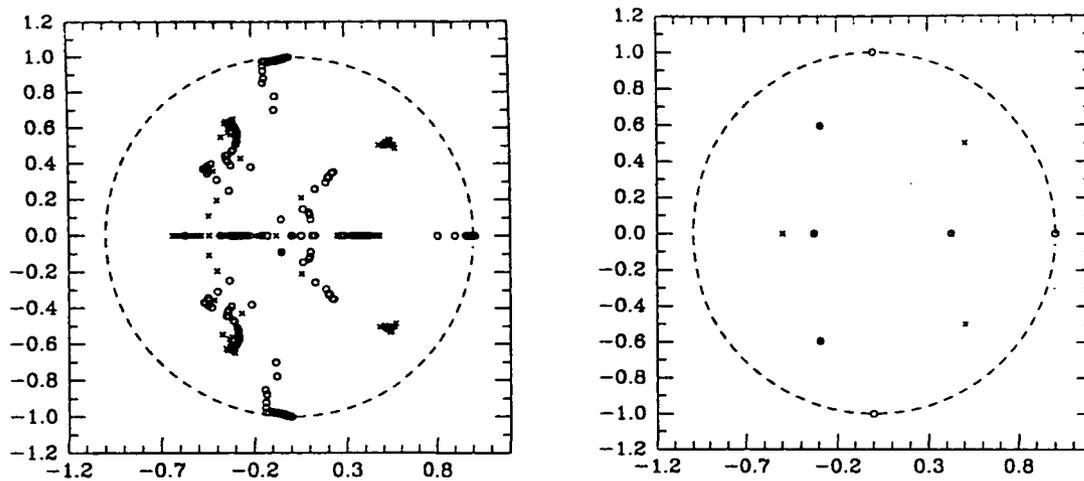


Figure 18: Pole Zero Tracks and Final Pole Zero Plot for Case C

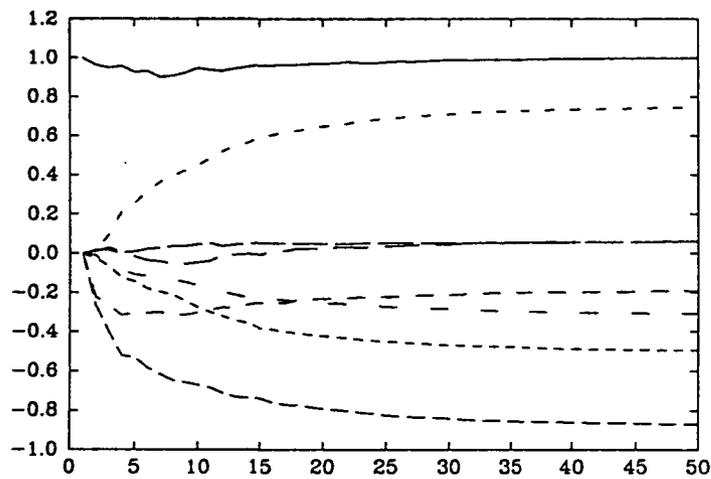


Figure 19: Forward weights for Case C

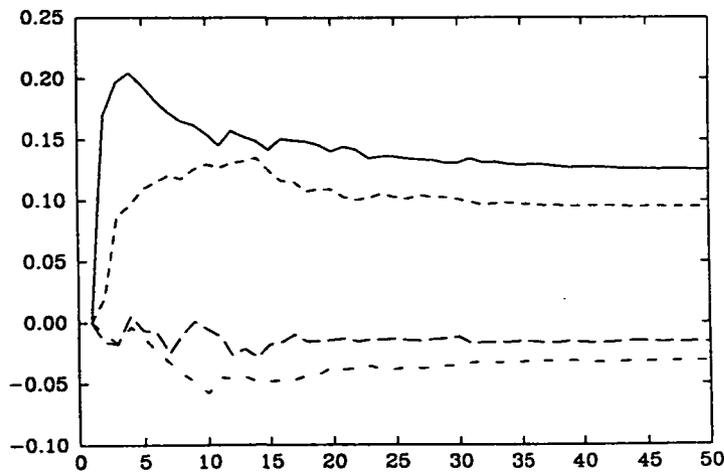


Figure 20: Backward Weights for Case C

References

- [1] T. Walzman. *Automatic Equalization and Discrete Fourier Transform Techniques*. PhD thesis, Polytechnic Institute of Brooklyn, June 1971.
- [2] R. M. Gray. On the asymptotic eigenvalue distribution of toeplitz matrices. *IEEE Transactions on Information Theory*, 725-730, November 1972.
- [3] T. Walzman and M. Schwartz. Automatic equalization using the discrete frequency domain. *IEEE Transactions on Information Theory*, 59-68, Jan 1973.
- [4] W. S. Hodgkiss and L. W. Nolte. Covariance between fourier coefficients representing the time waveforms observed from an array of sensors. *Journal of the Acoustical Society of America*, 582-590, March 1976.
- [5] B. Widrow, J. M. McCool, M. G. Larimore, and C R Johnson Jr. Stationary and nonstationary learning characteristics of the LMS adaptive filter. *Proceedings of the IEEE*, 1151-1162, Aug 1976.
- [6] R. D. Gitlin and F. R. Magee Jr. Self-orthogonalizing adaptive equalization algorithms. *IEEE Transactions on Communications*, 666-672, July 1977.
- [7] D. Maiwald, H. P. Kaeser, and F. Closs. *On Reducing the Number of Operations in Adaptive Equalizers*. Technical Report, IBM research report, Sept 1978.
- [8] M. J. Dentino, J. McCool, and B. Widrow. Adaptive filtering in the frequency domain. *Proceedings of the IEEE*, 1658-1659, Dec 1978.
- [9] G. A. Clark, S. K. Mitra, and S. R. Parker. Block adaptive filtering. *Proc IEEE Int Symp Circuits Syst*, 384-387, April 1980.
- [10] E. R. Ferrara. Fast implementation of LMS adaptive filters. *IEEE Transactions on Acoustics Speech and Signal Processing*, 474-475, Aug 1980.
- [11] S. S. Narayan and A. M. Peterson. Frequency domain least-mean-square algorithm. *Proceedings of the IEEE*, 124-126, Jan 1981.
- [12] G. A. Clark, S. K. Mitra, and S. R. Parker. Block implementation of adaptive digital filters. *IEEE Transactions on Acoustics Speech and Signal Processing*, 744-752, June 1981.
- [13] R. W. Christensen, D. M. Chabries, and D. W. Lynn. Noise reduction in speech using adaptive filtering. *Journal of the Acoustical Society of America*, S7-S8, April 1982. Supplement 1.
- [14] G. A. Clark, S. K. Mitra, and S. R. Parker. Efficient realization of adaptive digital filters in the time and frequency domains. *IEEE International Conference on Acoustics Speech and Signal Processing*, 1345-1348, May 1982.
- [15] D. Mansour and A. H. Gray Jr. Performance characteristics of the unconstrained frequency-domain adaptive filter. *Proc 1982 IEEE Int Symp Circuits and Systems*, 695-698, May 1982.
- [16] D. Mansour and A. H. Gray Jr. Unconstrained frequency domain adaptive filter. *IEEE Transactions on Acoustics Speech and Signal Processing*, 726-734, Oct 1982.
- [17] S. S. Narayan, A. M. Peterson, and M. J. Narisimha. Transform domain LMS algorithm. *IEEE Transactions on Acoustics Speech and Signal Processing*, 609-615, June 1983.

- [18] G. A. Clark, S. K. Mitra, and S. R. Parker. A unified approach to time and frequency domain realization of FIR adaptive digital filters. *IEEE Transactions on Acoustics Speech and Signal Processing*, 1073–1083, Oct 1983.
- [19] J. C. Ogue, T. Saito, and Y. Hoshiko. A fast convergence frequency domain adaptive filter. *IEEE Transactions on Acoustics Speech and Signal Processing*, 1312–1314, Oct 1983.
- [20] G. Picchi and G. Prati. Self-orthogonalizing adaptive equalization in the discrete frequency domain. *IEEE Transactions on Communications*, 371–379, April 1984.
- [21] J. C. Lee and C. K. Un. On the interrelationships among a class of convolutions. *IEEE Transactions on Acoustics Speech and Signal Processing*, 1245–1247, Dec 1984.
- [22] F. A. Reed, P. L. Feintuch, and N. J. Bershad. The application of the frequency domain LMS adaptive filter to split array bearing estimation with a sinusoidal signal. *IEEE Transactions on Acoustics Speech and Signal Processing*, 61–69, Feb 1985.
- [23] A. V. Oppenheim and R. W. Schaffer. *Digital Signal Processing*. Englewood Cliffs NJ: Prentice-Hall, 1975.
- [24] A. Papoulis. *Signal Analysis*. New York: McGraw-Hill, 1977.
- [25] H. C. Andrews and B. R. Hunt. *Digital Image Restoration*. Englewood Cliffs NJ: Prentice-Hall, 1977.
- [26] B. Widrow and S. D. Stearns. *Adaptive Signal Processing*. Englewood Cliffs NJ: Prentice-Hall, 1985.
- [27] E. R. Ferrara Jr. Frequency-domain adaptive filtering. In C F Cowan and P M Grant, editors, *Adaptive Filters*, Englewood Cliffs NJ: Prentice Hall, 1985.
- [28] D. M. Chabries, R. W. Christensen, D. W. Lynn, and G. M. Kenworthy. Adaptive noise suppressor. Oct 1985. Patent # 4,658,426.
- [29] D. W. Lynn *Frequency Domain FIR and IIR Adaptive Filters*. PhD thesis, Brigham Young University, 1988.
- [30] J. Chao, H. Perez and S. Tsujii. A Fast Adaptive Filter Algorithm Using Eigenvalue Reciprocals as Stepsizes. *IEEE Transactions on Acoustics Speech and Signal Processing*, 38-8:1343–1352, August 1990.
- [31] S. A. White. An adaptive recursive digital filter. *Proceedings of the 9th Asilomar IEEE Conference on Circuits, Systems and Computers*, 21–25, November 1972. Pacific Grove, CA.
- [32] P. L. Feintuch. An adaptive recursive LMS filter. *Proceedings of the IEEE*, 64-11:1622–1624, November 1976.
- [33] C. R. Johnson Jr. Charf convergence studies. *Proceedings of the 13th Asilomar IEEE Conference on Circuits, Systems and Computers*, 403–407, November 1979. Pacific Grove, CA.
- [34] M. G. Larimore, J. R. Treichler, and C. R. Johnson Jr. Sharf: an algorithm for adapting IIR digital filters. *IEEE Transactions on Acoustics Speech and Signal Processing*, 28-4:428–440, August 1980.

- [35] C. R. Johnson Jr. Adaptive IIR filtering: current results and open issues. *IEEE Transactions on Information Theory*, 30-2:237-250, March 1984.
- [36] B. Gold and K. L. Jordan Jr. A note on digital filter synthesis. *Proceedings of the IEEE*, 56-10:1717-1718, October 1968.
- [37] H. B. Voelcker Digital Filtering Via Block Recursion. *IEEE Transactions on Audio and Electroacoustics*, 18-2:169-176, June 1970.
- [38] C. S. Burrus. Block implementation of digital filters. *IEEE Transactions on Circuit Theory*, 18-6:697-701, November 1971.
- [39] C. S. Burrus. Block realization of digital filters. *IEEE Transactions on Audio and Electroacoustics*, 20-4:230-235, October 1972.
- [40] R. Gnanasekaran and S. K. Mitra. A note on block implementation of IIR digital filters. *Proceedings of the IEEE*, 65-7:1063-1064, July 1977.
- [41] S. K. Mitra and R. Gnanasekaran. Block implementation of recursive digital filters - new structures and properties. *IEEE Transactions on Circuits and Systems*, 25-4:200-207, April 1978.

Session 4
VLSI Design

Chairman: Kel Winters

NOVA A New Multi-Level Logic Simulator

L. Miles, P. Prins, K. Cameron
Microelectronics Research Center
University of Idaho
Moscow, Idaho 83843

J. Shovic
Advanced Hardware Architectures
Moscow, Idaho 83843

Abstract - This paper describes a new logic simulator that was developed at the NASA Space Engineering Research Center for VLSI Design. The simulator is multi-level, being able to simulate from the switch level through the functional model level. NOVA is currently in the Beta test phase and has been used to simulate chips designed for the NASA Space Station and the Explorer missions. A new algorithm was devised to simulate bi-directional pass transistors and a preliminary version of the algorithm is presented in this paper. This paper also describes the usage of functional models in NOVA and presents performance figures.

1 Introduction

Logic simulation is, arguably, one of the most important parts of the of the current VLSI design process. A fast, efficient, and accurate logic simulator can greatly enhance project turnaround time and personal productivity. The logic simulation system developed at the NASA Space Research Center for VLSI Systems Design located on the University of Idaho campus is designed to relieve the current logic simulation bottleneck. It is designed to be flexible and to meet the needs of VLSI designers for the near future.

NOVA is easily distinguished from other logic simulators currently available. First, it is fast; speed is gained by novel use of software data structures (which can be easily implemented in hardware) and by the development of a compiler (gibb) and flattener (glink) to take a hierarchical description of a circuit and compile it directly into a flat file for use by the simulator. NOVA also includes state models which incorporate most possible state values. It has been shown [2] that to completely describe a circuit with three distinct logic strengths, 21 states must be used; unlike other simulators, NOVA uses the required 21 states. This makes NOVA one of the most accurate simulators available. Third, NOVA is user-extendible. Not only can NOVA be made to interface with the most of the software that is currently available but even the internal primitives and functional models can be defined by the user to customize the simulator for the user's particular project. Lastly,

4.1.2

NOVA is very versatile; it can simulate from the switch level through the large functional model level.

In this paper, an overview of the key architecture features of the simulator is presented in Section 2. A discussion of logic states and a preliminary version of a new bidirectional pass transistor simulation algorithm is presented in Section 3. User-defined Functional models are discussed in Section 4. A comparison of NOVA versus the performance of other simulators is presented in Section 5 and the superiority of NOVA in terms of speed for equivalent accuracy is shown. Future uses and enhancements to NOVA are discussed in the final section.

2 Overview

In this section, the set of programs written for the NOVA system will be discussed, the use of busses and bus notation will be explained and the true multi-level nature of NOVA will be shown.

2.1 Multi-level Simulation

NOVA is a true multi-level simulator. The simulation engine and the circuit description language are robust enough to allow the user great flexibility.

NOVA can be used as a switch level simulator. In fact, it can be used as a switch level simulator with user-defined timing delays. This allows the designer to describe the circuit in terms of transistors alone and simulate the entire system.

NOVA also has as its primitives, most of the common gate structures used. Delays on these gates may be allowed to default or may be set individually (rise time and fall time may be different) for maximum flexibility. The circuit description language allows modules to be declared which can then be thought of as user-defined gates.

Functional models are supported in NOVA and are described in Section 4. Briefly, one may define anything as a functional model whether it be a chip, a board, a gate, or any arbitrary set of transistors and gates.

Hence, NOVA is flexible and can simulate using any mixture of levels of abstraction of the chip under design.

2.2 The Program Set

The NOVA system is composed of two basic subsystems. One is the system dealing with the circuit description language and the other deals with the simulation engine itself.

2.2.1 Circuit Description in NOVA

A circuit is described in the BOLT language. This is a hierarchical language and has a modularity which is similar to Pascal-like languages [1]. A program named *gibb* compiles the BOLT file into an intermediate format calling upon *bpp* to preprocess bus notation

into an expanded format, does the file inclusion, and performs string replacement. The intermediate format is then sent through *glink* which is the hierarchy primitive flattener. *glink* generates temporary files which are used by *greloc* to produce the flat file which is used by NOVA. A shell script is also created by *glink* if any functional models are present in the circuit description. This shell script directs the compilation of the user's 'C' code and links a program named *fmnova*.

2.2.2 The Simulator Programs

After successful circuit compilation, the simulator programs are used. The shell script *nova* is used to determine whether the program *fmnova* (circuit contains functional models) or the program *nfmnova* (no functional models) should be run [6]. Each of these programs runs the actual simulation.

2.3 Busses in NOVA

Special syntax allows a bus, a range of similarly named nodes, to be specified in a compact way in the BOLT language and in NOVA commands.

The syntax is fairly simple:

```
bus-name[index1:index2]
```

where index1 and index2 are non-negative numbers.

The bus is then expanded internally to:

```
bus-name[index1], ..., bus-name[index2]
```

As an example, a BOLT module for a 16 by 3 bit adder may be declared as:

```
sum[15:0] .adder16x3 a[15:0] b[15:0] c[15:0];
```

3 Logic STATES

The circuit node or wire may be divided into three types referred to as:

1. State information
2. Strength information
3. STATE (capitalized) information – both state and strength information collectively

State information is the the logical value of the node (wire). A binary system (0,1) requires three states to represent the logical state of a node:

4.1.4

- 1) 0
- 2) 1
- 3) X where X represents an unknown logic value, or a half level (neither 0 or 1).

Informally, the simulator requires the state information to calculate the output of the individual elements comprising the circuit. It is the State information regarding the inputs, of say, a NAND gate or an N-channel transistor, that is used to calculate what the output of the element would be if no other other circuit element were connected to it.

Strength information is used to resolve conflicts between multiple circuit elements that are connected to the same node. Thus, a classical logic simulator, which does not allow more than one circuit element capable of driving a logic state to be connected to a node, does not require any strength information at all. Modern logic circuits are rarely built solely out of the classical logic elements (NAND, NOR, INVERTER, AND, OR). Pass-transistor, tri-state, and pre-charged logic require a more sophisticated simulation scheme.

In general, different strengths may be assigned to the logic signals produced by a circuit element. If two or more elements are attached to a node, the node will assume the logic state with the greater strength. If a node has two circuit elements attached to it, both of the same strength but trying to output different logic values, then the node is evaluated to an X state.

3.1 Representing Indeterminate STATES

STATes representing either a 1 or a 0 must contain two pieces of strength information:

1. The greatest strength the signal may be driven to
2. The weakest strength it is possible for the signal to be

The first is significant because it sets a *lower bound* on the signal strength required to overdrive the signal. The weakest possible signal strength is important because it sets an *upper bound* on the strength of signal it can overdrive. In a case of contention between signals, the logic state goes to an X if neither signal can overdrive the other.

A logic system with the three basic strengths: Active (*a*), Resistive (*r*), and Floating (*f*) requires 12 distinct STATes to describe the zeros and the ones in the circuit. (Remember: **Active > Resistive > Floating.**)

0_{aa}	0_{rr}	0_{ff}	1_{aa}	1_{rr}	1_{ff}
	0_{ar}	0_{rf}		1_{ar}	1_{rf}
		0_{af}		1_{af}	

Table 1: Logic STATES Representing 0 or 1 in a Binary System With Three Basic Logic Strengths

In general if the system has N basic logic strengths represented by integers from 1 to N , these STATES may be represented by the triplet, K_{bd} , where:

K	represents the actual logic state (1 or 0);
b	represents the strongest possible strength of the signal and ranges from: $1 \leq b \leq N$;
d	indicates the weakest strength and ranges from $1 \leq d \leq b$.

The indeterminate, or X logic STATES also require two pieces of strength information:

1. the greatest strength of any circuit element connected to the node that may be attempting to drive it to a 0;
2. the greatest possible signal strength driving the node to a 1.

The first is necessary because it sets a *lower* bound on the strength of a signal capable of overdriving the node to a 1; and the second piece of strength information is used to set a *lower* bound on the strength of a signal that can overdrive the X to a 0.

The logic system with three basic strengths described above therefore needs 9 distinct logic STATES to represent the indeterministic or half-level logic conditions that may occur on a node.

X_{aa}	X_{ra}	X_{fa}
X_{ar}	X_{rr}	X_{fr}
X_{af}	X_{rf}	X_{ff}

Table 2: Logic STATES Representing Indeterministic Node Conditions in a System with Three Basic Logic Strengths

4.1.6

In general, if the system has N basic logic strengths that are represented by integers from 1 to N , these indeterministic STATES may be represented by the triplet, X_{pq} , where:

X	indicates the actual logic state is unknown or at half level;
p	represents strongest possible strength of any signal driving the node to a logical 0;
q	represents strongest possible strength of any signal driving the node to a logical 1;

Both p and q range from 1 to N .

The minimum number of internal logic STATES required to represent the information necessary to accurately simulate a binary system (0,1) with N basic logic strengths is seen to be:

$$\#STATES = N(2N + 1) \quad (1)$$

where: N^2 STATES represent unknown logic values and $(N^2 + 1)$ STATES represent the (0,1) STATES. For a system containing three basic strengths (active, resistive, floating) equation 1 gives a minimum of 21 STATES required for proper simulation.

3.2 The Number of Basic Logic Strengths

It should be noted here that adding NIL strength STATES corresponding to no connection (placed one step lower in the hierarchy of strengths than a floating strength), can simplify the formulation of the transistor models significantly, but is not, strictly speaking, necessary. The addition of a strength placed one level higher in the strength hierarchy than the floating strength can simplify the simulation of charge sharing effects and is appropriate when a large capacitance charge shares with a small capacitance. On the other hand, if depletion transistors, pullup resistors, and other overdrivable devices are not among the logic elements used in the circuit, then the resistive strength may be eliminated. The number of basic logic strengths may be extended to as many or as few as desired. (In the Table below, the alphanumeric representation is used for display only. The STATES are represented internally as in the first column.)

3.3 Simulation of Bi-directional Pass Transistors

Transistors are not inherently uni-directional, rather, they are bi-directional. In all but the most unusual cases, the designer can easily determine the source node and can, therefore, use the uni-directional model. For those instances when a bi-directional model of a transistor is required, an algorithm to correctly simulate the action of those nodes was developed. In the development, a couple of assumptions were made. First, a designer will not have long chains (greater than 8) of bi-directional pass transistors; due to the distributed RC time constants, the circuit slows down considerably when the chain becomes larger than 4 to 8 [5]. Second, the designer will not use the bi-directional model if a uni-directional

model will work as well; one must realize that a simulation of bi-directional transistors will slow down the simulation. The algorithm, then, can be a separate routine rather than a routine that is exercised every time the simulator is used.

3.3.1 Extensions to the basic States

As a signal propagates through a bi-directional pass transistor, its ability to overdrive another signal lessens, ie. the signal strength deteriorates slightly. To accommodate this within the framework of states that are already in place, the state strengths are given subscripts. The highest strength subscript is a 0 and the lowest strength is a 7. As an example of the use of the subscripts, let us suppose that a signal of state $1ar$ enters a chain of bi-directional pass transistors. After propagating through the first transistor, the state is $1a_0r_0$, after the second transistor, the state is $1a_1r_1$, and after the i^{th} transistor, the state is $1a_ir_i$.

3.3.2 Overdriving a State

One state may overdrive another state using the established rules of precedence as given in [2]. The substrengths 0 through 7 may be overdriven by any higher strength. This property is better shown by example:

- $1a_3r_3 + 1a_5r_5 \rightarrow 1a_3r_3$
- $0r_5r_5 + 1f_1f_1 \rightarrow 0r_5r_5$
- $1a_4a_4 + 0a_2a_2 \rightarrow Xa_2a_4$
- $Xr_3a_6 + 1a_7a_7 \rightarrow 1a_6a_6$

It is easily shown using the theory of Hamiltonian cycles, that the algorithm will converge to a proper state even with opposite ends of a chain being driven to different states.

One must be careful with regards to the timing algorithm. Through empirical study, it was decided to allow any transition to an X state happen in zero delay time but the transition from an X to a 1 or 0 state would happen in the normal delay time. Of course, a simple strength change within the same state happens with zero delay.

3.3.3 Interaction with Uni-directional Transistors and Gates

One can think of a group of bi-directional transistors as a cluster. A state comes into the cluster and is appended with the extra subscripts, ie. the state Spq becomes the state Sp_0q_0 . When the state finally leaves the cluster, the extra subscripts are simply dropped and the state takes on the original format.

4 Functional Modeling in NOVA

In today's complex design arena, a complete functional and behavioral language is a necessity for a complete logic simulation tool. Functional models are useful for modeling large blocks of circuitry. Examples of some blocks that one might want to create a functional model for include micro-processors, co-processors, and any number of large circuits that one might want to interface with. These models are also used during a design for top-level simulation and design and exercising gate/transistor level models. The NOVA functional modeling language is a super-set of 'C'. The user may write programs using I/O, sockets and other LAN access, math, graphics, etc.; the communications with the simulation engine is done through a set of carefully defined routines. The functional model language in NOVA is based on a set of 'C' language function calls. These calls organize the interaction between the NOVA engine and the functional models into an efficient model and provide integrity of the internal NOVA data structures.

There are two major parts to a functional model within NOVA.

1. the BOLT language module description
2. the 'C' language functional description

For a functional model to interface with other modules that may be functional models or may be transistor or gate level models requires information to be specified and passed to a functional model and back to the NOVA simulation. Following is a discussion of the major design choices to be made and the major paths of information flow from the user to the functional model and to the NOVA simulation.

4.1 Evaluation

The major means of passing information into the functional model from the rest of the NOVA simulation is through the input pins to the module block. This information can be used as input data, clocking information, and the forcing function for evaluating a functional model. The issue of when to evaluate a functional model can be a difficult decision. One must consider the architecture of the block that is being modeled (i.e. if the model only needs to be evaluated on a rising or falling clock edge) and weigh this versus having any change in any input evaluate the model. As NOVA is a timing simulator and all inputs do not change at exactly the same time, having the functional model evaluate on all changes to all inputs will cause the simulation to proceed at a much slower pace than is necessary. For this reason the execution speed tradeoff between building a functional model and using the gate and/or transistor level BOLT model may not be as obvious as initially thought. A clocked synchronous model will typically execute much faster than an asynchronous block such as a RAM of comparable size.

4.2 Local Storage

If a user wishes to only have one instantiation of a given functional model, creating local storage is simple. The user merely needs to declare the variables that he wishes to remain between calls to the functional model as static within the 'C' program. If there is to be more than one call to the functional model, then the user may create local storage which is allocated by the NOVA simulator on a per instantiation basis. One can allocate variable amounts of storage by using "malloc" and storing the pointer to the allocated structure in the NOVA local storage.

4.3 Communicating with NOVA

All the communication from the functional model to the NOVA simulation is done strictly through use of Output pins from the module. As this is how information is passed in a gate/transistor model, the functional model must utilize the pins to send information to the NOVA simulation.

4.4 External Communication

The functional model user is free to do anything that the 'C' language is capable of doing. An example might be to initialize a RAM or ROM model by reading a file or to prompt the user for information through a terminal. More elaborate uses might be building a "Virtual Logic Analyzer" by connecting a functional model through a graphical interface (such as X windows) allowing the user to view the simulation in a human oriented manner as it progresses and allowing the user to add asynchronous events if desired. NOVA has been specifically designed to allow functional models the full use of the UNIX operating system and the I/O available.

4.5 Scheduling Other Events

One problem with using the functional model system as presently constructed is the limitation of NOVA being able to only schedule one event on a node at a time. When constructing a functional model, one often knows that certain events will happen in the future at a given time. This state information must be stored by the functional model which can lead to some awkward coding practices. A set of functional models constructed at Advanced Hardware Architectures have implemented a method for handling this in a non-obtrusive manner and will be discussed in a future paper.

4.6 Other Uses For Functional Models

A number of additional uses for functional models (as opposed to modeling blocks that will be present in the design) have been developed. These include:

- a module for comparing differences in incoming vectors and documenting the time and variation

4.1.10

- a module describing a block and running in parallel to the gate/transistor model of the block and feeding into a comparison block
- analysis of the incoming vectors and providing a human readable analysis of the vectors to a file
- synthesizing complex input patterns that require feedback from the simulation to the pattern. The tool used by Advanced Hardware Architectures for pattern generation, PATGEN, lends itself well to these applications as it also is a super-set of 'C'.

5 Performance Issues

To a designer, time is of the essence. One would like to be able to perform as many simulations as possible in a given time frame both to decrease development time and to increase confidence in the chip being designed. As has already been discussed, logic simulation proceeds in two steps. First, the circuit description is compiled, then the circuit is simulated. For comparison purposes, the results of identical simulations using three other simulators have been compared to NOVA. All four simulators were tested using the Quick Simulator Benchmark [3] and were run on the HP9000/370 [4]. The only major difference between the runs was that NOVA was tested on a machine with 16MB of memory and the other three were tested on a machine with 32MB.

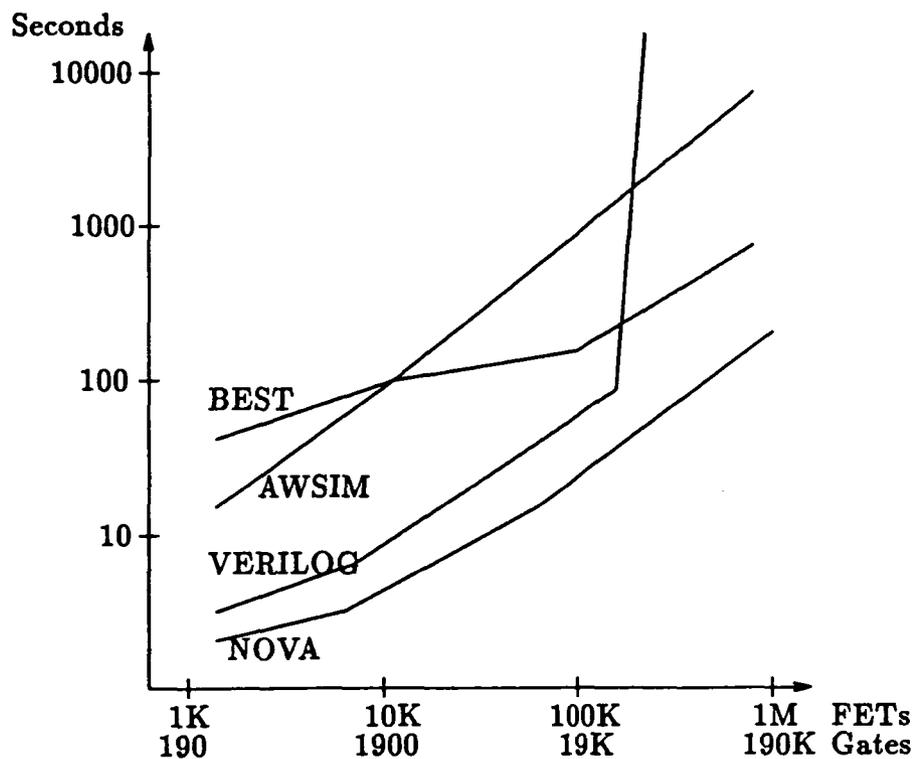


Figure 1: Compile Time Comparison

Figure 1 shows the compile time comparisons. As can be seen from the graph, NOVA (gibb) compiled circuits faster than AWSIM, BEST and VERILOG. It is interesting to note that VERILOG hit a physical memory limit at about 60K gates. This is because VERILOG uses about 400 bytes of memory per gate (whereas NOVA uses 32 bytes).

Simulation time is compared in Figure 2. The results are somewhat deceiving. None of the other simulators have as many features as NOVA so it is hard to compare. AWSIM is a zero delay simulator; therefore, it should be simulating much faster as propagation delay is not taken into account. BEST uses only 11 states and should be faster than NOVA but it isn't. VERILOG is faster than NOVA for the circuits that it can simulate. It should be noted that VERILOG cannot simulate as many gates as can NOVA due to memory limitations. VERILOG also cannot simulate at the transistor level. Neither BEST nor VERILOG can handle arbitrary delays; hence, they are tuned to run at a certain delay speed.

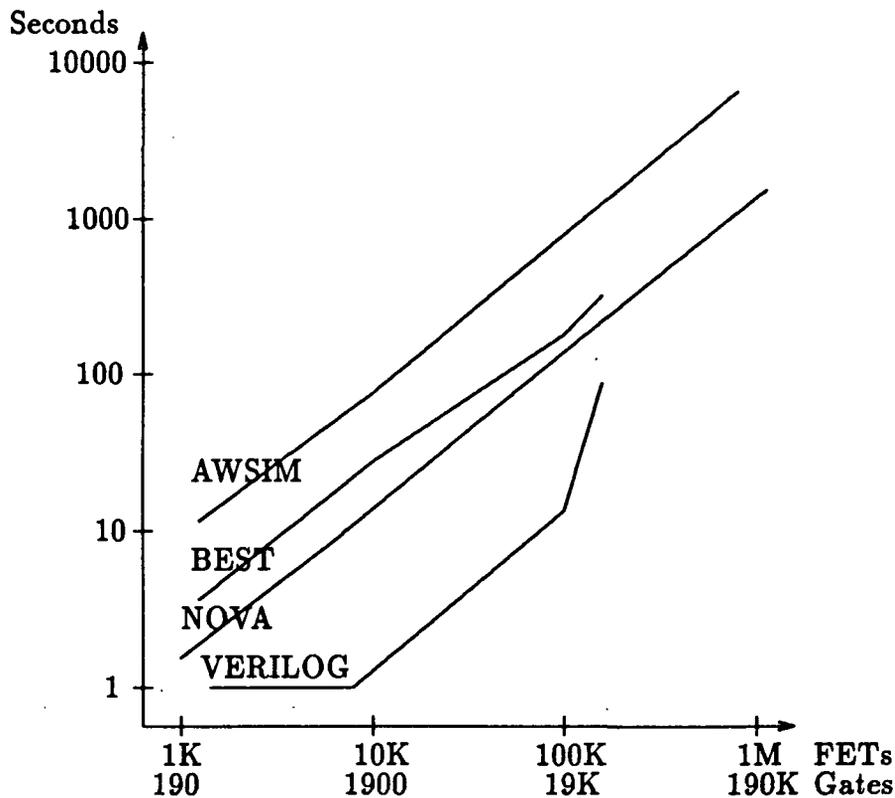


Figure 2: Simulation Time Comparison

Lastly, the relative performance (and portability) of NOVA is shown in Table 3. Using the HP9000/340 as a base for comparison, a suite was run to determine the relative simulation time. It is interesting to note that neither the size of the circuit nor the length of the simulation affected the relative simulation time appreciably. All four systems in the table are UNIX systems and it is the feeling of the ones porting the code, that NOVA could easily be made to run on any UNIX based machine. It might be noted that the DN10000

Type of System	Relative Speed
HP9000/340	1.0
HP9000/375	3.02
Apollo DN10000	5.08
Cray X-MP	6.56

Table 3: Relative Performance on Various Computers

is a two processor machine for which there is a very efficient 'C' compiler. The relatively low performance of the Cray X-MP can be partially explained because only about 6% of the code in the simulator could be vectorized. Much of the code is involved in pointer manipulation and in integer arithmetic which are the most inefficient operations on the Cray.

6 Future Uses and Enhancements to NOVA

Future enhancements to NOVA are currently focusing in two areas. One is the improvement in functional models and the other is coupling the I/O with the X-window system.

The industry seems to be standardizing upon two languages for describing behavioral and functional blocks. These languages are VHDL and VERILOG. A compiler can be designed that will translate these languages into the appropriate 'C' code and then compiled into the NOVA system for efficient execution and close tying of the functional blocks to the extant gate/transistor models. This allows a Top-down design with the same simulation tools being used throughout the design.

An X-window interface is currently being implemented and tested. The ability to see transitions graphically will greatly enhance NOVA's usefulness.

References

- [1] *BOLT/gibb User's Manual*, Idaho Research Foundation, 1990.
- [2] Cameron, K. B. and Shovic, J.C., "Calculating Minimum Logic State Requirements for Multi-Strength Multi-Value MOS Logic Simulators", *Proceedings of the 1987 IEEE International Conference on Computer Design: VLSI in Computers*, pp. 672-675; 1987.
- [3] Greer, D.L., "The Quick Simulator Benchmark", *VLSI Systems Design*, pp. 40-46; November 1987.
- [4] Heikes, C. and Koerner, C., "Logic Simulator Investigation Results," *Internal Hewlett-Packard Report*, 1989.

[5] Mead, C. and Conway, L. *Introduction to VLSI Systems*, Addison-Wesley, 1980.

[6] *NOVA User's Manual*, Idaho Research Foundation, 1990.

Path Programable Logic:

A Structured Design Method for Digital and/or Mixed Analog Integrated Circuits

B. Taylor

Bonneville Microelectronics Inc.
1399 S. 700 E. Suite 10
Salt Lake City, Utah 84105

Abstract-

The design of Integrated Circuits has evolved past the black art practiced by a few semiconductor companies to a world wide community of users. This was basically accomplished by the development of computer aided design tools which were made available to this community. As the tools matured into different components of the design task they were accepted into the community at large. However, the next step in this evolution is being ignored by the large tool vendors hindering the continuation of this process. With system level definition and simulation through the logic specification well understood, why is the physical generation so blatantly ignored. This portion of the development is still treated as an isolated task with information being passed from the designer to the layout function. Some form of result given back but it severely lacks full definition of what has transpired. The level of integration in I.C.'s for tomorrow, whether through new processes or applications will require higher speeds, increased transistor density, and non-digital performance which can only be achieved through attention to the physical implementation.

1 Introduction

The future of circuit integration is rapidly moving toward high speed and/or analog elements. Digital signal processing techniques can take advantage of the fastest possible logic and with the Gallium Arsenide process becoming available this means clock rates of 250-500 Mhz. The "real world" is also inherently analog in nature and this means more system integration will require analog operations. Both of these areas require absolute control of the physical placement of transistors and interconnect.

Tool makers have historically treated physical layout almost as an after thought in the design process. The conventional design methods ignore the physical placement until the end and can then only define it in a geometrical format with no real relationship to the original definition except for connectivity. There have been tools developed to help manage this problem such as Frameworks but nothing to "fix" the problem.

The way to correct this problem area is to place equal importance on the physical as well as the logical and system definition. Equal importance means that they are all done at the same time in the design process and best by the same person. Adding this third dimension cannot be realized unless the design method is changed to allow it.

Physical definition fundamentally contains positional information and effects the performance and possibly the function of transistors. In this new design method the designer can take advantage of this positional information by defining what logic elements are to be adjacent and how interaction, whether connection or function, is to take place on all sides. To give equal importance to the interconnect and simplify its implementation would mean to have it on the same level as logic functions. Implementing logic is having predefined transistors personalized at a local position. Applying this equally means having predefined wire interconnects with personalized connectivity at a local position.

The requirement for this new design method clearly could not be satisfied by current tools. The benefits and demands for such a method could no longer be ignored. This led to the development of a new VLSI design tool called Path Programmable Logic (PPL) [1] [2].

2 The PPL Design Method

PPL is a two dimensional specific cellular grid-based design method which combines logic function and physical layout definition at a higher level. This higher level relates to a system level description in terms of binary values, register descriptions, logic operations, as well as connectivity. In the case of analog operations, it can utilize opamps, switches, comparators, and any other components that can be realized on an I.C. These functions are represented as a character. The character has a one-to-one correspondence of position, form factor and area to both its logic icon and physical implementation on the chip. Each cell with connecting wires running through all four sides is placed into a "sea-of-wires" through the use of these characters. The connection between adjacent cells is implicitly made unless a break is inserted to electrically isolate them from each other.

The PPL method places more emphasis on wiring than on transistors. The PPL circuit plane is covered with two sets of wires, one running horizontally, the other running vertically as shown in Figure 1. Cells are placed under wires to form logic function. Subsets of the wires in each direction are collected into an area called a unit cell. The size of the unit cell is determined by the number of wires that must pass through in each direction and the number of transistors for forming the simplest function.

PPL cells are designed to join all four sides with interconnecting wires running through the edges of each cell at the predefined wire locations. Interconnection between adjacent cells is implicitly made, unless a break is inserted. Breaking a wire which comes from a port of a cell means that port of the cell does not connect to the outside world as shown in Figure 1. Studies have shown that more connections are made than breaks of connections.

The PPL design method simultaneously specifies logic, layout, and interconnect since the spatial mapping of the PPL character to logic symbol or physical layout is one-to-one. Thus, like full custom design, the designer has control over placement and routing which effect signal coupling, inductance and stray capacitance, while, like semicustom design, hiding layout details from designer. This feature is ideal for the design of analog and high speed, critical path circuits.

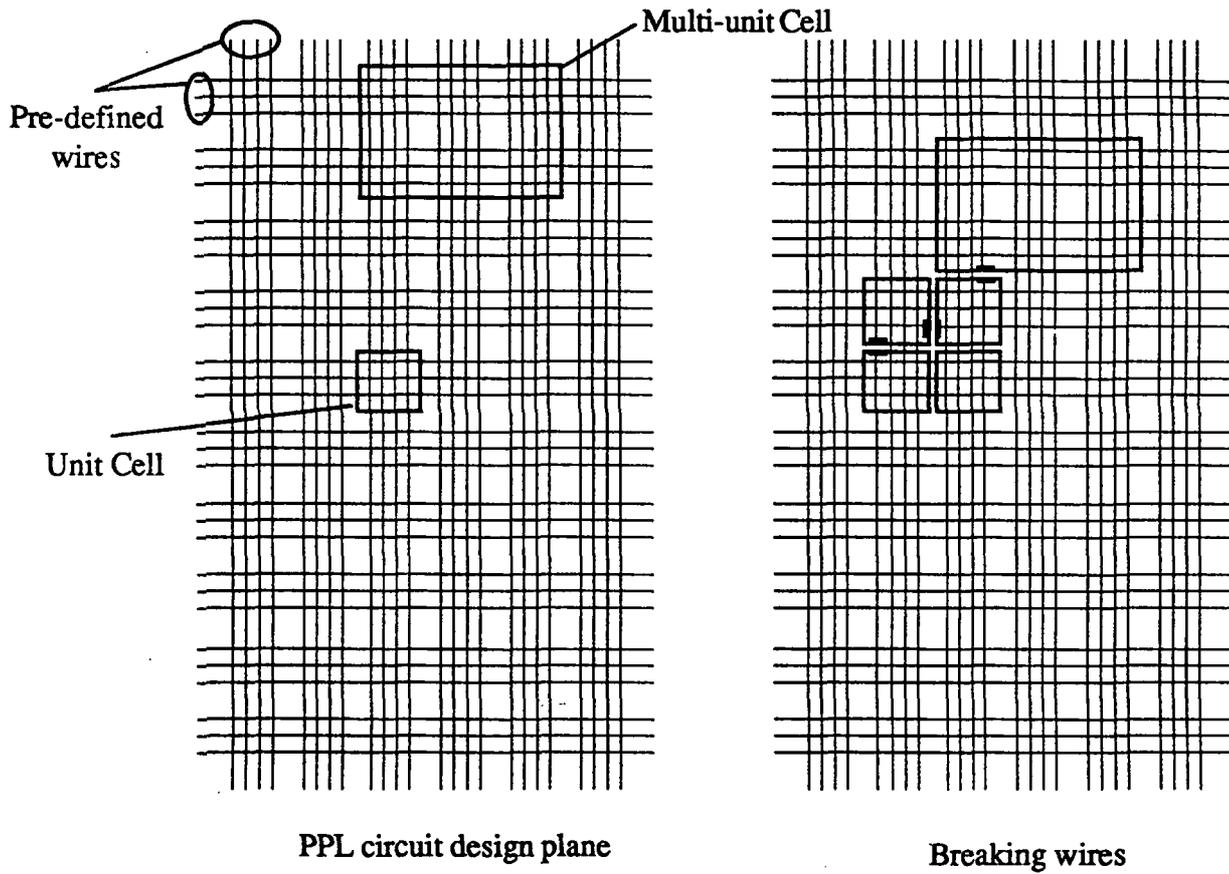


Figure 1: PPL circuit desing plane & Breaking wires

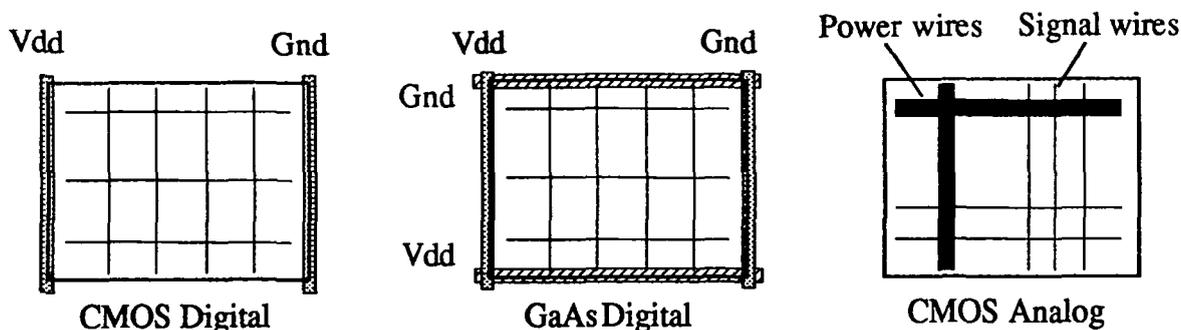


Figure 2: Varied implementation for power distribution

The power busses can be included in the unit cell (Figure 2) so that no explicit power and ground need to be done by the designer. The configuration can be changed according to the need such as alternating in the Y-axis for lower power Cmos logic, in both X and Y directions forming a mesh in different layers of metal for high speed, high current GaAs logic, or routeable for Analog implementation. Power busses can also be shared between cells in row or columns which implies that cells at alternate row or columns are being mirrored automatically by the design tool.

3 PPL Cells

With an initial design plane filled with blank cells, PPL circuits are designed by replacing blank cells with cells that form logic functions. In Figure 3, one blank cell is replaced by a cell connecting a column wire to a row wire. Another blank cell is replaced by an unit capacitance cell. All cells are rectangular. Ports at four edges of a cell correspond to the points where signal wires or maybe power wires (as in the case of analog circuit section) cross cell borders. Cells can be as small as a unit cell or a multiple unit cell size in both x and y directions. Cells may have modifiers that may change a cell wire scheme, its function, or its value. For instance, cell "c" (capacitor cell) with modifier of 0 means its capacitance equals 1.00 unit capacitance (C_u), while a modifier of 99 means its capacitance is $1.99C_u$.

Digital cells are quite primitive, consisting of 2 to 4 transistors and covering only one grid location (unit cell size) in many cases. They have been carefully designed to fit together to form a useful distributed logic function. Figure 4 shows a PPL circuit of an EXOR function which looks very similar to the format in the EXOR truth table. Each PPL cell is represented by a logic character with specific meaning. For example, the logic symbol 1 indicates that this "1" cell senses a column containing a logic Q or TRUE output.

The 1 and 0 characters will actually group when they are on the same row forming an AND logic function. The plus characters will group in the column or vertical direction forming an OR function. The design tool will automatically complete the gate structure by connecting to power and, in the case of complementary CMOS logic, adding the appropriate connections to each output forming the node and the series the N-channel transistors are

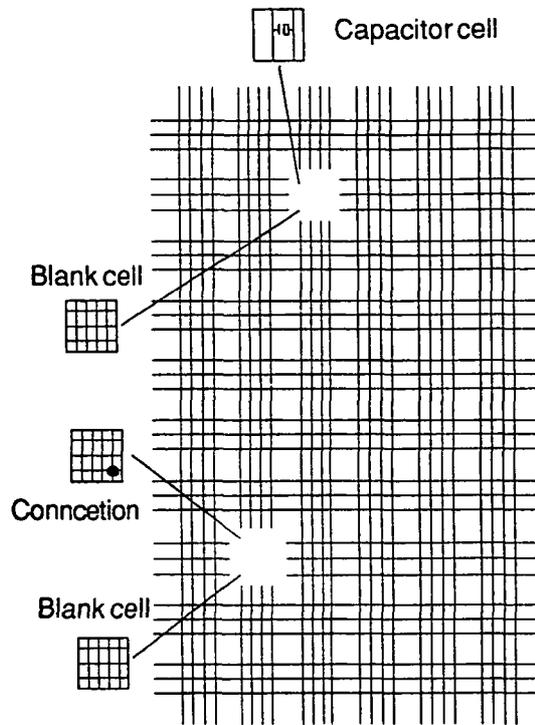


Figure 3: PPL Design-replacing blank cells

grounded at one end and shorted to the node on the other. This expression is then extracted as gates in a netlist format complete with performance characteristics annotated to the gates because the physical layout is also defined when the character is placed.

A	B	Out	EXOR truth table
1	1	+	A
1	0	+	B
0	1	+	Out
0	0	+	1
0	1	+	1
1	1	+	0
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0
1	1	+	1
1	0	+	0
0	1	+	1
0	0	+	0

Characters	22
Transistors	190
Area	150 mi. sq. @ 2u rules
Description File	670 bytes
Schematic File	25173 bytes
Layout Cif file	20447 bytes
Gate Netlist	3737 bytes
Transistor Netlist	11764 bytes

Table 1: Counter Statistics

fixed, this notation did not need to explicitly display it. Removing interconnect from a drawing greatly reduces its complexity and removes a undesired burden from the designer. With a coarse grid defined by the unit cell placements an icon representation equal to this can be text characters. To show size of cells, a character with fill characters occupying the extent of a large cell can be displayed. The use of characters is also more translatable to a system level description.

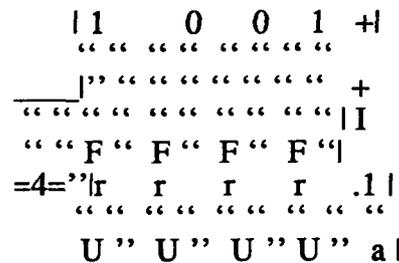


Figure 5: Design notation of counter

Figure 5 is the design notation for a BCD counter. It is constructed of 4 basic flip-flops ("F" characters occupying 4 rows by 2 columns) placed next to each other causing them to be connected together. The "4" character generates a two phase clock driver for the flip-flops. A synchronous reset is added to the flip-flops through the use of the "r" character. The flip-flops are made to count by adding the "U" character. A data path in the vertical direction is constructed in this manner and its width is a simple expansion or repetition in the horizontal direction. The binary value of 1001 (nine) is decoded at the top and ORed with an external reset signal then sensed by a "1" character that drives the synchronous reset of the flip-flops. This reset is also ANDed with an external count enable signal by the "a" character to control the carry-in of the "U" cells. This notation completely defines the function and placement of 190 transistor occupying 88 (8 x 11) unit cell locations. The size of the physical layout would now depend upon the technology to be used ie. Gate Array, GaAs, Custom, 2u rules, 1.2u rules, etc. This same notation is transportable across any of these technologies without any redefinition, only a reverification of performance.

The design notation requires only 22 character placements to fully define the BCD counter. This represents the amount of actual effort required by the designer. The con-

nectivity was generated automatically and for clarity some signal wire names were added that resulted in the description file being 670 bytes of data. This file contains enough information to allow the automatic generation of the schematic, layout, and annotated netlists. Table 1 of counter statistics shows designer's effort is leveraged approximately 3000X.

5 Conclusion

IC's which contain analog or high speed sections require the design tools to give the designer the ability to place the interconnections as well as the logic. The new PPL design method incorporates tools that meet this requirement. No other tools, aside from the old methods of handcrafting a custom chip, provide this capability. Other considerations include transistor density, productivity, and design portability. PPL designs are typically within 10% of custom densities, are completed in one tenth the time of custom or one half the time of standard cell implementations and are readily ported to various technologies and/or design rules. Total design effort can be easily done by a single designer without need of scheduling and interfacing with specialized resources such as layout engineers. The critical aspects of the design are always taken into consideration because the person defining the critical areas is also doing the implementation. The data compression is sufficient to allow designs to incorporate millions of transistors in a manageable form. The mundane but critically important task of documentation is taken care of by the concurrent generation of schematics from the logic placement.

References

- [1] K.F. Smith, B.E. Nelson, T.M. Carter and A.B. Hayes, "Computer-aided design of integrated circuits using path-programmable logic," IEEE Electro '83 Professional Program Session Record, 1983.
- [2] B.E. Nelson, D.R. Morrell, C.J. Read and K.F. Smith, "The PPL integrated circuit design methodology," Computer Aided Design, 1986.

Automated Synthesis of Sequence Invariant State Machines

D. Buehler, S. Whitaker and J. Canaris
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - A CAD tool for the design of VLSI synchronous sequential controllers is presented. Both the design and layout of the state machine are automatically generated. The program is process independent allowing a choice of design rules to base generation upon. An incremental layout creation approach has been implemented which makes the tool useful in a wide range of layout applications. Flow table descriptions are input to characterize the desired machine and a layout archive is output.

1 Introduction

A state machine can be implemented as a programmable PLA based structure or as random logic. The realization of state machines based on random logic often results in the most compact and highest performance circuits, but the logic, which is a function of the state assignment, flip flop type and flow table, does not lend itself to easy automatic synthesis. Controllers implemented as PLA structures can be generated automatically, but are less area efficient and have reduced system performance.

An architecture that retains the traditional strengths of dedicated state machines, but offers the programmability of a microcontroller, was presented in [1]. This architecture produces controllers whose logic is invariant with respect to the actual sequence desired. State machines designed using this method approach the performance and size of random logic based state machines and have a programmability superior to a PLA based design.

This paper presents a CAD tool for generating VLSI dedicated controllers based on the sequence invariant architecture, described in Section 2. The tool, named "sm" (for SISM maker), is a flexible tool which not only provides the layout of complete state machines, but can generate stand alone sub-circuits as well. The layouts are design rule independent and correct by construction. Layout generation is done on the fly, with a process rule file being the minimum input required. The software, which is a pipeline of sub-circuit generators, provides great flexibility and speed, for an engineer and layout designer, in the implementation of CMOS controllers. A significant reduction in engineering design time is also attained, as "sm" performs all of the logic design tasks normally associated with state machines. The interesting and creative design task, flow table definition, remains in the hands of the engineer, while the time consuming portions are handled in software.

	I_1	I_2	I_3
A	C	B	A
B	D	C	B
C	E	D	C
D	F	E	D
E	A	F	E
F	B	A	F

Table 1: Example flow table.

Section 3 of this paper describes the approach "sm" takes to layout generation itself, while Section 4 provides a description of the input options available. Section 5 outlines the benefits of the CAD tool.

2 Sequence Invariant State Machines (SISMs)

2.1 What does "Sequence Invariant" mean?

State machines are used primarily as control structures in digital circuits. A traditional format for the description of a state machine is the flow table, such as shown in Table 1. The flow table has dimensions of i inputs wide by s states tall. Entries in the flow table determine the state transitions that are made during operation.

The driving idea behind the sequence invariant architecture is to build a state machine given only the dimensions (i and s) of the flow table. The state machine created must be capable of performing the operations of *any* sequence of states that are described in the i by s flow table.

2.2 How is "Sequence Invariance" accomplished?

The sequence invariant architecture is broken into functional blocks as shown in Figure 1. In effect, the Destination State Codes are a representation of the flow table to be implemented. The Input Switch Matrix selects a single column of the flow table and passes the next state information for the entire selected column to the Next State Logic. The function of the Next State Logic is to select which of the states from the column of states presented to it should be selected as the next state. This choice of next state is based upon the current state. The hardware implementation of the Input Switch Matrix and Next State Logic circuits is dependent only on the dimensions of the flow table to be implemented. The only difference between state machines with dimensions $i = x$, $s = y$ is the programming of the Destination State Codes.

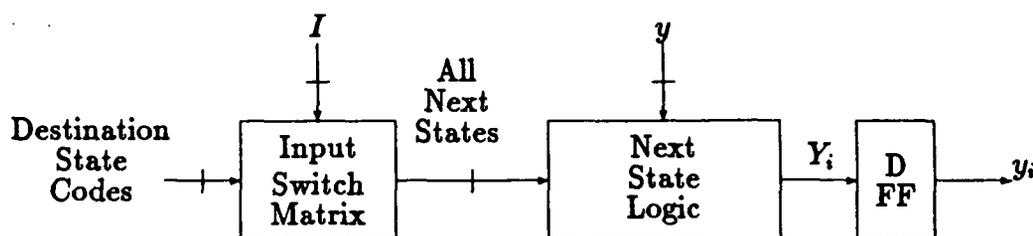


Figure 1: General block diagram.

	I_1	I_2	I_3
S_0	N_{01}	N_{02}	N_{03}
S_1	N_{11}	N_{12}	N_{13}
S_2	N_{21}	N_{22}	N_{23}
S_3	N_{31}	N_{32}	N_{33}
S_4	N_{41}	N_{42}	N_{43}
S_5	N_{51}	N_{52}	N_{53}
S_6	N_{61}	N_{62}	N_{63}
S_7	N_{71}	N_{72}	N_{73}

Table 2: General eight-state three-input flow table.

2.3 Operation

The following illustrates specifically how this architecture works. Let Table 2 depict an example for a general 3 state variable, 3 input state machine. I_1, I_2 and I_3 are the inputs, $S_0 \dots S_7$ are the present states, and $N_{S_0 I_1}, N_{S_0 I_2} \dots N_{S_7 I_3}$ are the next states. This can be generalized so that $N_{S_i I_j}$ are the next states for S_i under input I_j . $N_{S_i I_j}$ has been abbreviated as N_{ij} . The set of N_{ij} also comprise the destination state codes. Let the state assignment be $S_0 = 000, S_1 = 001, S_2 = 010, \dots, S_7 = 111$.

The next state logic is a general BTS circuit [2,3] with paths that decode each state. The input switch matrix is a pass transistor matrix, that passes the destination state codes to the next state pass network as shown in Figure 2. The circuit realization of this network operates in the following manner: All of the destination state codes N_{ij} are presented to the input switch matrix. For each input state I_i , all of the destination states in I_i are presented to the next state logic. The present state variables, y , select one and only one next state entry which is passed to the flip-flops. If the machine is in state S_1 and input I_2 is asserted, then N_{12} would be passed to the input of the flip-flop for next state variable Y_i . The current input state selects the set of potential next states that the circuit can assume (selects the input column) and the present state variables select the exact next state (row in the flow table) that the circuit will assume at the next clock pulse.

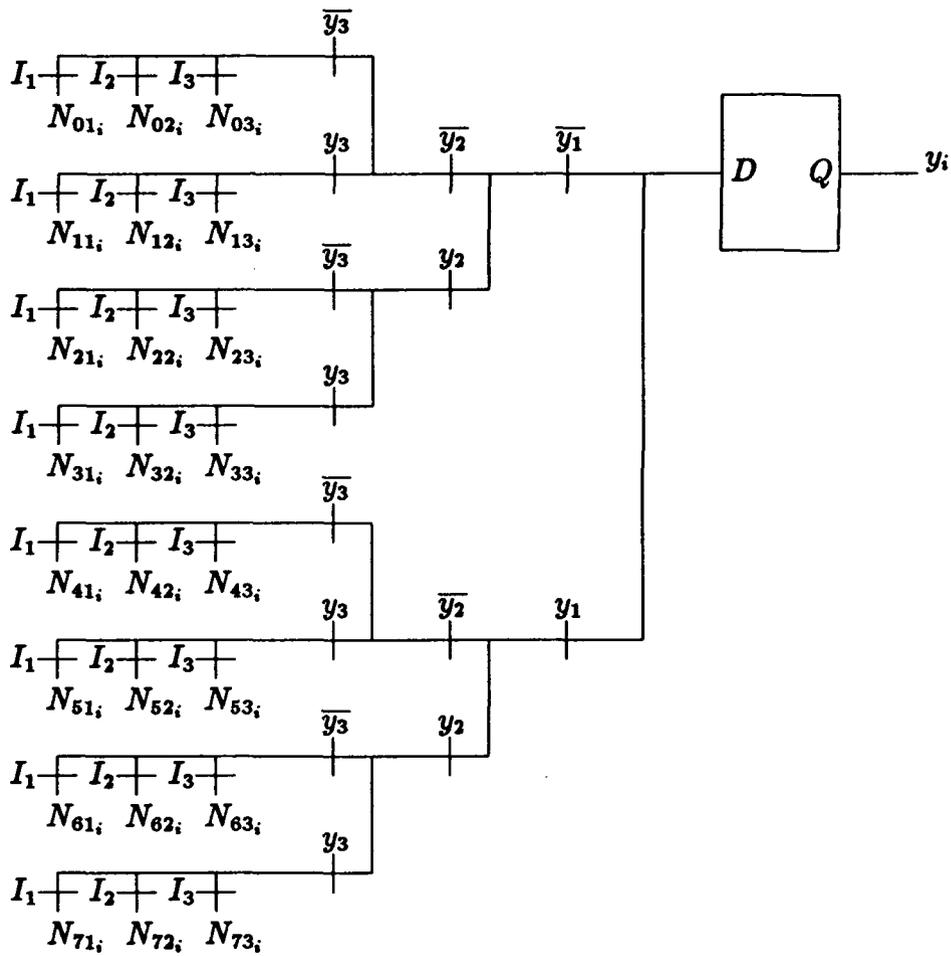


Figure 2: General eight-state three-input next state equation circuit.

2.4 SISM's and state machine output equations

It has been shown [4,5] that the same logic which is used to implement Sequence Invariant State Machines can also be used to generate the forming logic for the state machine output equations. In this case the feedback terms are removed. The present state information is provided by the logic blocks in the SISM. The same sequence invariance is available, as is the programmability. The CAD tool described here has the ability to generate logic blocks suitable for implementing these output equations.

3 Automating Layout

3.1 Layout approach

The structure of the SISM logic lends itself to two "natural" layout approaches. First, the regularity of poly spacings vertically and metal spacings horizontally, (see Figure 3), suggests a gate-matrix layout approach [6]. In this approach, horizontal poly-silicon and vertical metal pitches are calculated, then contact and diffusion areas are placed to form transistors. The gate-matrix approach proves undesirable due to the inflexibility of the monolithic layout structure created. The second approach is also suggested by the regularity of the SISM logic. The layout can be divided into a number of tiles which can be calculated, created, then placed. The tile-laying methodology was chosen because it allows decomposition of the layout problem into smaller cell layout problems.

Although the ultimate goal is a complete state machine layout, this methodology allows useful layouts to be made available, as intermediate steps, in the compilation. The next section describes the layout decomposition.

3.2 Layout Decomposition

The organization of the SISM layout, (see Figure 4), parallels the functional blocks shown in Figure 1. Decomposition of the layout for creation purposes follows the functional decomposition shown. "sm" creates a cell implementing the Input Switch Matrix and a cell for the Next State Logic. Additional layout cells provide programmable connections for power and ground. Feedback taps for the state variables are also generated.

A single state bit cell can be created and replicated for each state bit because the Input Switch Matrix and Next State Logic are identical for each state variable bit (due to Sequence Invariance). After this base architecture is created, a programming mask is placed over the Input Switch Matrix to program the desired state transitions.

As described in Section 2.3, the Next State Logic block is a general BTS network and Figure 2 indicates that the Input Switch Network is a general pass transistor OR structure. Both of these functional blocks are found in logic implementations, other than state machines. Therefore, "sm" was designed to optionally create layouts of these sections independent of a particular SISM design. The user can then copy, connect, and program

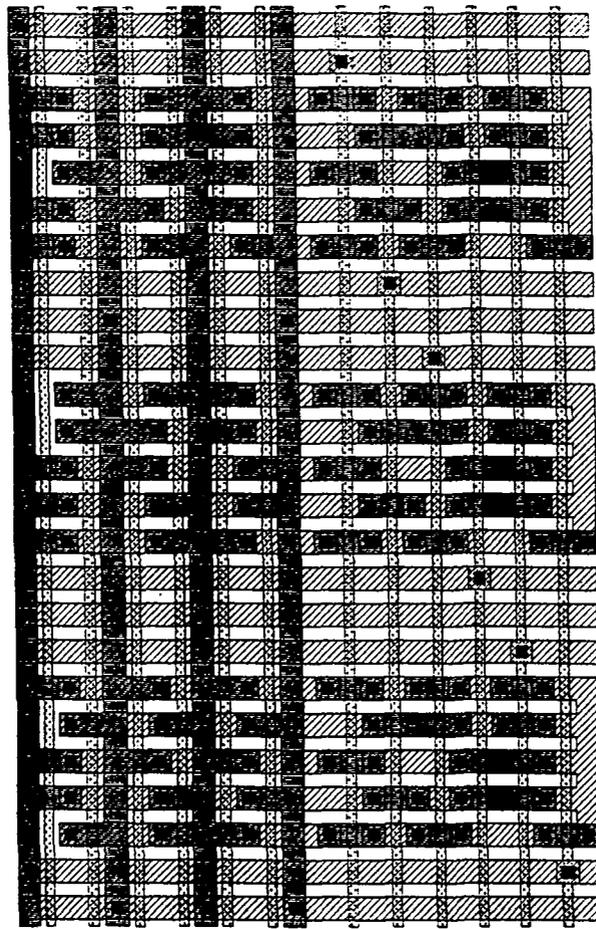


Figure 3: Three input, five state SISM layout

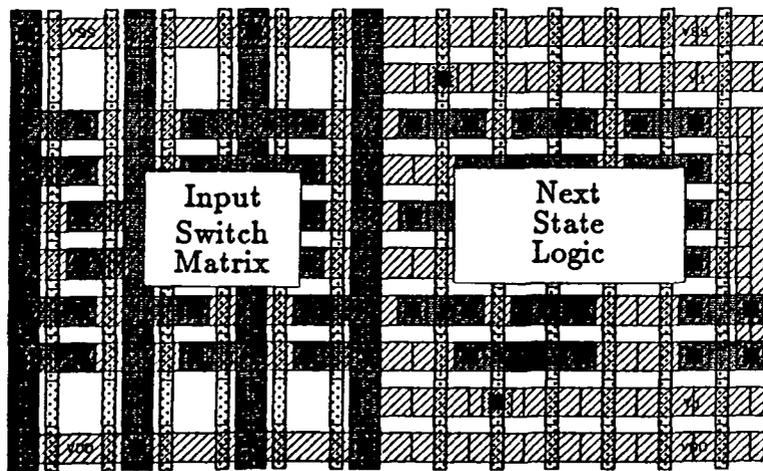


Figure 4: Layout showing functional areas

```
NAME: MACH1
PROCESS: CMOS34
a:  c b a.
b:  d c b.
c:  e d c.
d:  f e d.
e:  a f e.
f:  b a f.
```

Table 3: Flow table entry format

them manually. This allows more flexibility in the layout process, and provides automatic generation of structures which may have uses in applications other than SISMs.

4 “sm”

4.1 Layout options

The amount of layout generated is very flexible. The user can choose to specify the number of states and generate the Next State Logic cell. Specifying the number of inputs in addition to the number of states allows the user to generate an unprogrammed Input Switch Matrix and/or a Next State Logic cell, or a complete, unprogrammed state machine. These options allow the layout designer to create blocks which generate the output signals of a state machine, as described in Section 2.4. To generate a complete, programmed state machine the user must create a file with a flow table description of the state machine.

4.2 Flow table description

When the user requests creation of a complete, programmed state machine, “sm” requires a flow table description of the desired state machine. The flow table description is a simple ASCII text file consisting of state transition information. Table 3 shows the flow table entry format used to create a state machine corresponding to the flow table shown in Table 1. State declarations are followed by a colon and the state transition information for each input, then closed with a period. Undefined or “don’t care” states are not allowed. States are assigned sequentially, beginning with the first state declared, which is assigned state variable value zero. It is often advantageous to assign this state to the reset state.

4.3 Process choices

The CAD tool described here is process independent. Layouts are created on the fly, using a tiling algorithm. Fabrication of VLSI circuits, however, is not process independent.

4.3.8

When targeting a specific process, "sm" requires a Design Rule file for that particular process. "sm" currently has two Design Rule files available:

- Hewlett-Packard 1.6 μ m CMOS-40 rules.
- Hewlett-Packard 1.0 μ m CMOS-34 rules.
- Implementation of the MOSIS Scalable Rule set is planned.

4.4 Transistor width

The transistors used in the SISM layout default to a minimum transistor width, which is the size of a metal/diffusion contact plus diffusion overlaps on each side. If the user wishes, the transistors can be created larger than this minimum. The resolution of transistor widths allowed is 0.2 μ m .

5 "sm" and SISM Benefits

Use of "sm" and sequence invariant state machines has many benefits over PLA and random logic state machine implementations. Running on an HP 9000/375 "sm" took two seconds real time (half a second CPU time) to create the layout shown in Figure 5. This layout implements the state machine from the flow table description shown in Table 3. The layout created is correct by construction and will pass Design Rule Checks (DRC). A layout designer would require half a day to lay out, check continuity, and run DRC verification on a hand crafted SISM state machine layout.

A traditional state machine requires a significant amount of work. The engineering tasks required include flow table construction, state assignments, choosing a flip-flop, designing input forming logic, designing output forming logic, and circuit design. The layout task is one of the most time consuming in a VLSI design. An experienced layout designer may be able to draw about 8 transistors/hour of random logic. Traditional controller designs tend to have large amounts of random logic associated with them. A traditional implementation of the flow table, Table 3, might require several days of layout.

Time is saved, using "sm", because no state machine design time, other than the flow table construction, is required. The sequence invariant implementation will yield additional savings in the event of design changes which require modification of the state flow sequence or the addition of new states.

"sm", while remaining a simple program to use, provides flexibility to the layout designer, as well as time savings to the design engineer.

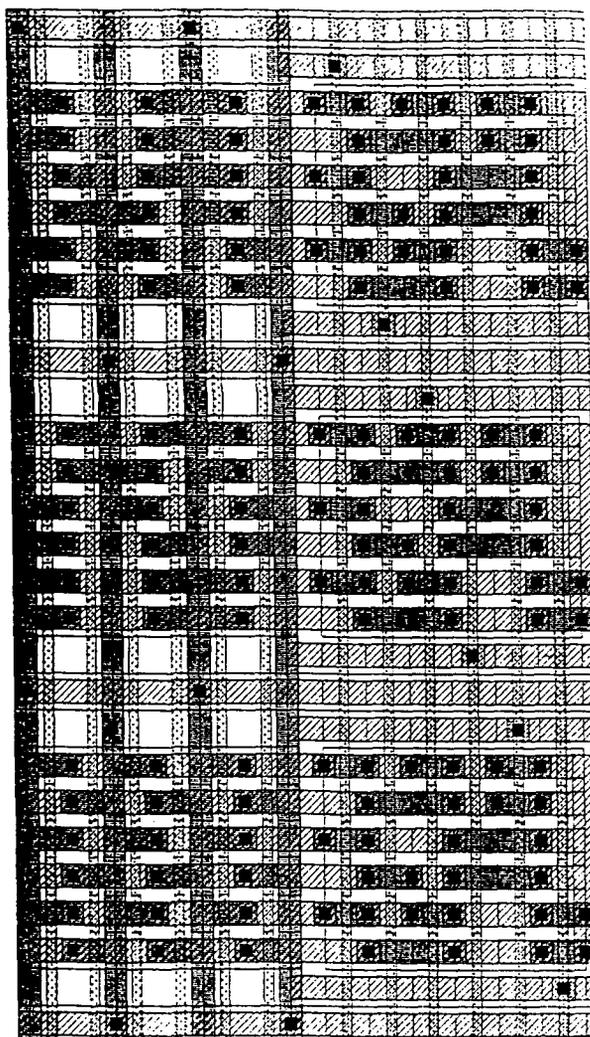


Figure 5: "sm" generated layout

References

- [1] S. Whitaker, S. Manjunath and G. Maki, "Sequence Invariant State Machines", submitted to the IEEE Journal of Solid State Circuits.
- [2] G. Peterson and G. Maki, "Binary Tree Structured Logic Circuits: Design and Fault Detection," Proceedings of IEEE International Conference on Computer Design: VLSI in Computers, pp. 139-144, Oct. 1984.
- [3] D. Radhakrishnan, S. Whitaker and G. Maki, "Formal Design Procedures for Pass-Transistor Switching Circuits," IEEE Journal of Solid State Circuits, pp. 531-536, Apr. 1985.
- [4] S. Whitaker and G. Maki, "Pass Transistor Asynchronous Sequential Circuits", IEEE

4.3.10

JSSC, Vol. SC-24, pp. 71-78, February 1989.

- [5] S. Whitaker, G. Maki and M. Canaris, "A Programmable Architecture for CMOS Sequential Circuits", Proceedings of the NASA SERC 1990 Symposium on VLSI Design, Moscow, Idaho, pp. 223-229, January 1990.
- [6] O. Wing, S. Huang and R. Wang, "Gate Matrix Layout", IEEE Transactions on Computer-Aided Design, pp. 220-231, July 1985.

This research was supported in part by NASA under the NASA Space Engineering Research Center grant NAGW-1406 and by the Idaho State Board of Education under grants 88-038 and 89-041.

CMOS Output Buffer Wave Shaper

L. Albertson, S. Whitaker and R. Merrell
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - As the switching speeds and densities of Digital CMOS integrated circuits continue to increase, output switching noise becomes more of a problem. This paper reports a novel design technique which aids in the reduction of switching noise. The output driver stage is analyzed through the use of an equivalent RLC circuit. The results of the analysis are used in the design of an output driver stage. A test circuit based on these techniques is being submitted to MOSIS for fabrication.

1 Introduction

There are two main types of output switching noise in CMOS circuits which can cause problems for circuit designers. These types, although closely related, can be defined as ground bounce, which is the variation of the chip ground to the external ground and ringing (undershoot/overshoot) on the output signal lines. Ground bounce, is caused by the rapidly discharging or charging of the output capacitance and is accentuated by simultaneously switching output drivers. Ringing on output signals is caused by the improper damping of the inherent RLC discharge or charging loop. Both of these sources of noise are attributed to having an inherent inductance in the charging or discharging path of the output load capacitance.

There are several reasons why output switching noise should be eliminated or at least reduced below some specified level. Noise caused by ground bounce can cause false switching of internal gates leading to incorrect logic states and reduced reliability of the integrated circuit. Ringing on signal lines is also undesirable in that it can falsely trigger an input resulting in incorrect data to other circuits in the system. Both types of switching noise are also a problem in the area of integrated circuit testing.

Over the past decade or so there has been a significant amount of effort put forth in finding solutions to the problems of output switching noise. Many of these solutions have not been published in the literature due to integrated circuit manufacturers attempting to maintain their competitive edge. However, most of these solutions are based upon three main approaches:

1. providing a slower turn on time of the output transistors,
2. staggering the turn-on times for the different output transistors [1], and
3. a noise sensing feedback approach.

All of these techniques have one main drawback; the output buffer will be inherently slower, thus speed is sacrificed for a decrease in the amount of generated output noise.

This paper concentrates on analysis of the undershoot/overshoot problem and a possible solution for use in high speed CMOS VLSI circuits.

2 Analysis

A typical CMOS output driver circuit is shown in Figure 1. It consists of scaled inverter stages which are generally a factor, f (in the range from 3 to 8) times larger than the previous stage. The problem of scaling to optimize delay and/or area of the driver chain has been examined by several authors [2]-[6]. For the purposes of this work, scaling to optimize delay of the output stage was not addressed, however it can become very important when maximum speeds are required since the ringing noise reduction techniques create an extra time delay.

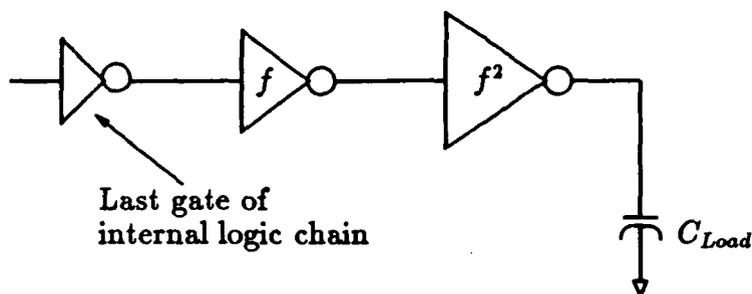


Figure 1: Typical scaled output buffer stage

The output driver structure of a CMOS VLSI circuit can be complicated to analyze, if viewed as a complete circuit. Therefore, it is necessary to break it down into simpler subcircuits which can be easily analyzed. The first part of this analysis will deal with the RLC discharge loop of the load capacitance during a low going transition of the last stage of the output buffer. Secondly, the resistance of the output driver FET will be analyzed and finally a technique for minimizing the problems of ringing will be discussed.

2.1 RLC Discharge Loop

A typical CMOS pull-down circuit of an output buffer stage is shown in Figure 2. It consists of an n-channel transistor with its drain tied to a load capacitance, its source tied through an inductance to ground, and the gate tied to an input voltage source. The pull-up circuit would be very similar, with a p-channel transistor with the source tied through an inductance to V_{cc} .

Using the simplified equivalent RLC circuit shown in Figure 3, the output switching noise produced by the circuit of Figure 2, can be analyzed.

In the equivalent RLC circuit, the n-channel transistor is replaced by a switch, S and a resistance R . The resistance R represents the ON resistance, R_{on} of the device and the

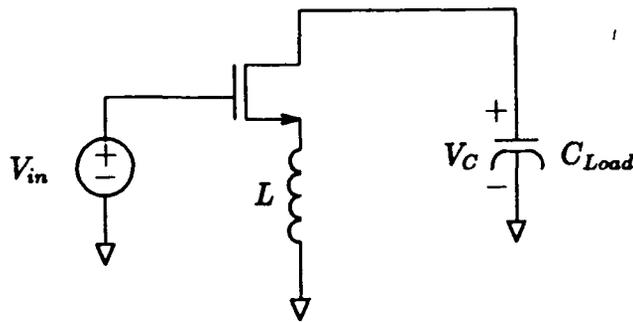


Figure 2: CMOS pull-down circuit of an output inverter stage

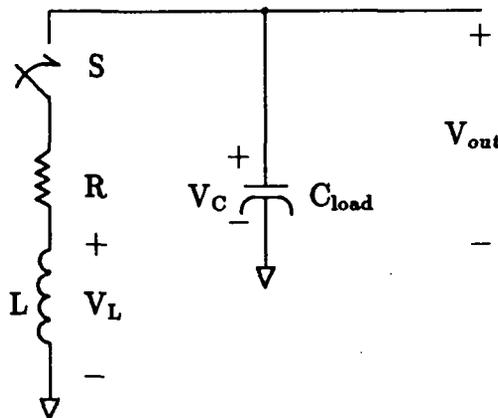


Figure 3: RLC circuit representation of the CMOS pull-down circuit of an output stage

switch S represents the switching point of the device. The value of C_{Load} is the lumped sum of the capacitances associated with the n and p-transistor drain regions, bonding pad, circuit packaging, PWB traces and input capacitances of the receiving devices. The value of L is the sum of the inductances associated with the internal ground ring, the connections between the bonding pads and the IC pins, (packaging parasitics) and the PWB ground trace (plane) and signal trace. The RLC circuit of Figure 3 can be analyzed as follows. Initially the capacitor will be charged to V_c (by the pull-up p-transistor). With the switch open, no current flows in the circuit, thus there is no initial condition for the inductance, L . At time $t = 0$, the switch is closed. The circuit is excited by the initial charge on the capacitance and the total response of the circuit can be determined as follows:

$$\sum V_d = 0 = Ri(t) + L \frac{d_i}{dt} + \frac{1}{C} \int i(t)dt - V_c(0^-) \quad (1)$$

and

$$V_c(0^-) = Ri(t) + L \frac{d_i}{dt} + \frac{1}{C} \int i(t)dt \quad (2)$$

Applying the properties of Laplace transforms yields:

$$\frac{V_c(0^-)}{s} = RI(s) + sLI(s) + \frac{1}{sC}I(s) \quad (3)$$

4.4.4

Solving for the network function $\frac{I(s)}{V_c(0^-)}$ yields:

$$\frac{I(s)}{V_c(0^-)} = \frac{1}{L} \left(\frac{1}{s^2 + \frac{R}{L}s + \frac{1}{LC}} \right) \quad (4)$$

The characteristic equation of the network is given by the denominator of the equation given above. Solving the quadratic equation for s yields:

$$s = -\frac{R}{2L} \pm \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} \quad (5)$$

Thus the roots of the characteristic equation are:

$$s_1 = -\frac{R}{2L} + \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} \quad (6)$$

$$s_2 = -\frac{R}{2L} - \sqrt{\left(\frac{R}{2L}\right)^2 - \frac{1}{LC}} \quad (7)$$

Since the quantity under the radical sign can be positive, zero or negative, three distinct solutions exist which will produce an overdamped, critically damped or underdamped response respectively.

Case 1: Roots are real and unequal resulting in an overdamped response

$$\left(\frac{R}{2L}\right)^2 > \frac{1}{LC}$$

Case 2: Roots are equal resulting in a critically damped response

$$\left(\frac{R}{2L}\right)^2 = \frac{1}{LC}$$

Case 3: Roots are complex conjugates resulting in an underdamped response

$$\left(\frac{R}{2L}\right)^2 < \frac{1}{LC}$$

For a typical case of $L=20\text{nh}$ and $C=5\text{pf}$, if the MOSFET resistance is less than 127Ω , then the circuit to be underdamped and will ring as shown in Figure 4. If the value of the inductance was even larger and the capacitive load even smaller, then R would be required to be even greater. Thus, in order to properly damp the RLC circuit, it is necessary to analyze the resistance characteristics of the MOSFET so that a good estimation of its ON resistance can be obtained.

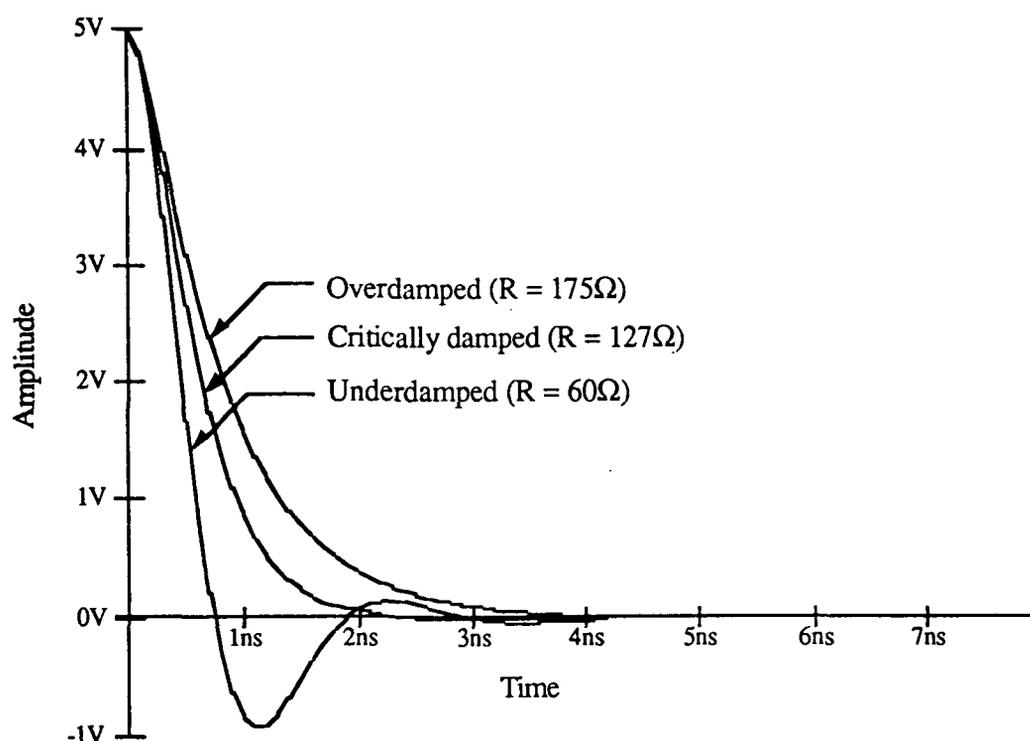


Figure 4: Voltage across the load capacitance for $L = 20nh$ and $C = 5pf$

2.2 MOSFET Resistance Characterization

The ideal (first order) equations describing the behavior of the nMOS transistor in the different regions of operation are given by [7] and shown below.

$$I_{ds} = \begin{cases} 0; & V_{gs} - V_t \leq 0 \text{ cut-off} \\ \beta [(V_{gs} - V_t)V_{ds} - \frac{V_{ds}^2}{2}]; & 0 < V_{ds} < V_{gs} - V_t \text{ linear} \\ \frac{\beta}{2}(V_{gs} - V_t)^2; & 0 < V_{gs} - V_t < V_{ds} \text{ saturation} \end{cases}$$

where I_{ds} is the drain to source current, V_{gs} is the gate to source voltage, V_{ds} is the drain to source voltage, V_t is the transistor voltage threshold voltage, and β is the gain factor of the MOS transistor and is given by:

$$\beta = \frac{\mu_0 \epsilon_r \epsilon_0 W}{t_{ox} L} \quad (8)$$

The MOSFET resistance equation for the device in the linear region can be found by taking the partial derivative of I_{ds} with respect to V_{ds} .

$$R = 1 / \frac{\partial I_{ds}}{\partial V_{ds}} = \frac{1}{\beta(V_{gs} - V_t - V_{ds})} \quad (9)$$

A simple method of approximating the MOSFET resistance is by using a linear resistor. However, upon a more detailed examination, this does not prove very accurate since the

4.4.6

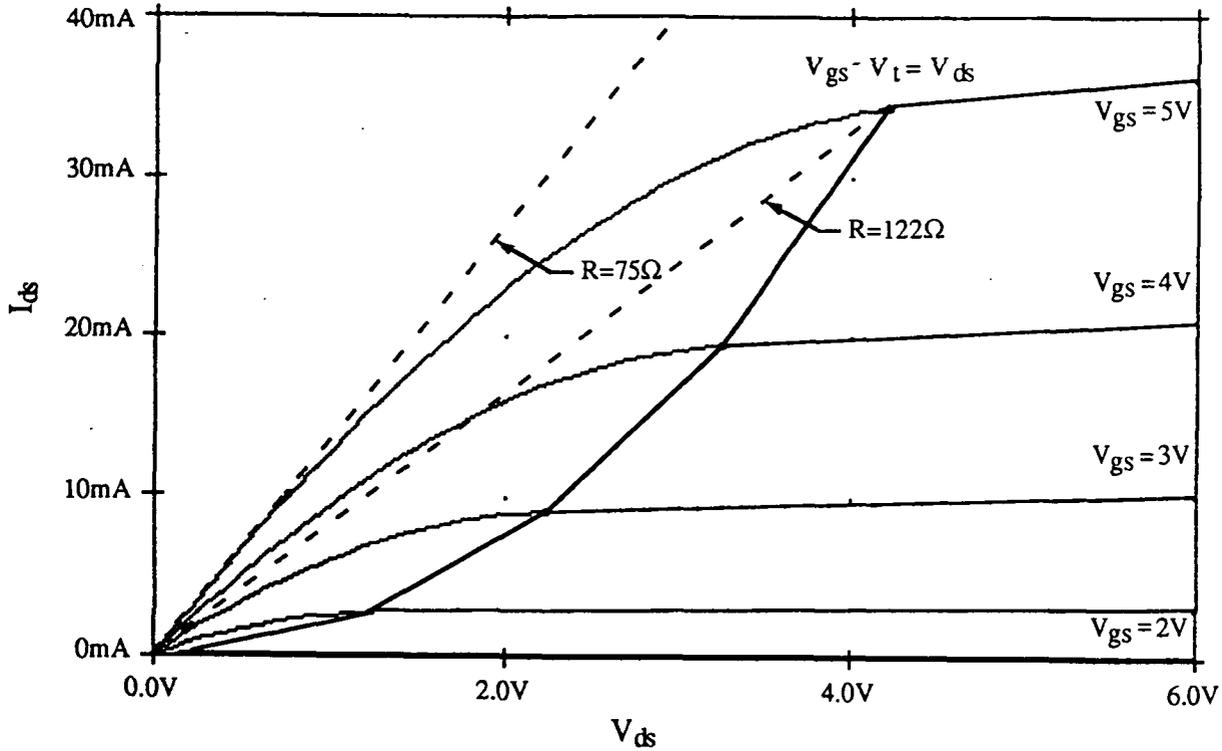


Figure 5: I-V curves and equivalent resistance of a MOSFET

approximation does not account for the nonlinear, time-dependent current-voltage characteristics of a MOSFET. Thus the equivalent resistance is a function of both V_{ds} and V_{gs} , such that $R = \beta f_n(V_{ds}, V_{gs})$. Since the input signal V_{gs} , in a normal system will not have an infinite rise time, this makes the resistance characterization very difficult.

Shoji [8] estimates the equivalent DC resistance in the following manner. Firstly, he assumes that the gate input signal switches instantaneously, thus eliminating V_{gs} from the function yielding:

$$R = \beta V_{gs} f_n(V_{ds}) \tag{10}$$

From Fig. 5, it can be seen that the resistance at the maximum input voltage, $V_{gs} = 5V$ is equal to $\frac{V_{gs} - V_t}{I_{ds}} = R_0 \approx 122\Omega$, the point the transistor moves into the linear region of operation. When the load capacitance has been completely discharged at time $t = \infty$, V_{ds} nears $0V$, and the resistance is equal to $\approx 75\Omega$, the lower bound.

In light of this, and looking back at the previous example where $L = 20nh$ and $C = 5pf$, the circuit would be underdamped when the equivalent resistance was less than 127Ω . This point on the I-V curves is when the capacitor was fully charged to $5V$, thus from that point down to when the capacitor becomes fully discharged, the MOSFET resistance will be less than the required 127Ω for overdamping.

In reality, the transistor operates quite differently on the I-V curves when a finite rise time signal is applied to the gate of the device as shown in Figure 6.

As the gate voltage rises towards $5V$, at $V_{gs} = V_t$ the transistor begins to turn on, and

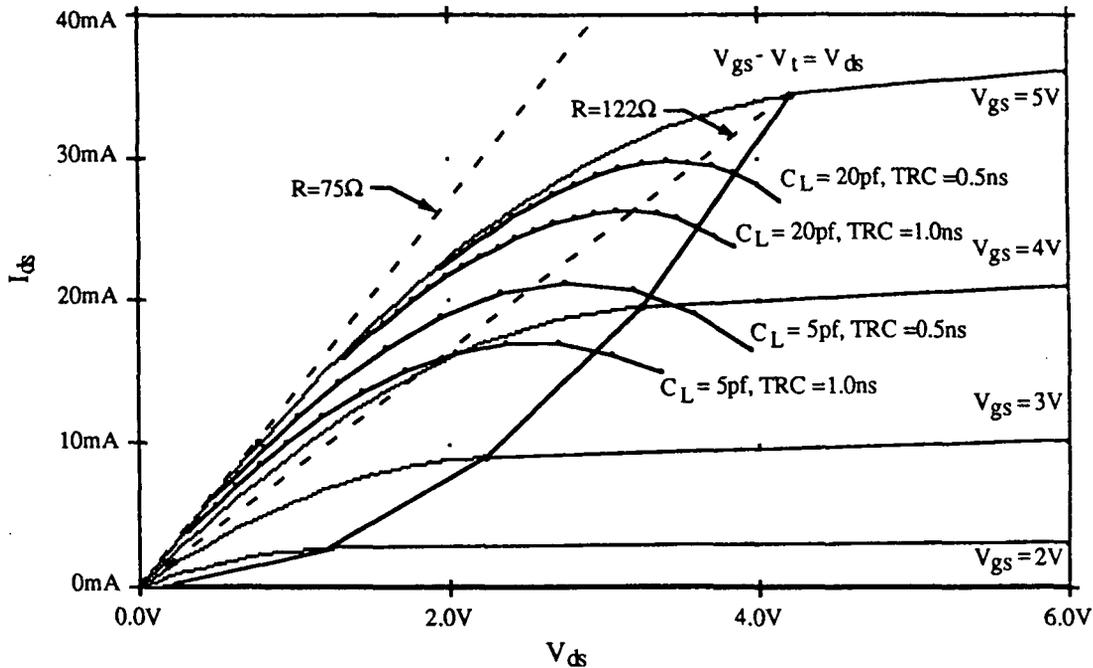


Figure 6: MOSFET I-V curves and equivalent resistance as a function of C_L and input rise time

the capacitive load begins to discharge, thus never reaching the point on the $V_{GS} = 5V$ curve where $V_{GS} - V_t = V_{DS}$. The transistor actually operates for a period of time on the lower V_{GS} curves as a function of the rise time. This is very beneficial to the damping of the RLC circuit since the resistances are higher on these lower curves as V_{GS} is transitioning toward 5V. Therefore, waveshaping the input voltage such that the transistor operates on the lower V_{GS} curves for a longer period of time is beneficial in damping the RLC discharge loop. The disadvantage to using this technique for properly damping the RLC discharge loop is that an added time delay is introduced.

2.3 Input Signal Waveshaping

There are several techniques which can be used for waveshaping the input signal to the output driver transistor. One possible way would be to size the pre-driver stage such that it can not effectively drive the input capacitance of the output driver transistor, thus slewing the input signal. Another technique [1], uses a nMOS pass-transistor, controlled by a voltage, V_{Bias} , in series with the output driver gate input as shown in Figure 7.

This technique has the advantage that the amount of resistance can be controlled somewhat by varying V_{Bias} . The drawback is that there is not a very large range over which V_{Bias} can be adjusted and still provide enough voltage to the gate of the output driver transistor. The MOSFET resistance also increases dramatically as the input voltage to the pass-transistor is increased.

These problems can be overcome by using a transmission gate in place of the nMOS pass-transistor, as shown in Figure 8.

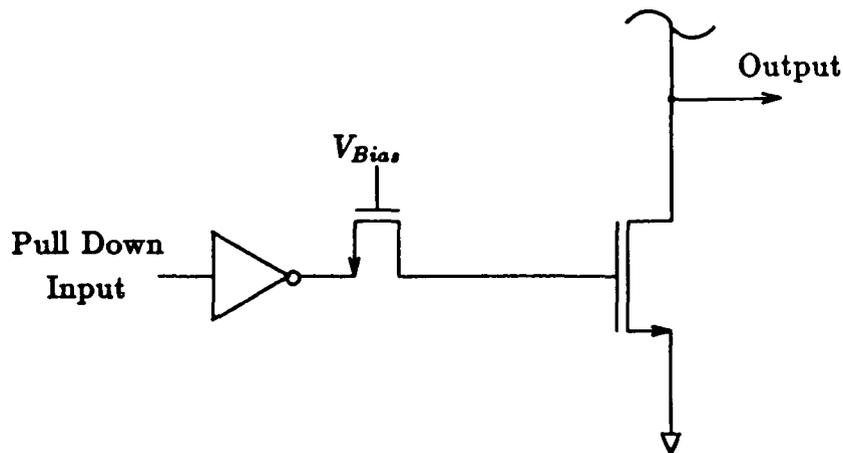


Figure 7: Output Buffer with input rise time control using an n-channel pass transistor

This also has the advantage of being adjusted by bias voltages, and will allow a wider range of adjustment. The resistance is also more constant since it is the parallel combination of the n-transistor and p-transistor resistances.

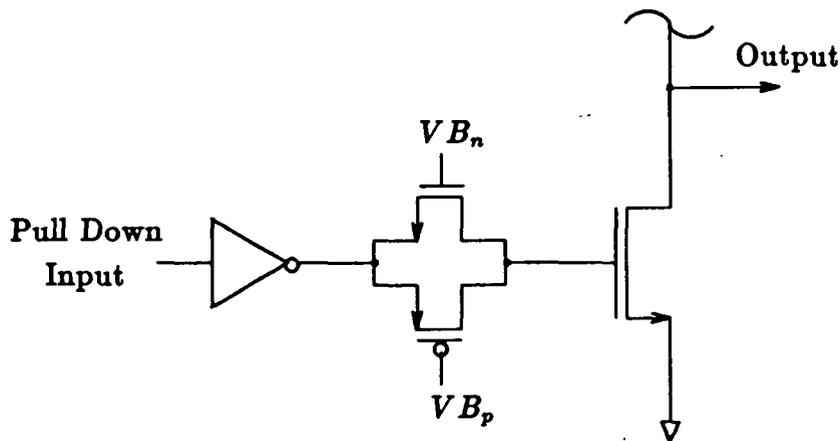


Figure 8: Output buffer with input rise time control using a transmission gate

A bi-directional TTL compatible I/O pad was designed in the MOSIS $2\mu\text{m}$ CMOS process using the techniques outlined above and shown in Figure 9.

3 Simulation Results

Shown in Figure 10 is the simulation model which was used to simulate output ringing as well as delay and rise and fall time characteristics of the uncompensated and compensated output buffers. The model includes 30nh of inductance between the buffer and the Vdd and Vss supplies and the 5pf output load capacitance. Simulations were performed using the PSpice level 1 electrical model. A set of typical parameters for the model were derived by averaging extracted parameters from 10 recent runs from a $2\mu\text{m}$ CMOS MOSIS foundry.

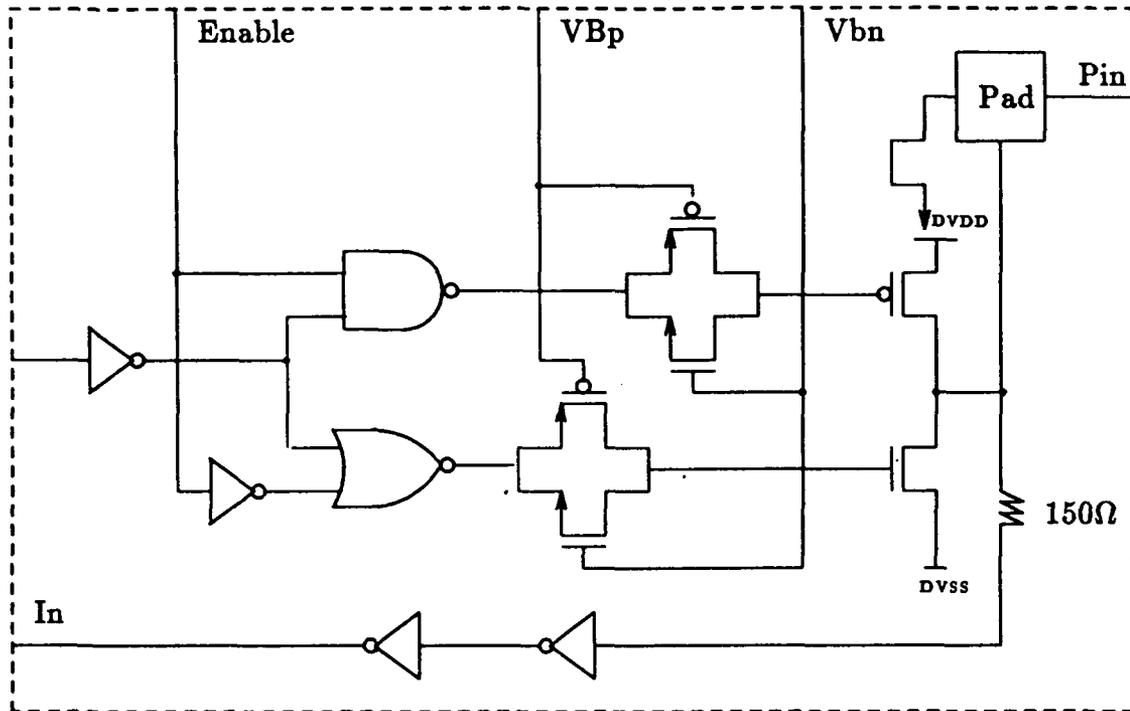


Figure 9: Bi-directional I/O Pad with ringing compensation

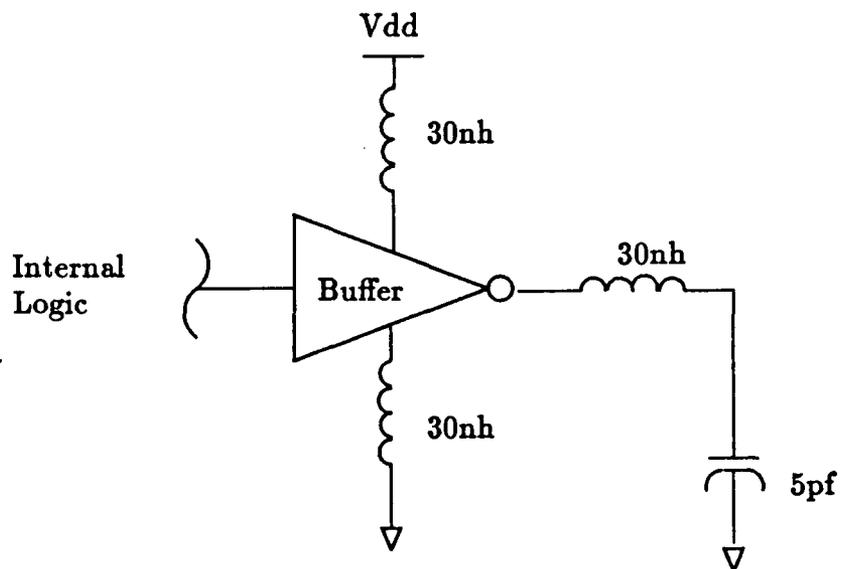


Figure 10: Output buffer simulation model

4.4.10

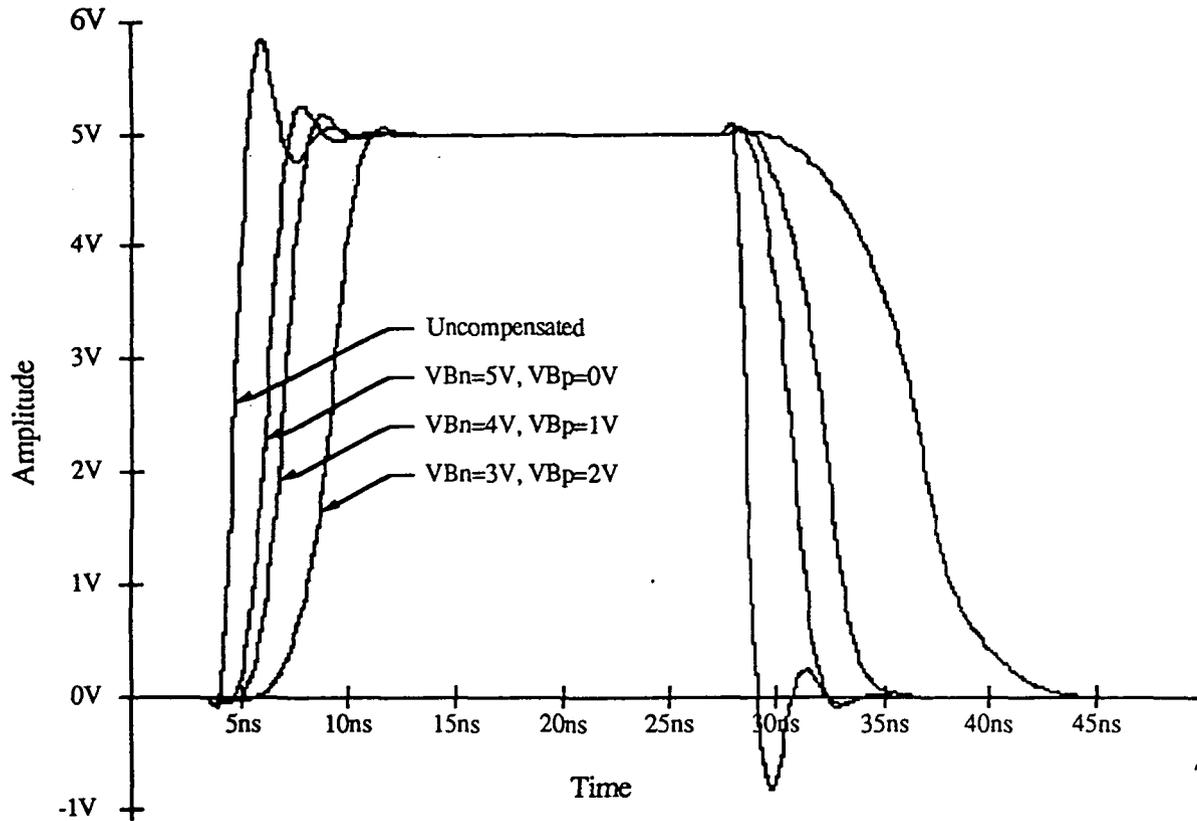


Figure 11: Uncompensated and compensated output buffer waveforms

The results of the simulations of the uncompensated and compensated buffers are summarized in Table 1 and shown in Figure 11. The results for the compensated buffer include three sets of gate voltages for the transmission gates to show the effects they have on the output ringing, delay time and rise and fall time. As can be seen in Figure 11, the cost of reducing the output ringing is sacrificing some speed performance.

Typical Case Parameters, 25 deg C, $V_{dd} = 5V$, $C_L = 5pf$				
Simulated Parameter	Standard Buffer	Compensated Buffer @ various gate bias		
		$V_{B_n} = 5V$ $V_{B_p} = 0V$	$V_{B_n} = 4V$ $V_{B_p} = 1V$	$V_{B_n} = 3V$ $V_{B_p} = 2V$
Overshoot (peak)@ 5V	0.839V	0.168V	0.250V	0.050V
Undershoot (peak)@ 0V	-0.818V	-0.084V	-0.002V	none
Delay Time	2.34ns	4.12ns	5.28ns	8.53ns
Rise Time	0.90ns	1.64ns	1.96ns	2.88ns
Fall Time	0.81ns	2.38ns	3.43ns	6.99ns

Table 1: Comparison of simulation results

4 Evaluation Test Circuit

An evaluation test circuit has been designed in the $2\mu\text{m}$ CMOS n-well process, and is being submitted to MOSIS for fabrication. Both an uncompensated and compensated buffer stage were fabricated on the same test chip. Special considerations were given to the layout to help eliminate measurement errors due to the differences in power bus drops, lead inductances, and time delays of input and output paths so that accurate results can be compared. The buffers were designed such that they are identical with the exception of the inclusion of the transmission gates in the compensated version. Their inputs were driven from the same source which was placed an equal distance from each buffer. The pads were placed such that the distance to the Vdd and Vss pads was as nearly identical as possible.

5 Summary

The problems dealing with ringing on the output signals of CMOS VLSI circuits has been discussed, analyzed and a solution proposed. A compensated output buffer was designed, using the techniques discussed, which helps to minimize the ringing effects by changing the resistance of the output driver transistors. This is accomplished by slewing the input signal to their gate using voltage controlled transmission gates. This technique allows for the proper damping of the RLC charging and discharging loops. Simulation results have been presented which substantiate the theoretical analysis. A comparison between the amount of output ringing versus the delay time and rise and fall times of the buffers was presented. A test circuit was designed in the $2\mu\text{m}$ CMOS n-well process and has been submitted to MOSIS for fabrication.

Acknowledgements

The authors wish to acknowledge the NASA Space Engineering Research Center program whose support made this investigation possible. Additionally, we would like to thank Barbara Martin for typesetting this document.

References

- [1] G. Canepa, et al., "A 90ns 4Mb CMOS EPROM," IEEE Int Solid State Circuits Conf. Dig. Tech. Papers, Feb 1988, pp. 120-121, 323.
- [2] H. C. Lin and L. W. Linholm, "An Optimized Output Stage for MOS Integrated Circuits," IEEE J. Solid-State Circuits, vol. SC-10, no. 2, Apr. 1975, pp. 106-109.
- [3] R. C. Jaeger, "Comments on 'An Optimized Output Stage for MOS Integrated Circuits'," IEEE J. Solid-State Circuits, vol. SC-10, no. 3, June 1975, pp. 185-186.

- [4] A. M. Mohsen and C. A. Mead, "Delay-Time Optimization for Driving and Sensing of Signals on High-Capacitance Paths of VLSI Systems," *IEEE J. Solid-State Circuits*, vol. SC-14, no. 2, Apr. 1979, pp. 462-470.
- [5] M. Nemes, "Driving Large Capacitances in MOS LSI Systems," *IEEE J. Solid-State Circuits*, vol. SC-19, no. 1, Feb. 1984, pp. 159-161.
- [6] N. C. Li, G. L. Haviland and A. A. Tuszynski, "CMOS Tapered Buffer," *IEEE J. Solid-State Circuits*, vol. SC-25, no. 4, Aug. 1990, pp. 1005-1008.
- [7] N. Weste, and K. Eshraghian, "Principles of CMOS VLSI Design, A Systems Perspective," Addison-Wesley, Reading, Massachusetts, 1985.
- [8] M. Shoji, "CMOS Digital Circuit Technology," Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1988.

Session 5
Featured Presentations

Chairman: Jean Teasdale

Characteristics of a Semi-Custom Library Development System

M. Yancey

Product Technology Manager

R. Cannon

SCLD Section Manager

Gould AMI Semiconductors

2300 Buckskin Road Pocatello, ID 83221

Abstract— Standard cell and gate array macro libraries are in common use with workstation CAD tools for ASIC semi-custom application and have resulted in significant improvements in the overall design efficiencies as contrasted with custom design methodologies. Similar design methodology enhancements in providing for the efficient development of the library cells is an important factor in responding to the need for continuous technology improvement. This paper presents the characteristics of a library development system that provides design flexibility and productivity enhancements for the library development engineer as he provides libraries in the state-of-the-art process technologies. An overview of Gould's library development system ("Accolade") will also be presented.

1 Introduction

Over the past decade significant advances have been made in CAD technology to support the design and layout of ASIC circuits. Today, many systems are available to assist the design engineer with such tasks as schematic capture, logic simulation, automatic placement and routing, design rule checking, back-annotation and post-layout simulation. With the advent of such CAD tools and the standardization of libraries and design methodologies engineering design productivity has been greatly enhanced. Design cycle times have been reduced from months to weeks or days. Design engineers are now in a position to focus on innovation and value-added engineering in other aspects of their system design.

Key to many current design methodologies is the availability standard cell and gate array libraries in the latest process technologies for the various design environments. Continuous improvement in process technologies creates a need for the library development system to be flexible and efficient in incorporating the ever changing process requirements into the available cell libraries. A review of the status of tools currently available to support the library development activity plus suggested characteristics of an integrated library development system to further enhance the productivity of the library development engineer and an overview of Gould AMI's library development system known as "Accolade" are presented.

2 Library Development Tools

Various tools exist for completing the layout portion of the cell libraries. Some of these tools are computer aided drafting, symbolic layout and compaction, layout synthesis, and module generators. Computer aided drafting systems or hand layout are the simplest of the tools [3],[4]. Basically the designer draws colored polygons that are mapped to specific mask layers for use in the silicon fabrication process. This process is time consuming and is prone to design rule errors.

Symbolic layout and compaction are another set of layout tools [3],[4], [5]. The designer uses a subset of primitives (mainly transistors, wires, and contacts) to formalize the topology of the cell in a symbolic layout. The primitives may be parameterized. As an example, transistors may have parameters for width, length, and the number of contacts to the source and drain. This symbolic layout contains no information about the actual spacing or distance between objects. Once the symbolic layout is defined, the compaction tool using a set of design rules for the process technology then moves the objects as close together as possible without violating the process design rules.

Layout synthesis takes a transistor level schematic representation (present tools use a netlist) and converts it into a symbolic layout using place and route algorithms [2], [9]. The symbolic layout can then be compacted to generate mask data compatible with the design rules. This results in a significant time savings as the layout of cells is completely handled by the synthesis system. Of course this method will benefit from human interaction to control how the cell is built, especially for analog cells where capacitance and crosstalk is of great concern.

Module generators are composed of procedural code that defines the cells. The code is parameterized to handle possible variation of the cell such as the number of inputs, types of control logic, the number of memory elements, etc. Some sophisticated generators take into account delay, and area considerations. Module generators require coding experts or programmers and are difficult to debug and maintain [6]. On the other hand they work well for structures containing regular repetitive blocks such as RAMs, ROMs, and PLAs. Also this tool makes an attempt at integrating the whole design task of layout, extraction, verification and data generation.

As mentioned previously, other tools exist to aid in the design of the cells. Gate sizing algorithms are used to optimize performance and area parameters. One such algorithm is given in [7]. Mask data extraction tools are a valuable aid in the library development process. These extractors are useful for design rule checking, layout versus schematic checks, netlist extraction and the extraction of electrical parameters for use with circuit simulators [3], [4], [8].

Extracted data such as cell netlists and capacitance loading information can also be useful in the development of the library datasheets. Propagation delay values for the cell to be used for verification, logic model generation, and datasheets can be obtained using circuit simulators [3].

The circuit simulators contain models that allow the user to model the behavior of transistors and other physical devices in silicon for a given process technology. A netlist

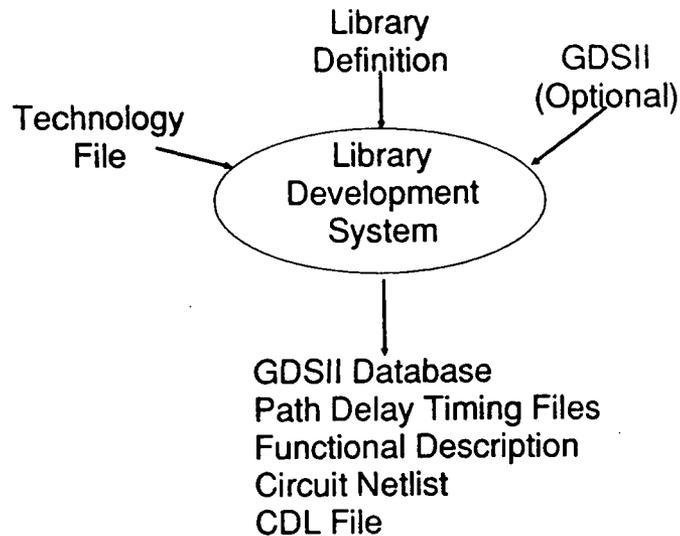


Figure 1: Library development System

of the cell including extracted parasitic capacitances and waveform stimuli can be sent to the simulator. The simulator then performs an incremental transient analysis yielding information about the changes in the node voltage and device currents as a function of time. From this data the delay from a switching input to a change in the output can be determined. Adding different load capacitances to the circuit can be used to characterize the propagation delay as a function of capacitance loading.

3 Library Development System Features

The CAD tools reviewed in the previous section have yielded significant productivity gains for the library development engineer. Further productivity gains can be realized as these tools are integrated into a single development system optimized to meet the specific needs of the library development engineer. Figure 1 illustrates the objectives of such a system. The inputs to the system would include a technology file and the library definition. Optionally, a GDSII reader would be desirable to permit data for cells that were not developed on this system to be entered into the system. The library development system using data from the technology file together with the cell library definition would then output the data required to support a library released for use with various engineering workstations. The output data to support the libraries would include the following:

1. Physical layout data in GDSII format,
2. Datasheets including cell schematics and icons, timing diagrams and cell parameter specifications, etc.,
3. Workstation path delay timing files,

5.1.4

4. functional description,
5. Circuit netlist, and
6. CDL (Circuit Description Language) file for use in Layout versus Schematic (LVS) checks.

In the following sections we will review some of the desirable characteristics and features of this library development system.

3.1 Input Features

The user interface is one of the more important characteristics of the system. This interface should be designed with the human being in mind and adapted to the human style of doing the task [1]. Therefore it is proposed that graphical data entry be used for most of the interfacing. This allows IC design engineers to focus on the engineering aspects of the library development task instead of the procedural code to implement the library cell generators, etc. Efficiencies gained through graphical data entry permit the design engineer to efficiently perform "What-If" analysis in evaluating alternative library designs.

The schematics for the circuit can be captured graphically using a mouse and screen. Various primitives can be selected such as transistors and a parameter menu can be pulled up to add values of width, length, and other values including variables related to a sizing table or expressions. This schematic now defines the data submitted to the layout synthesis tool. The schematic data can also be used for the logic comparison with the mask layout data as well as for generating cell simulator formats for use in system simulations.

The option should also be provided to allow the designer to enter symbolic layouts using a graphical entry style similar to that used by layout editors. The symbolic layout as entered does not include geometric data. Sizes can be automatically transferred to the symbolic layout from the schematic of the cell.

Icon definition is an important feature to allow for hierarchical definitions of cells. As megacell generators are developed, the capability to design blocks using a graphical interface which can automatically be converted to parameterized code is desirable.

A graphical interface should also be devised for specifying the characterization circuit netlist, input signal conditioners / generators, and the instrumentation probe points to automatically obtain the cell characterization data using a circuit simulator. Several instances of the cell (icon) can be placed with various waveform generators and loads connected to them to setup the characterization conditions.

Delay probe pairs can be placed on nodes to designate timing measurements. These probes can have parameters defining voltage level and direction (rising or falling) of the waveforms to compute the desired measurements. The probes can be assigned names so that the measurements can be identified in the characterization database. Text can also be added to set variables equal to expressions that use the delay measurements from the characterization database.

The circuit simulator netlist and control information can then be extracted from the graphically entered data. This data is then used by the circuit simulator to setup and perform the transient analysis. Results are then extracted from the circuit simulation output and specified computations are completed and returned to the characterization database.

Graphical interfaces should also be developed to provide network and icon information required for datasheets and logic models. Some of the output files may be strictly textual and may only need a textual template. All of the graphical interfaces should be menu driven. Some functions could benefit by the implementation of both menu and keyboard drivers.

Integration of these graphical front-ends with the various design engineering tools to provide for user-friendly input to the library development system can significantly improve engineering productivity.

Another important data input consideration is that the process technology inputs and the cell library inputs should be isolated to provide true process independence. A consistent set of standard library cell definitions and models is important. Once the library data has been entered for one technology subsequent libraries can be developed to support different process technologies by updating the process technology file and then using the library development system to produce the new library of cells.

3.2 Operational Features

In operation the library development system is an integration into a single package of many of the different tools which are available. For example, this means combining such tools as the following into a single design environment: gate sizing algorithms, layout, verification, simulation, and data generation and formatting tools. The preferred layout tools for library cell development are layout synthesis and compaction tools. Layout synthesis provides for the auto place and route of the symbolic layouts. Several modes may be needed to control the layout synthesis and compaction of the cells. Decisions have to be made as to whether the cell will use double or triple layer metal and which metal levels will be used for routing the signal and power ports. It is also desirable to be able to place layout constraints on the cell such as preventing certain signal lines from crossing.

Since instances will arise where the designer needs control over the placement and routing of certain cells, a symbolic layout editor needs to be provided. The compactor is useful in producing area efficient layouts for a given technology file or set of design rules. Another option for the compactor is to be able to define symbols for existing mask blocks which can be placed into the symbolic layout. These blocks will then be placed into the cell mask data without being compacted. The process independence created by using the symbolic layout reduces the amount of rework required when migrating cells to a new technology. A hierarchical compactor is required to handle the large number of objects involved in creating such megacells as RAMs, ROMs and PLAs. Another useful feature of the system is the ability to use the symbolic layout generators to develop the different memory sizes.

5.1.6

Tools for the verification algorithms also need to be integrated into the system. A design rule checker should be added to verify the correctness of the mask data according to the design rules contained in the technology file. A layout versus schematic checker to verify that the mask data agrees with the schematic should also be included.

Access to various simulation tools is an important feature of the library development system's design environment. For example, a circuit simulation updated to reflect layout parameters such as capacitance loading is needed to provide data for the logic models and cell datasheets. Graphical interfaces to setup the netlist and control functions as discussed previously in the input features section greatly enhances the design engineers productivity. This interface should permit the specification of waveform generators, output loading, nodes for delay measurements, voltage levels that designate valid level changes for the node, and calculations of the required delay measurements. A software routine should then translate the setup and control information and the parasitic parameter data extracted from the mask into a transistor level format readable by the circuit simulator. As the simulation is completed, a delay extractor retrieves the requested data, performs necessary calculations and stores the data in the characterization database. Other cell data such as node capacitances, area, device count, leakage currents, power consumption can be stored in the characterization database.

The system should provide for both interactive and batch mode operation. Since the tools being integrated in this system can run independently, the interactive mode is easily implemented. This mode is most useful for the development of new designs. The batch mode is used for the production of previously defined libraries in a new process technology. A restart capability at various points through the library development process is important. For example, with the restart capability an existing library can be re-characterized with new models without having to re-synthesize and compact all of the library cells. Similarly, if the datasheet templates are changed for a library, new datasheets can be provided without a new compaction or characterization. This can provide a significant time saving for many library maintenance tasks. With restart capability comes the ability to read in existing mask data and then use the library development system to characterize these cells and provide updated datasheets.

3.3 Output Features

The final tool to be added to the system is the data format generators. These generators are used to format the cell data into the various formats required to complete the formal release of the library. The mask data needs to be written out in a common format such as GDSII along with the pin locations which can be read into the place and route tool. In additions to the GDSII layout database, the circuit netlist and control data should be included as a system output file to be released with the library. This will allow library users to perform additional simulations on the cell as required by the application.

It is also necessary to update the logic simulation models. Some logic simulators accept an update file (capacitances/delays) that incorporates the delays into the simulator models. These update files can be generated directly from the characterization database.

If the external logic simulator requires a more complex update file then a template can be designed with variable designators that can be filled in from the characterization database. Other simulators require manual entry of the delay information and yet these files will be beneficial to the IC design engineer performing the data entry.

Another important set of forms is the datasheets. Again users can define a template with variables for the cell specifications which the system can then replace with symbols and values as the cell is created in a given technology. The objective is to have the library development system output the layout data, simulation models and all supporting data files in the proper format required by the user of the library.

3.4 Open System Features

The library development system needs to be user-extendible because of the many circuit and logic simulators and other external as well as internal design and layout tools that can use data from or be part of this system. The user needs access to the internal databases to be able to format data for use by other tools. Also as improvements are made in various design tools it is desirable to have an open architecture that allows for the integration of different and/or additional tools.

As these changes are made the user needs the capability to create the necessary graphical interfaces and/or textual templates to maintain the user-friendly environment. High level functions should be available to make this task easy. The integration of new tools suggests the need for a programming language capability with high level functions such as program caller and data retrieval functions. A standardized programming language such as LISP or C is preferred.

4 The Accolade System

The Accolade library development system is an integration of various commercially available CAD hardware and software products together with Gould AMI Semiconductor's proprietary software. This system realizes many of the features outlined above. A high level block diagram of the Accolade System is shown in Figure 2. Accolade does not currently support layout synthesis but it does have a one dimensional constraint graph compactor. Plans are underway to add a hierarchical one dimensional constraint graph compactor.

Data entry is accomplished using graphical entry for the schematic, icon, symbolic layout, timing schematic (the circuit simulator interface), datasheet symbols, and the datasheet schematic. A technology file defines design rules, electrical parameters, and transistor level models. A gate size file is created using algorithms to update transistors on the cell schematics by referencing variable names. The symbolic layouts are then updated from the schematics.

Accolade has schematic and mask data extractors to perform design rule checks and logic comparisons. As previously mentioned, the timing schematic defines the delay charac-

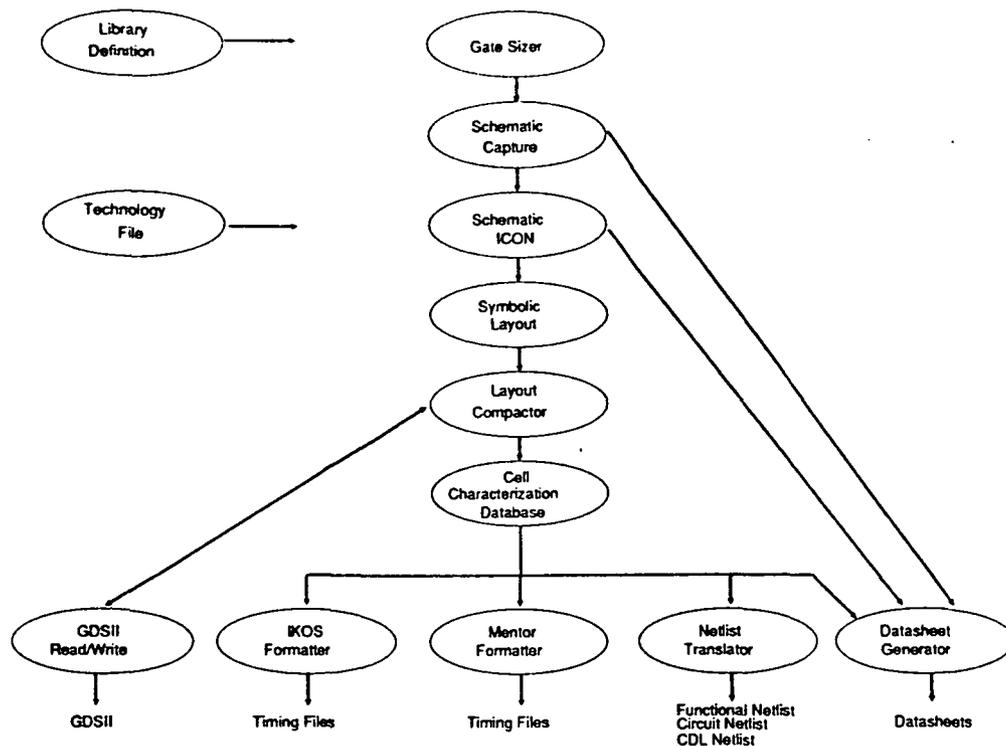


Figure 2: Accolade System Diagram

terization file for the circuit simulation, delay measurement extraction and timing calculations. Instead of using the probe pairs suggested earlier the Accolade delay measurements are described in textual form. After the simulation and delay extraction, a characterization database is created.

The schematic and mask extractors are used to develop a generic network database from which Accolade can create spice-like cell definitions as well as the functional cell definitions for use in creating a chip netlist. Accolade outputs the layout data in GDSII format. The system also can read a GDSII database and extract electrical parameters for use in characterizing the cell and providing datasheets. Logic model update files are formatted for IKOS simulator as well as Mentor's simulator with the 8.0 operating system. The Mentor file is produced with the aid of a text template.

Datasheets use a template where the datasheet schematic and data from the characterization database are merged with the template to produce the actual cell datasheet.

Restart points in the program are made by saving the mask data and the characterization database to files. The mask data can then be retrieved to re-characterize the cell if changes are made in the electrical parameters. Similarly the characterization database can be retrieved to permit a reformatting of the datasheet without the need to start at the beginning of the process.

User extensions to add interfaces, to add new data generators, or to integrate programs are implemented using the Common LISP [10] language.

5 Summary

This paper reviews various tools available to support the library design engineer and suggests desirable features of an integrated library development system. Gould AMI's Accolade library development system has incorporated many of the suggested features. Productivity gains of a factor of two have been realized using the Accolade system as compared to the previous library development system. Predominant factors in the realization of the productivity gains are as follows:

1. graphical data entry versus development of procedural code,
2. system integration from technology file inputs to GDSII and datasheet outputs, etc.,
3. process independent library definition and
4. system restart capability.

Due to Accolade's open architecture/database and the objected-oriented language implementation the systems lends itself to future upgrades.

References

- [1] McFarland, Michael C., "The Social Implications of Computerization," in Proc. of the 26th Design Automation Conference, pp. 129-134, June 1989.
- [2] Chen, Chao C., Chow, Shaw-Lin, "The Layout Synthesizer: An Automatic Netlist-to-Layout System," in Proc. of the 26th Design Automation Conference, pp. 232-238, June 1989.
- [3] Mavor, J., Jack, M.A., and Denyer, P.B., *Introduction to MOS LSI Design*, Addison-Wesley Publishing Company, pp. 166-178, 219- 220, 1983.
- [4] Preas, Brian, "Introduction to Physical Design Automation," in Physical Design Automation of VLSI Systems, edited by Brian Preas, and Michael Lorenzetti, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, pp. 1-29, 1988.
- [5] Wolf, Wayne H., and Dunlop, Alfred E., "Symbolic Layout and Compaction," in Physical Design Automation of VLSI Systems, edited by Brian Preas, and Michael Lorenzetti, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, pp. 211-281, 1988.
- [6] Lacroix, Didier, and Menkis, Sarah, "An Interactive Graphical Approach to Module Generator Development," in Proc. of the Custom Integrated Circuits Conference pp. 30.1.1-30.1.5, May, 1990.

5.1.10

- [7] Richman, Bruce A., Hansen, James E., and Cameron, Kelly, "Deterministic Algorithm for Automatic CMOS Transistor Sizing," *IEEE Journal of Solid-State Circuits*, Vol. 23, No. 2, pp. 522 - 526, April 1988.
- [8] Szymanski, Thomas G., and Van Wyk, Christopher J., "Layout Analysis and Verification," in *Physical Design Automation of VLSI Systems*, edited by Brain Preas, and Michael Lorenzetti, The Benjamin/Cummings Publishing company, Inc., Menlo Park, California, pp. 347-407, 1988.
- [9] Ong, Chong-Leong, Li, Jeong-Tyng, and Lo, Chi-Yuan, "GENAC: An Automatic Cell Synthesis Tool," in *Proc. of the 26th Design Automation Conference*, pp. 239-243, June 1989.
- [10] Steel, Guy L. Jr., *Common LISP The Language*, Digital Equipment Corporation, 1984.

GaAs VLSI For Aerospace Electronics

G. LaRue and P. Chan
Boeing Aerospace and Electronics
High Technology Center
P.O. Box 3999 MS 7J-56
Seattle, WA 98124-2499

1 Introduction

Advanced aerospace electronics systems require high-speed, low-power, radiation-hard, digital components for signal processing, control, and communication applications. GaAs VLSI devices provide a number of advantages over silicon devices including higher carrier velocities, ability to integrate with high performance optical devices, and high-resistivity substrates that provide very short gate delays, good isolation, and tolerance to many forms of radiation. However, III-V technologies also have disadvantages, such as lower yield compared to silicon MOS technology.

Achieving very large scale integration (VLSI) is particularly important for fast complex systems. At very short gate delays (less than 100 ps), chip-to-chip interconnects severely degrade circuit clock rates. Complex systems, therefore, benefit greatly when as many gates as possible are placed on a single chip. To fully exploit the advantages of GaAs circuits, attention must be focused on achieving high integration levels by reducing power dissipation, reducing the number of devices per logic function, and providing circuit designs that are more tolerant to process and environmental variations. In addition, adequate noise margin must be maintained to ensure a practical yield.

2 Applications

Specific applications of GaAs ICs are in fiber optic communications and digital signal processing. The use of fiber optics on board aircraft and spacecraft provide significant reductions in weight. GaAs electronics have achieved fiber optic data rates well beyond 1 Gigabit per second. GaAs circuits can also benefit aerospace applications in the high speed data processing by occupying a smaller volume, and reducing power dissipation and thus saving weight. Although ECL technology can come close to the speed of GaAs, its power dissipation is much higher. CMOS technology can be used in some applications by processing data in parallel at the expense of larger volume. Some applications require low latency and must be performed at a high data rate, thus eliminating parallel solutions entirely. ECL technology can come close to the speed of GaAs, but has higher power dissipation.

3 Floating Point Multiplier

We designed a 32-bit floating point multiplier to investigate the yield and performance of GaAs VLSI for applications in digital signal processing. With over 10,000 equivalent gates, the multiplier approaches the current complexity limits of GaAs. It also provides a good example of a GaAs VLSI integrated circuit targeted for aerospace applications.

The multiplier accepts normalized 32-bit floating point numbers expressed in the IEEE Standard 754, version 8.0 or 10.0 single precision format[1]. GaAs 1 micron E/D MESFET technology was chosen because of the maturity of the fabrication process for LSI production. Operation over the full military temperature range is required.

4 GaAs Logic Family Considerations

There are several logic families that are commonly used to design GaAs E/D MESFET circuits. In choosing a logic family, we were most concerned about noise margin. At these high integration levels, noise margin must be higher due to increased device variations, power-bus noise and crosstalk on signal lines. This high noise margin must be sufficient over the entire military temperature range to ensure adequate yield. We also wanted single supply operations.

Families that meet the above criteria are source-coupled FET logic (SCFL)[2] Gain FET logic (GFL)[3] and FET-FET logic (FFL)[4]. FFL was invented at Boeing and has a better delay power product than either GFL or SCFL. Although SCFL has a much higher power dissipation, it can perform very complex logic functions which include implementing a full adder with two gates and providing the sum and carry outputs in only one gate delay each.

Adder Type	Device Count	Sum Delay.ps	Carry Delay.ps	Adder power.mW	Wallace Tree delay.ns	Wallace tree delay power product
Nor FFL	91	480	290	9.6	1.9	18.2
Complex FFL	34	585	282	2.1	2.0	4.2
SCFL	44	460	450	2.4	1.8	4.3
Complex GFL	39	750	395	1.8	2.6	4.7

Table 1: Comparison of Different Adder Designs

5 Full Adder Design

The most important building block of the multiplier is the full adder. The design of the full adder is a determining factor in the final speed and power dissipation of the chip. Figure 1 shows the schematic of an all NOR implementation of a full adder. It requires 12 gates

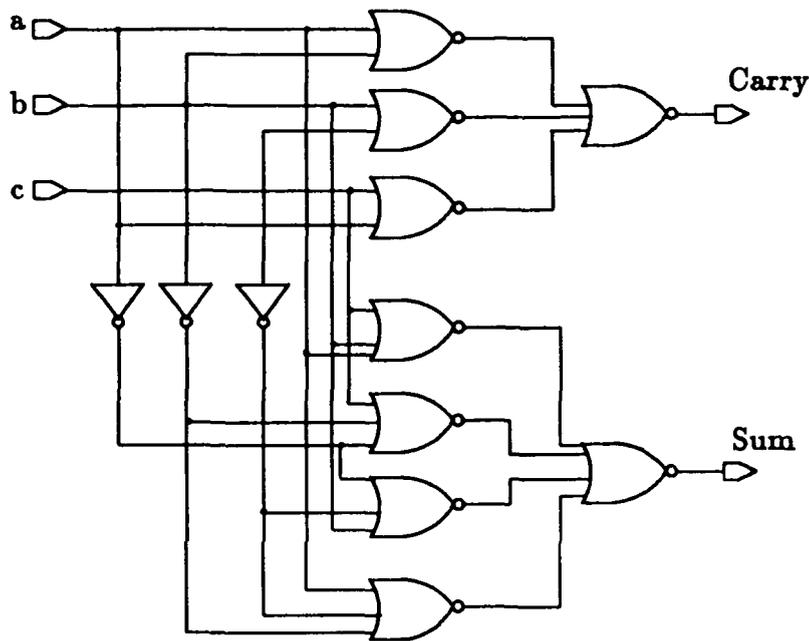


Figure 1: NOR Implementation of Full Adder

to implement and the carry and sum are generated in 2 and 3 gate delays, respectively. This design of a full adder result in high device count and high power.

A complex AND/NOR gate full adder was designed using FFL and GFL gates (fig. 2). This full adder requires only 2 complex gates and the carry and sum are generated in 1 and 2 gate delays respectively. As a comparison, an SCFL full adder was designed (fig. 3). The SCFL adder was designed to have comparable power dissipation to the complex FFL and GFL adders.

It takes 3 sum delays and 1 delay delay for the Wallace adder tree to reduce 13 partial products to 3. Table 1 shows device count, the sum and carry delay, power dissipation, as well as the Wallace tree delay for the different adder design under nominal processing

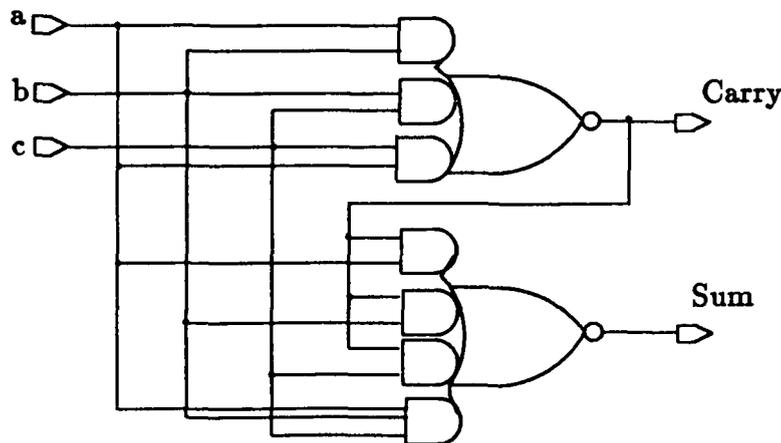


Figure 2: Complex AND/NOR Full Adder

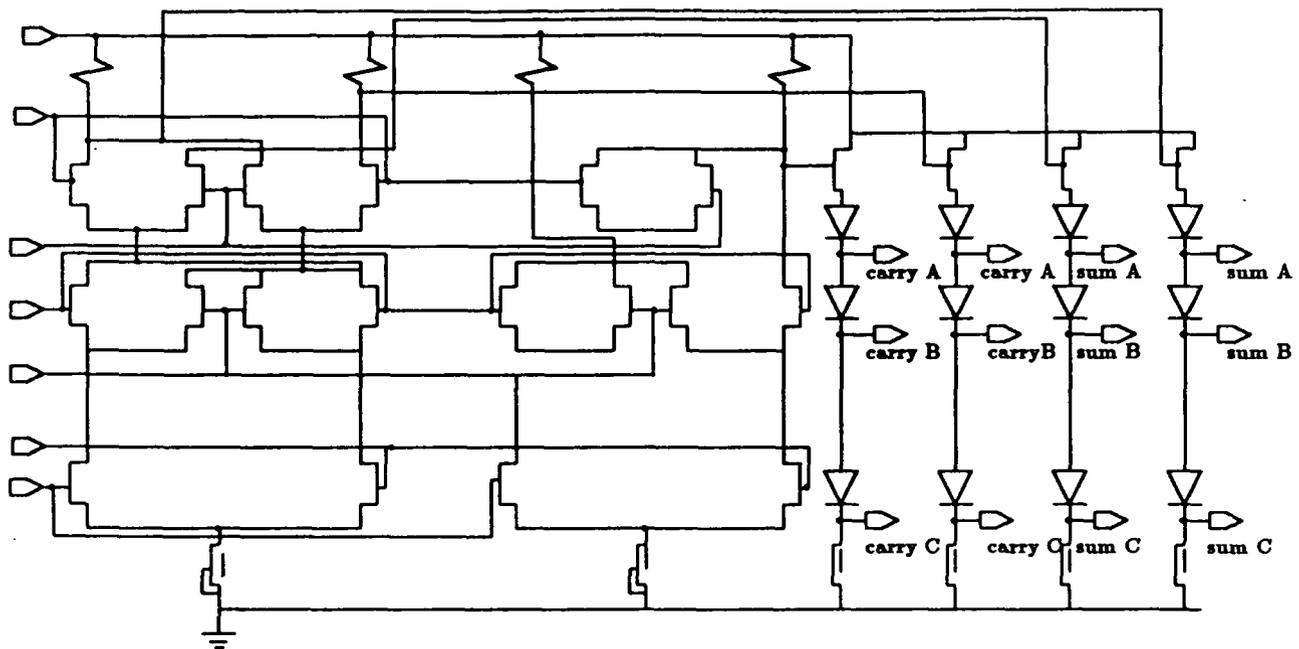


Figure 3: SCFL Adder

conditions.

The all NOR implementation has much higher device count and power than the other designs.

We choose FFL with the complex gate full adder approach to implement the multiplier. FFL has the lowest delay-power product for the Wallace tree and the smallest device count. GFL has 15% lower performance than FFL with comparable layout area. SCFL is comparable in delay-power performance to FFL but requires a substantially larger layout area. The area is larger mainly because SCFL is a differential logic family and requires two interconnects between gates instead of one.

6 Multiplier Architecture

Figure 4 shows a simplified block diagram of the floating point multiplier. The chip has a 4-stage pipeline architecture employing high-speed pass-transistor pipeline latches. The 32-bit inputs are screened for invalid inputs and the signs of the numbers are multiplied by an exclusive-or gate. The exponent adder performs addition of the two 8-bit exponents and outputs the sum, as well as the sum incremented by 1 for the possible right shift of the 24-bit mantissa result. The modified Booth Encoder produces a 69-bit code from the multiplicand.

Thirteen 26-bit partial products are generated by the partial product generator and are reduced to three partial products by the first Wallace Tree. The second Wallace Tree further reduces the three partial products to two and the look-ahead-carry generator

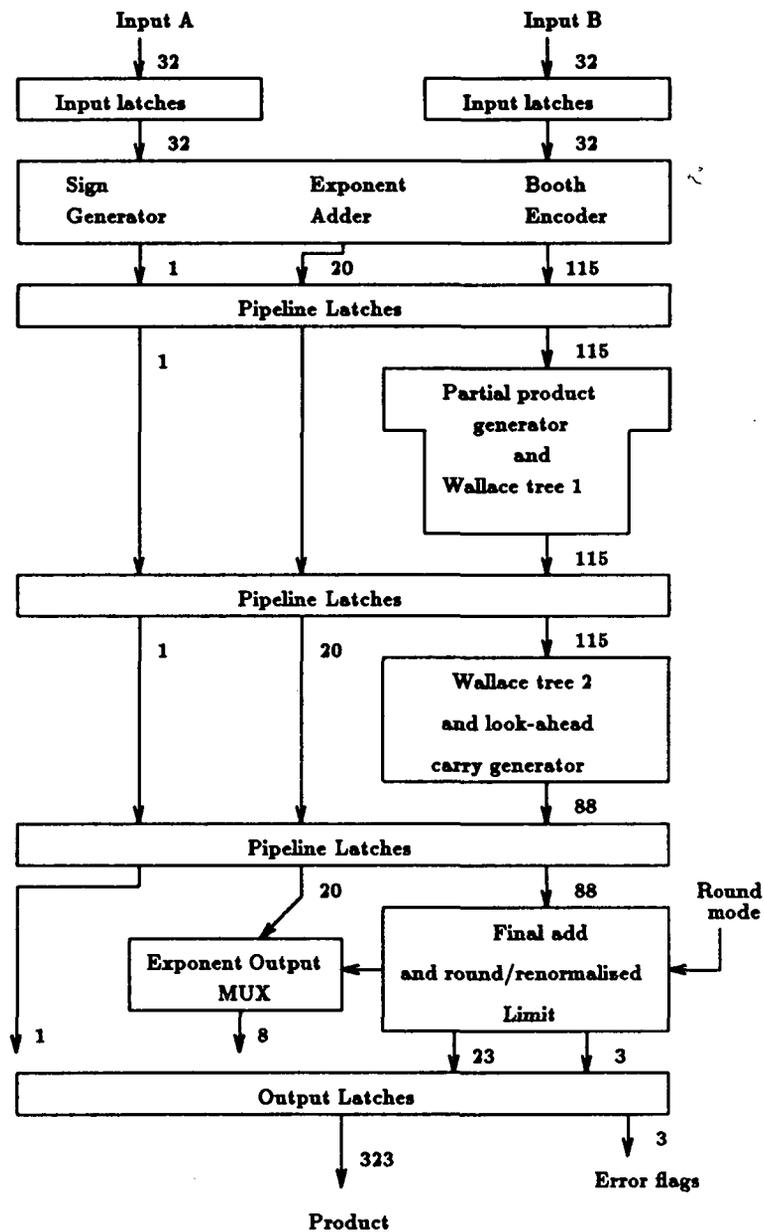


Figure 4: Block Diagram of Floating Point Multiplier

generates the carries for the final adder. Two rounding modes are available: round to the nearest and round toward zero. The result from the final adder is rounded, checked for overflow and underflow, and renormalized into the 23-bit mantissa product. The correct 8-bit exponent result is then chosen and the 32-bit (sign bit, 8-bit exponent, and 23-bit mantissa) product is obtained.

7 Simulated Results

Automatic placement and routing of FFI standard cells was used to lay out the circuit. Interconnect capacitances were then extracted. The critical paths were resimulated and found to be less than 3 ns between latches. Operation near 350 MFLOPS is expected for TriQuint Semiconductor's 1 micron E/D MESFET process. Power dissipation will be under 4.5 W. The die size is about 7.5 mm by 8 mm with about 40,000 devices.

8 Conclusion

The design of a GaAs VLSI floating point multiplier was described. The chip is expected to perform multiplication at data throughput rates of about 350 MHz when the pipeline latches are enabled. With the pipeline latches disabled, the multiplier will operate at about 110 MHz.

The high-speed, low-power and radiation hardness of the multiplier will demonstrate the benefits of using GaAs VLSI for aerospace electronics.

References

- [1] "IEEE Standard for Binary Floating Point Arithmetic," ANSI/IEEE Std 754-1985, New York, The Institute of Electrical and Electronics Engineers, Inc., 1985.
- [2] T. Takada et al., "A 2 GB/s Throughput GaAs Digital Time Switch LSI using LSCFL," IEEE Trans. on Electron Devices, Vol. ED-32, pp. 2478-2483, 1985.
- [3] G. M. Lee et al., "A High Performance, Low Power GaAs Gate Array Family," VLSI Systems Design (USA), Vol. 8, No. 8, pp. 24-25, 28-30, July 1987.
- [4] G. S. LaRue, T. J. Williams and P. Y. Chan, "FET FET Logic: A High Performance, High Noise Margin E/D Logic Family," GaAs IC Symposium, Oct. 1990.

New Direction In CAD Tools

S. Peterson
Mentor Graphics Corporation
5295 South 300 West
Murray, Utah 84065

Abstract-

Chip and software development communities are concerned with the run-away pace of chip development and the monumental problems which can quickly accumulate and overwhelm those charged with the task of IC development. This "weight" must be transferred from the shoulders of the human developers to the software tools which assimilate and control the defining data of the development process. This paper details many of the problems which are being resolved by reviewing the obstacles and resultant advancements which the industry has made. Specifically, it will describe many of the things which are being done in one company to provide this much-needed relief.

1 Introduction

Our industry's ability to develop complex ICs has grown at colossal rates as witnessed by many of the large devices currently in production. This ability to pack functionality on a chip continues to increase, limited only by the imaginations of dreamers. It's promising to note that researchers are optimistic that there are still plenty of advances to be made which will propel this industry toward new heights. Similarly, there are indications from marketing groups that as long as the technology keeps coming mankind will be able to apply it.

Obstacles however, are combining to make the task of engineering like products difficult at best with many of the development tools available today. We will address some of these in detail.

Many software tool developers are stepping up to the challenges by becoming involved in the development process. This is often accomplished through structured alliances between chip development houses and software tool developers and has, of course, been happening all along. But those building the tools have realized that the industry has moved so rapidly that unless there is a great deal of interaction with the engineers on the front lines it will be impossible to create the next generation of wonder chips. Therefore, new partnerships are being formed now and will continue to be needed to responded to the needs.

2 Outlining Some Deficiencies

There are several factors which substantially impede a development team's talent. Bear in mind that development of large chips is no longer the domain of one or two people.

Typical groups today are staffed by dozens of individuals. A team effort is needed to engineer today's system-level chips. Coordination of all this talent falls on the shoulders of a core of managers, each overseeing specific pools of responsibility. This partitioning continues on up the management tree. The hierarchy of project management begins to resemble the structure of the chip or the object of the design itself.

Note that the design process borrows heavily from the discipline of management. Accessing, controlling, and directing human engineering resources will always be an arduous task. But, the application to recorded engineering information should not be nearly that difficult, and yet in many engineering departments it remains a major stumbling block. Management of design data itself, then, is an item to be addressed.

There are other areas which have complicated chip design and layout as well. The following categories merit special attention due to the unique problems which each represents.

2.1 Process Evolution

Research and Development has been benevolent to those with aspirations of ultra-complex chips. Among others, improved lithography [1] and Statistical Process control have pushed the processing edge to where electrical lengths are less than 0.5 microns, and interconnect pitch with multiple levels of metal has crossed 2 microns [1]. Vertical stacking of devices will also soon be available [2].

If all that isn't enough (and it's not) processes have matured to the point where 15 to 25 reticule layers are capable of yielding a twin-well CMOS process which sports bipolar, electrically-erasable, polysilicon fuse, and high-voltage capabilities. Clean rooms have become so clean that there have been instances where all chips from a given wafer were fully functional.

Such amazing advances will of course continue and become more prevalent.

2.2 Complexity

Only ten years ago, a chip in a 40-pin DIP with five or six thousand transistors was considered elegant. The only thing more spectacular was the spontaneous celebration observed when the chip came out of fab and worked the first time. It happened, but not as often as hoped.

At this writing, there are many organizations with operational chips containing one to two million transistors and operation above 100 Mhz. There are devices currently being developed with more than four million devices on board, with that figure continuing to mount. These are digital "systems" chips. All you add is power (very little of it) and some I/O. Other groups have been successful in making complementary analog and digital systems, such as high-speed modems, coexist harmoniously on the same die.

This industry has even had to develop its own packaging materials and methods to answer the advancing needs of interconnect which a device needs to talk to its neighbors.

2.3 Resolution

The dimensions at which circuits are laid out have begun to outstrip needed computer resources. Shape boundaries are still being stored using integer mathematics, but as geometries begin to take on all-angle attributes to squeeze circuitry into ever small areas, floating point operations will increase, as will the working storage required to accommodate necessary algorithms or methods. Verification operations are already floating point intensive.

Shrinking processes started to dictate special rule sets several years ago. Two significant example issues are optical reflections during exposure, and mechanical stress within each of the physical materials deposited or grown during the process. Stress has become a significant reliability issue, and new design rules have been "invented" which cannot be accommodated in many of the current verification tools. They've simply run out of gas.

2.4 Performance Prediction

At smaller geometries, the ability to predict or simulate the performance of a small sub circuit, let alone the whole chip, has become very difficult. Indeed, it is the opinion of some that there is not enough time available to perform a thorough simulation. The engineer sometimes takes the risk that his circuit will be 'fast enough', and pushes the design on through reticule generation and fabrication to verify his hypothesis. That's an expensive test. If he has a lot of experience, sometimes he is lucky. Most of the time he is not. Keeping track of all critical paths in one's head is just impossible.

Because performance of datapath chips is often measured in MIPS, it becomes clear that the engineer needs a tool which will allow him to model sub-micron devices and associated parasitics accurately, so that he can optimize paths. This will allow him to squeeze performance from circuitry which has often been referred to as "design margin".

2.5 Time to Market

It seems that there are many industries which tend to drive themselves. IC development is one of those where the time to develop and get a device to manufacturing must be minimized. Redesigns and delays due to tools, once commonplace and accepted, now have no place in the process. If a chip cannot be delivered on time, the project's end product will usually suffer a great deal of lost market share. That goes to the bottom line as lost revenue.

3 Providing Solutions

Having set the stage with many of the problems which need to be solved, it is now possible to outline some of the possible remedies.

There are many vendors attempting to provide solutions to the problems mentioned herein. Speaking for these other developers would not be possible, being unfamiliar with

some of their key philosophies. A reasonable approach would be to discuss the issues as they are being addressed within Mentor Graphics Corporation. This will provide insight into treatment of the industry's needs from the tool developer's perspective - specifically that of the staff of Mentor Graphics.

3.1 Frameworks

A little over three years ago, Mentor Graphics began to seriously consider frameworks. Many in the industry felt that chip complexity was beginning to level off. Several discussions with key customers, however, drove home the point that soon the complexities of planned chips would exhaust the capabilities of current tools. Mentor Graphics not only realized this, but also faced early the grim realization that some of its planned products and future enhancements would not be able to meet many of these needs.

This was when Mentor Graphics began to develop the idea that a heavily-supported environmental approach might fill this need. A decision was made that a "from the ground up" approach was the only realistic one, and that a new language would be required to meet these needs. In this environment, individual tools could operate autonomously, yet communicate effectively with other tools at levels of abstraction defined by the user. Today, there is a great deal of activity in defining frameworks, but Mentor Graphics has made significant headway not only in defining the system, but also in establishing the standards and methods to make it work. And it's about to be released. The result is a framework called the **Falcon Framework™** for concurrent design. Some of the highlights follow.

Motif™ is the graphical user interface standard developed by the Open Software Foundation (OSF) which brings a common "look and feel" to all tools intended to operate within the Falcon Framework. The compliance standards [3] conform to the **OSF/Motif™ Style Guide** (Current revision). This approach eliminates much of the need to relearn when going from tool to tool, and when new tools are added to the framework.

History has shown that those industries which provide their goods and services with an "open" attitude are usually the ones which take root and grow. This has been demonstrated many times in the hardware and software areas with which most will be familiar. It seems obvious that if any framework is going to be successful, one must realize that openness is paramount. There will be big players, and there will be small niche players. To that end, Mentor Graphics has established an **Open Door™** policy providing any software tool developer or customer all he or she needs to be successful in integrating his tool into the framework.

A **Design Management Environment™** allows users to control the design as well as its management process through an object-oriented paradigm. Tools and data are created as objects in which version control can be applied to both. The historical benefits as applied to design chronology are indispensable.

The **Falcon Framework™** allows the user community to develop tools to assimilate objects or groups of objects in one environment. The decision support paradigm allows the engineer to rapidly determine how a design decision will affect the parameters of another part of the design. For example, thermal aspects of a chip need to be considered within

the chip, as well as in the board where it lives. While not easy to do in the past, this provides the capability of transcending such boundaries.

The **Bold™** Browser provides an integrated, on-line documentation delivery system. All 65,000 pages of Mentor Graphics' documentation are provided. The user can also create his own documentation, and include it with the system. Linked Hypertext is part of its capability.

3.2 Object-Oriented Methods

Already mentioned, the decision was made to embrace a language which would facilitate the framework thrust. The decision to go with C++ was almost automatic. An object-oriented approach was needed to support Falcon's virtues, and yet performance was paramount. Project for example, the fact that multi-million-transistor chips might involve more than a billion polygons or shapes, not to mention the volume of engineering data collected to form specifications, netlists, simulations, and the like.

There are quite a few tool developers who have felt that going to C++ represents too significant an investment: that C++ is not worth the re-education process which would be required. Most of them are still developing products in the same, standard languages today. There is no doubt that the cost of embracing C++ is substantial. Unofficially, Mentor Graphics has spent more than seventy-five million dollars on Falcon, much of that has gone into the effective adaptation of the object-oriented capabilities of C++. But this learning process has paid off in the staff's ability to write reusable code. A discussion of some of the capabilities and features of C++, as applied to design, is in order.

Variables defined in most modern languages are typed. In procedural languages, such as C, it is the programmer's responsibility to assure that improper procedures are not applied to variables. Misapplication, at the very least, results in incorrect data but often causes system crashes, and debugging is difficult.

C++ allows one to build a class, which when instantiated represents an object. A `logic_elem` (logical element) might represent a class of objects, for example, while `comb_elem` (combinational element) and `mem_elem` (memory element) would both be subclasses of `logic_elem`. NOR and NAND gates are elements which would belong to the `comb_elem` class. An entire hierarchy of gates could be developed, but that need not be done here. Other references [4,5,6,7] can provide the needed details to interested parties.

A key attribute of C++ is that rather than applying a procedure to an object of concern, operative messages are sent to the object to perform functions. Through its classification, the object has been given the intelligence which it will need to interpret or understand the message which has been sent to it. This capability exists because the object always carries its class with it, the same way that a human being carries its own gender information. To do this, it is necessary that an object's class contain not only the instantiated information which may make it unique, but also contains the procedures which will be used to manipulate its data. All of this information is encapsulated as part of the class.

The encapsulated information is broken down into categories of accessibility. One might define functions which are private, as well as functions which are public. An example of a

5.3.6

public operation on a logical NOR gate would be to place() a gate at a specific location in the physical domain. A private function might be to checkPlacement() to assure that the provided location is allowable. Again, these functions are a defined part of the class of an object.

This represents the briefest of introductions. There are many other virtues to consider[9]. The point of the discussion is to show that object-oriented methods represent a significant departure from the 1980's methods of programming. But application to design data is an area where the management of objects such as cells, gates, views, and similar elements does provide the methods needed to manage such vast amounts of data which are seen even now.

3.3 Verification

Given information presented in the first part of this paper, it might be obvious that there are several capabilities from previous sections which could easily be taken advantage of and applied to the task of design verification. As seen from this perspective, the possibilities take on their own characteristics. Some of the highlights follow.

True hierarchy will allow one to correct a cell which contains errors, rather than having to examine each instantiation of that cell to determine if a cited error is real or redundant. Thanks to object-oriented methods, the process of error reporting will become much more meaningful. Since redundancies will be eliminated, True errors will stand out.

Significantly enhanced polygonal operations are being implemented. Within weeks multiple tools will be available which will provide this extended checking capability.

The **Design Management Environment**TM operating in the Falcon Framework environment mentioned earlier promises to facilitate a great deal of incremental checking which has not been commercially available until now.

3.4 Parasitic Extraction

Chip complexity has begun to severely impact many aspects of chip performance. Diminishing geometries have contributed toward positive performance in areas such as lower l-effectives, gate-oxide thicknesses, power supply voltages, and similar areas. The converse of this, however, is that reduced geometries cause other problems with increased fringe capacitance and interconnect resistance.

Mentor Graphics will soon ship a tool with significant parasitic capacitance and resistance extraction capability. This tool will fill the need of accurate extraction and consideration of parasitics during simulation. Per existing customer communications, it is expected to be of substantial value when designing cells or libraries of cells. It will also facilitate the analysis of global signal nets, such as a clock and asynchronous reset lines. Capacitance extraction is derived from the boolean operations which are already available within the checking environment previously discussed. Resistance calculations are performed based upon optimized heuristic methods to enhance throughput.

Further research continues in an attempt to satisfy the needs of analog and bipolar designers. High resistance and capacitance extraction accuracy, while not critical in digital applications, is an absolute must in the mixed-signal world. Finite element methods, where the customer feels that he needs the accuracy, are expected to provide the needed accuracy of resistance extraction.

Similarly, the high-speed digital developers will soon find themselves in need of tools capable of extracting segmented transmission lines to analyze crosstalk and square up interconnect carrying clock signals. These will certainly have to be provided as well.

4 Conclusions

The information herein has given a representative snapshot of current efforts of IC design and development automation, as seen by those currently involved in the development of these tools. While the software industry is stepping up these challenges, it is aware and concerned with the obvious: keeping up with chip complexity has, to date, only been a hope. The goal is to somehow get ahead of the currently accelerating capabilities. While the few solutions presented herein will hopefully meet near-term needs, recent history has demonstrated that the only way that this can be realized is through significant interaction between the IC design and tool development communities.

References

- [1] J.H. Brannon, "Eximer Laser Ablation and Etching", IEEE Circuits and Devices, 6, #5, pp. 19-24, 1990.
- [2] G.W. Neudeck, "Three-Dimensional CMOS Integration", IEEE Circuits and Devices, 6, #5, pp. 32-38, 1990.
- [3] Mentor Graphics UIMS/Motif Compliance Checklist;;, Revision 1.0.1, 23 August 1990, Mentor Graphics corporation.
- [4] B. Stroustrup, "What is "Object-Oriented Programming?"", Selected Readings, 307-144, pp. 4-1-4-25, 1989.
- [5] B. Stroustrup, "The C++ Programming Language", Addison- Wesley, 1987.
- [6] S.B. Lippman, "C++ Primer", Addison-Wesley, 1989.
- [7] T. Hansen, "The C++ Answer Book", Addison-Wesley, 1990.
- [8] D. Thomas, "What's in an Object?", Byte, 14, #3, pp.231- 240, 1989.

High Frequency Integrated MOS Filters

C. Peterson

International Microelectronic Products
San Jose, CA 95134

Abstract— Several techniques exist for implementing integrated MOS filters. These techniques fit into the general categories of sampled and tuned continuous-time filters. Advantages and limitations of each approach are discussed. This paper focuses primarily on the high frequency capabilities of MOS integrated filters.

1 Introduction

The use of MOS in the design of analog integrated circuits continues to grow. Competitive pressures drive system designers towards system-level integration. System-level integration typically requires interface to the analog world. Often, this type of integration consists predominantly of digital circuitry, thus the efficiency of the digital in cost and power is key. MOS remains the most efficient integrated circuit technology for mixed-signal integration. The scaling of MOS process feature sizes combined with innovation in circuit design techniques have also opened new high bandwidth opportunities for MOS mixed-signal circuits.

There is a trend to replace complex analog signal processing with digital signal processing (DSP). The use of over-sampled converters minimizes the requirements for pre- and post filtering. In spite of this trend, there are many applications where DSP is impractical and where signal conditioning in the analog domain is necessary, particularly at high frequencies. Applications for high frequency filters are in data communications and the read channel of tape, disk and optical storage devices where filters bandlimit the data signal and perform amplitude and phase equalization. Other examples include filtering of radio and video signals. Integrated filters offer obvious advantages in reducing PC board space and eliminating trimming or the requirement for tight passive component tolerances. Also, integrated filters offer advantages over passive filters when it is necessary for the filter characteristics to be programmable. This paper will review techniques for building integrated CMOS filters with a focus toward high frequencies.

Since tolerances of resistors and capacitors available in integrated circuit technologies are typically too large for most applications, circuit design techniques have been developed to take advantage of the tracking of components on a single die. There are two general types of approaches. The first utilizes discrete time sampling and the second approach is continuous-time and utilizes a tuning scheme to compensate for process variation. These two general approaches will be discussed in more detail.

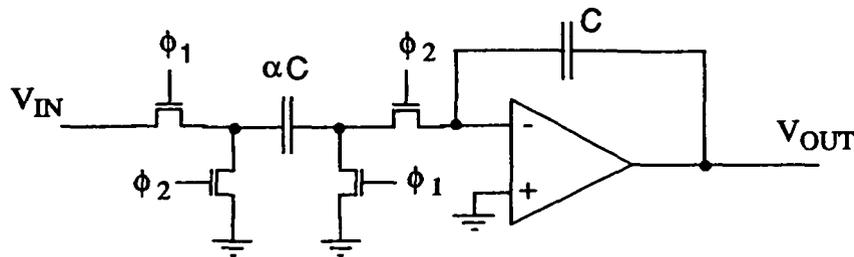


Figure 1: Switched-Capacitor Integrator

2 Sampled Filters

In sampled filters a delay element is implemented by storing charge on a capacitor. In a switched capacitor filter, signals are summed by charge sharing between weighted capacitors. Pole and zero frequencies are set by ratios of capacitors and the clocking rate. An example of a switched-capacitor integrator is shown in Figure 1. The output of the integrator has a transfer function given by:

$$H(z) = \frac{V_{out}(z)}{V_{IN}(z)} = \frac{\alpha z^{-1}}{1 - Z^{-1}} \quad (1)$$

Switched-capacitor technology is now a mature design technique that has been used extensively in audio band communication circuits over the last 10-15 years. This design technique has matured such that sophisticated tools have been developed to automate the design of switched-capacitor filters. Switched-capacitor filters require a floating linear capacitor which adds processing steps that are not needed for standard digital circuits. This may not be a significant burden if the majority of a chip is analog, however more and more mixed-signal designs are dominated by digital.

A new alternative to switched-capacitor filters are switched-current filters [1],[2]. This technique represents signals as sampled currents rather than sampled charge. Signals are simply summed by connecting current mirror outputs together. The delay element is made of a current mirror in which the voltage on the gate of the output transistor is sampled. Figure 2a shows a simple delay element. Figure 2b shows how the delay element can be modified into a switched-current integrator. The output of the integrator has a transfer function given by:

$$H(z) = \frac{i_o(z)}{i(z)} = \frac{Az^{-1}}{1 - Bz^{-1}} \quad (2)$$

These circuits can also be extended to a fully differential topology. Cascades may be added to improve accuracy of the current mirrors. Poles and zeros are set by the ratio of current mirrors and the sampling rate. This technique offers several advantages. First, the holding capacitor need not be linear; gate capacitance works well. Second, amplifiers are

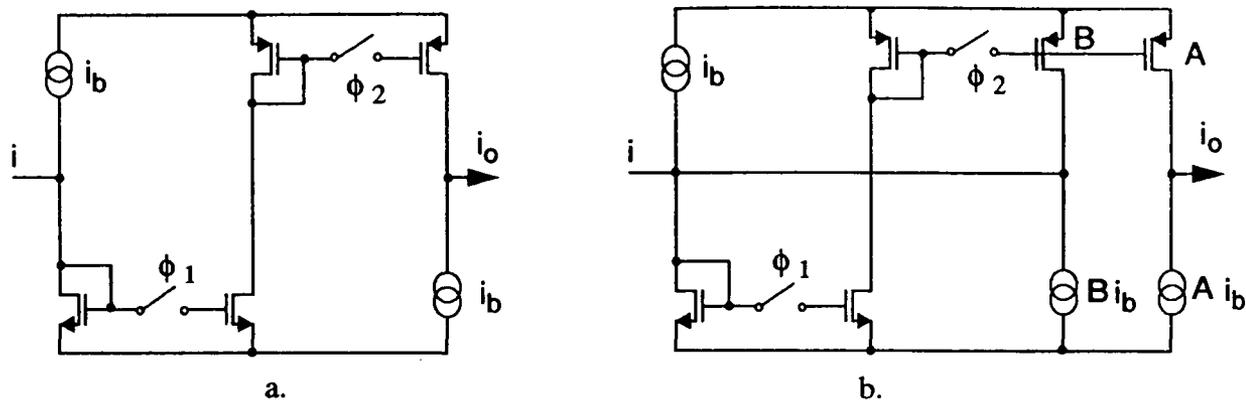


Figure 2: a. Switched-Current Delay Cell, b. Switched-Current Integrator

not needed. This is an obvious advantage for power efficiency and for bandwidth. Third, signals are represented as currents rather than voltages. This allows signal nodes to be low impedance, which maximizes the frequency of parasitic poles. Also, current-mode circuits offer a signal-to-noise advantage, particularly as supply voltages are scaled down.

Sampled-data filter performance is limited by the non-ideal characteristics of its components. In the case of switched-capacitor filters, performance is limited by capacitor matching, charge injection, opamp gain and bandwidth and noise from switches and opamps. Switched-current filter performance is limited by matching of transistors, output impedance of the current mirrors, charge injection on the hold capacitor and noise contributions of the switch and mirror transistors. In both switched-capacitor and switched-current filters, broadband noise is sampled and aliased back into the base band.

Switched-capacitor filters have been demonstrated up to 10.7MHz pole frequencies [3], but practical limits for this technology today are considered to be around 200KHz pole frequencies. Sampling rates are typically greater than 10 times the pole frequency. Otherwise, the anti-aliasing filter becomes complex enough to render the switched-capacitor filter approach impractical. A common rule-of-thumb is to make the opamp gain bandwidth at least 5 times the sample rate. This would indicate that the opamp bandwidth should be at least 50 times the pole frequency. The achievable opamp bandwidth depends upon its capacitive load up to the point where the opamp's own output capacitance greatly exceeds the load it is driving. In a single-stage opamp the gain-bandwidth is determined by the output pole and in the case of a two-stage opamp the output pole must be greater than the gain-bandwidth to achieve reasonable phase margin. For a $1\mu\text{m}$ process an opamp tops out at 100MHz into a 5pF load, not to mention that it is burning about 5mA of current and only has about 40dB of low frequency gain.

Techniques for high frequency switched-current filters are not fully developed yet. The inherent limit would appear to be the sample rate that would allow the sample and hold to settle out sufficiently. Trade-offs have to be made between charge injection error and settling time in sizing the switch and hold capacitor. This issue is common with switched-

capacitor filters, however, switched-current filters are not limited by the bandwidth of an opamp. This approach certainly has interesting advantages and deserves more attention.

3 Continuous-Time Filters

Tuned continuous-time filters have been reported over the last several years. However, they are now starting to receive serious attention in industry. Continuous filters have several key advantages over their sampled counterpart: they do not suffer from a aliasing of broadband noise, they do not require anti-aliasing filters, and they are more suitable for high frequency applications since they do not need to be sampled at 10 or more times the signal band.

There are several ways to implement integrated continuous-time filters. Thin-film resistors can be trimmed by lasers, however this requires an undesirable complexity and increase in production costs. One novel approach published uses thin-film resistors to build a classic active R-C filter. All integrator capacitors are identically digitally programmable and tuned during an initialization sequence under the control of the host processor, thus compensating for the process variation of the capacitors and resistors [6].

Another technique utilizes MOS transistors operated in their linear region to implement a classic active R-C filter. Typically the filter is operated in a fully differential mode to cancel out non-linearities in the MOS resistor (MOS-R). To maintain reasonable linearity the signal swing is kept small, especially if only a 5V supply is utilized. The control voltage for the MOS-R transistors is derived from a phase-locked loop (PLL) using a reference filter locked to a stable reference clock.

To build high bandwidth active R-C filters, high bandwidth opamps are needed. A good rule-of-thumb is to make the unity-gain-bandwidth of the opamp over 50 times larger than the signal bandwidth of the filter. This places a practical limit for MOS active R-C filter today to under 2MHz.

Another approach growing in popularity is that of transconductance-C filters. This technique has the advantage that the unity-gain-bandwidth of the amplifier is the desired filter pole frequency. Signals are summed by connecting transconductance amplifiers, or OTA, outputs together. Figure 3 shows a typical transconductance-C biquad. Optimum high frequency operation can be achieved if all the nodes in the filter that the signal is represented as a voltage are integrator nodes. Thus no high frequency amplifiers are needed. This greatly reduces the amplifier current or increases the filter bandwidth that can be achieved when compared with an active R-C approach.

Since the transconductance of the OTA is affected by process and temperature it can be tuned similar to the MOS-R filter by a tuning PLL. The transconductance can be tuned by adjusting the bias current of the input devices. Large variations in bias current will however degrade the allowable signal swing for good linearity. Other techniques for coarse tuning may be utilized, such as changing the integration capacitor or current gain of the OTA output stage to reduce the tuning range for the bias current. Techniques exist to linearize the input range of an OTA, however they all limit the ability to tune the OTA

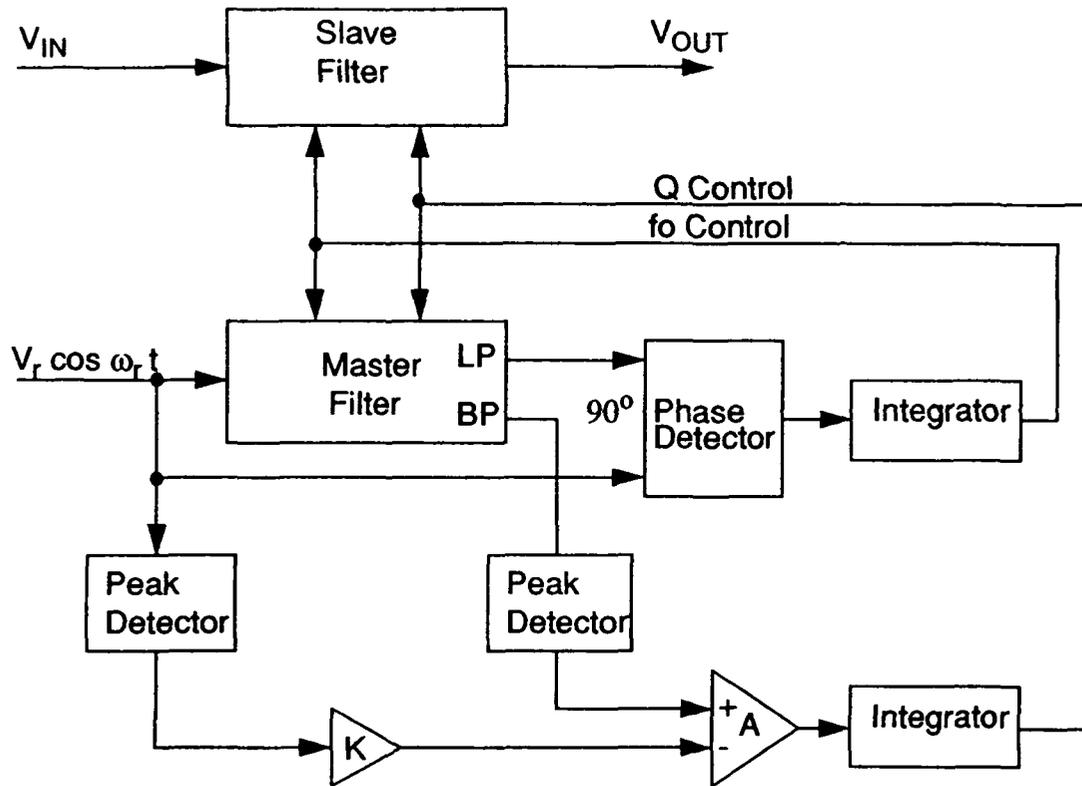


Figure 4: Vector-locked tuning loop

slave filter.

Design of high frequency circuits require accurate modeling of parasitic capacitances and resistances. A challenge that must be faced during the design of high frequency filters is that of the distributed R-C of the MOSFET channel. Current SPICE like simulation models ignore this limitation [13]. One solution involves modeling individual transistors as multiple devices in series.

Experimental MOS transconductance-C filters have been published that have achieved bandwidths up to 90MHz. An example of a commercial application of this technology, is an automatically tuned, programmable filter that has been developed for the read-channel of the disk drives at IMP. This product includes a programmable second-order allpass and a 6th-order Bessel lowpass with programmable pulse-sliming zeros. The filter is digitally programmed through a serial interface. The frequency response can be shifted by changing the clock reference. Pole frequencies may be set up to 30MHz.

4 Conclusion

Pre- and post filtering requirements of sampled filters and high sample rates render sampled filters undesirable for high frequency applications. Opamp based continuous-time filters are constrained by the requirement for stable opamps with bandwidths substantially beyond the filter's pole frequencies. Of the techniques described in this paper, superior high

frequency performance can be achieved by continuous-time transconductance-C filters. Automatic tuning allows for the compensation of process variations and effects of non-ideal integrators. Transconductance-C filters should have as big of an impact on high-frequency filter applications as switched-capacitor filters have had in audio-band communication circuits.

References

- [1] John B. Hughes, "Analogue IC Design: The Current Mode Approach," IEEE 1990, pp. 415-450.
- [2] John B. Hughes, et al, "Switched Currents- A New Technique For Analog Sampled-Data Signal Processing," IEEE International Symposium on Circuits and Systems Proceedings, pp. 1584-1587, 1989.
- [3] Bang-Sup Song, "A 10.7MHz Switched-Capacitor Bandpass Filter," IEEE J. of Solid State Circuits, Vol. 24, pp. 320-324, April 1989.
- [4] F. Op't Eynde, et al, "A 150MHz OTA In 3 Micron CMOS Silicon Technology," IEEE International Symposium on Circuits and Systems Proceedings, pp. 86-89, 1989.
- [5] Germano Nicollini, et al, "High Frequency Fully Differential Filter Using Operational Amplifiers Without Common-mode Feedback," IEEE J. of Solid State Circuits, Vol. 24, No. 3, pp. 803-813, June 1989.
- [6] John P. Roesgen et al, "An Analog Front End Chip For V.32 Modems," IEEE Custom Integrated Circuits Conference, pp.16.1.1-16.1.5, 1989.
- [7] F. Krummenacher, "Design Considerations In High Frequency CMOS Transconductance Amplifier Capacitor (TAC) Filters," IEEE International Symposium on Circuits and Systems Proceedings, pp. 100-105, 1989.
- [8] Peter M. VanPeteghem, et al, "Design of a Very-Linear CMOS Transconductance Input Stage for Continuous-Time Filters," IEEE J. of Solid State Circuits, Vol. 25, No. 2, pp. 497-500, April 1990.
- [9] Tsutomu Wakimoto, et al, "A Low-Power Wide-Band Amplifier Using a new Parasitic Capacitance Technique," , IEEE International Symposium on Circuits and Systems Proceedings, pp. 106-109, 1989.
- [10] Rolf Schaumann, "The Problem of On-Chip Automatic Tuning In Continuous-Time Integrated Filters," IEEE International Symposium on Circuits and Systems Proceedings, pp. 106-109, 1989.
- [11] Rolf Schaumann, *Design of Analog Filters*, Prentice Hall, 1990, pp.419-446.

5.4.8

- [12] Todd L. Brooks, et al, "Simultaneous Tuning and Signal Processing in Integrated Continuous-Time Filters: The Correlated Tuning Loop," IEEE International Symposium on Circuits and Systems Proceedings, pp. 651-654, 1989.
- [13] Haideh Khorramabadi, et al, "High-Frequency CMOS Continuous-Time Filters," IEEE J. of Solid State Circuits, Vol. SC-19, No. 6, pp. 939-948, Dec 1984.

NASA Institute Workshop for Secondary Physics Teachers

L. Volkening
Moscow High School
Moscow, Idaho 83843

The NASA Institute Workshop for Secondary Physics Teachers at the University of Idaho's NASA Space Engineering Research Center (SERC) was open by application. Preference was given to secondary physics teachers where it appeared the information presented in the workshop would most directly be utilized in the classroom. All levels of teaching experience were considered and no prior knowledge of computer chip technology was necessary to apply.

The NASA Institute Workshop offered courses of instruction on logic systems technology, the chemistry of building and operating microchips, constructing and designing analog to digital projects, designing and manufacturing of microchip circuits, utilization of Boolean logic, recognizing gate functions, and developing processes for trouble shooting problems in these systems. Labs and problem solving sessions were used extensively to develop skills the teachers need to teach students theory, logic, design, and manufacturing methods of microchip driven devices. In addition, teachers were instructed in the use of the CAD - PIGLET program for the development and design of microchip hardware. Each course was organized around the theme of making the instructor feel confident enough to design lessons in the classroom and lab or units of study to become a regular part of the curriculum.

1 The Work

Teachers were given enough information so, by the week's end, they could begin to see how an idea could be worked into a microchip circuit and how to build a single function system or sensing device utilizing microchip logic. Several small projects were built by the teachers using logic gate, sensing, and decision functions. One of the first project goals was to design and develop a microchip device that could act as a photogate, keep time, and calculate the velocity or acceleration of a moving object. Once this level of understanding was achieved, the instructors were then asked to explore the possibilities of developing further ideas into single function microchip systems of their particular area of expertise. Example: Design a microchip system that could sense and make decisions based upon the level, color, and intensity of light passing through a plant leaf along with doing some simple calculations regarding the rate of light intensity changes. This theme of designing a system to analyze and solve a problem was carried throughout the workshop. It was done in this manner to develop and foster confidence in each teacher in order to encourage the transfer of skills and knowledge into secondary classroom activities.

2 Workshop Specifics

Each teacher was given a text of "Fundamental Logic Design," two sets of instructor's notes from each class, a projects kit, several microchips of various gate functions, and as many 'take-home' goodies the institute could gather together such as a silicon wafer chip, printed layout, hat, etc Each teacher was also given the opportunity to request extra kit materials to act as 'seeds' to start student projects in the classroom. The workshop started at 8:00 am and continued through 5:00 pm for five days with the afternoon of the last day open to free investigative work. Each evening at 7:00 pm a guest speaker presented topics of concern followed with idea sharing sessions. These lasted until 10:00 pm or later.

3 Background

The workshop was an idea resulting from NASA SERC Director Dr. Gary Maki's interests in public education and concern that some modern technologies are not being presented in the typical physics and math classroom. Math and science curriculums are rarely updated to current technology standards and fear of not knowing enough background and/or of not having hands on training has kept many instructors from presenting new and/or modern information. For these reasons, the workshop had to be done in a friendly and non-threatening manner.

When contacted by Dr. Maki to help design and set-up the workshop it was obvious where the work had to begin. A few energetic physics teachers have been designing projects specifically along the lines of solving some of Dr. Maki's concerns. Few, however, have achieved much success. This is due mainly to the lack of experience and lack of workshops offering technology courses at this level. In addition, discoveries in physics, technology advances, and curriculum design rarely coincide which further complicates updating course content. Other obstacles such as text, equipment, curriculum standards, time to learn and time to implement changes, almost make the concept of updating course content to fit the needs of students a nearly impossible task for the teacher. If a workshop was to be given it would have to be done in a non-threatening manner to increase the probability of having teachers feel successful enough to want to take the challenge of designing and implementing new skills and technology into the classroom.

Currently, teachers using breadboards, logic gates, or computer chip technology in the classroom are labeled as 'experts' by other teachers. Yet, in many cases, these teachers are working on their own. With a little encouragement they will most likely develop better and more updated curriculums. Today's students are eager to learn modern technologies and they welcome modern course content. The secondary goal of the workshop was then to develop a feeling of confidence within the teachers to enable them to try new ideas in the classroom. As a result, a theme of keeping the learning enjoyable, yet not lacking in substance, had to be developed.

The conditions under which many physics teachers work are extremely varied yet there are some similarities that could be used in developing the NASA workshop. In most cases,

funding for the secondary physics classroom is low and in many cases nonexistent in the area of capital outlay. The workshop chose to use breadboards since breadboarding is cheap, consumable and supplies are easily acquired, without putting a major dent in the school's budget (or in the student's pocket if the student try some independent work).

Student interest in taking physics sometimes remains low due to the 'antique' equipment and methods employed in the classroom. Yet, as soon as an instructor shows a device that appears as something special or computer-like the interest level in the course rises since this is the area in which they have been told over the years the next generation will be working. Robotics, logic, computer technology, math pattern recognition, etc. are all buzz words students love to use. The idea that a physics student could tell mom and dad how the laser reader at the local grocery works or that she could help her brother build a science fair project using a computer chip is astounding both for the student and the family. This gives the physics teacher a high level of credibility and also helps generate more student interest in taking the class. The workshop chose to develop teacher understanding of topics ranging from microchip manufacture and chemistry to decision making gate logic operations. This was done to encourage teacher involvement with students at the level of understanding needed to capture proper usage of the buzz words.

4 Workshop Results

Teachers were asked to complete questionnaires (Appendix A) about every aspect of the workshop, with the guide that their comments and criticisms would be used to help develop the next workshop. In no case was a teacher negative about the workshop, course instructors, or overall methods of presentation of the material. Several teachers commented along the lines of saying it was nice for a change to attend a workshop of some real updated value and the hands on skills acquired were excellent. Several requested that the workshop be presented in a more detailed fashion, yet others felt it was just right. Almost all the comments indicated that the teachers will most definitely develop some curriculum changes in their classrooms and that they felt much more confident in discussing microchip technology. Several teachers expressed the desire to return for more information at a more technical level and others expressed the desire to return for more practice in developing breadboarding skills. Some of the changes that seemed to be of concern were mostly directed towards administrative concerns or evening free time. In no case was a teacher negative about any of the conditions in which they did their work. And in no case did a teacher express a negative comment regarding time spent attending the workshop. Overall, the workshop could be described as a major success, especially when several teachers expressed sorrow in having to leave.

4.1 One outcome – my own classroom

This year, I have chosen to start the course in a non-traditional manner. Normally, physics is taught starting with Newtonian mechanics. I have always been concerned over the concept of inundating students with equations, terms, units, and math that is often just

5.5.4

plain overwhelming, especially to the average non-excelling student. This year I chose to start with the modern physics concepts of field theory, astronomy, and Einsteinian mechanics in a very light, non-threatening discussion mode. During this time I emphasized the need to know basic mechanics since this is what most of us see in our regular lives. Eventually, I asked about the need to have tools that can help make the small precise observations needed to understand basic and modern mechanics. As a result, the course shifts from one of transferring interesting knowledge to one of "how to build devices that can help us become good observers." I have then set the stage for students to become scientists who are ready to learn the electronics and logic needed to develop precise timing devices, light sensing devices, decision making logic circuits, etc

With this shift the course can now use basic electronic equations and simple math and algebra skills. The course constantly utilizes electronic energy equations to show that energy is the same whether electrical or mechanical. Energy analysis just needs different equations to show various relationships. The reason for learning electronics is based on the student's desire to build a timing device and computation device in order to look closely at basic mechanical events. In addition, I have suggested that maybe some of the electronic skills used in looking at mechanics could be modified later when it becomes necessary to look at electromagnetic events closely. This I hope will set the stage for developing the desire to look closely at modern physics phenomena.

4.2 One Example

At this time we are using Ohm's law and power equations to help design simple breadboard circuits using microchips and transistor gates to sense events and drive a logic circuit that can make decisions. In order to get to this level, we have had to discuss such things as the purposes of Boolean logic, binary numbers, asserted versus non-asserted operation modes and interfacing techniques with the computer. Trouble shooting techniques are a major part of the lab work, yet they are viewed as somewhat of a challenge since each student's problems are unique to the circuit board being built. We just finished a sailcart race where carts were designed to run by the power of a window fan and timed by hand to go a specified distance. The circuit board technology to build a timing device became a very desired tool to have in helping time carts down the course. The question of building a single circuit chip device that could detect time rates and calculate acceleration is a result of student interests, not instructor's desires. The class is now ready to build devices to help detect Newtonian events and to develop logic systems that will analyze the events.

Appendix A

Comments on the logic lectures:

- Burton, Fred: Well prepared. Carefully presented. A little fast for some. Good relations with class. Enjoyable.
- Aiken, David: Good but got too deep too quickly for me.
- Taylor, John Lynn: Suggest future candidates study associated materials and send materials to participants for review and study.
- Steidley, Perry: I could follow Dr. Maki when he was discussing the topic but for me to duplicate it would be tough. Dr. Maki is very thorough.
- Foster, Douglas: Great instruction and good introductions. Good use of selected problems and solutions.
- Wagner, Charles: Another half hour of introduction would have been good.
- Gilmore, Tom: Cut back 30% and simplify. Keep the Boolean algebra, diagrams and truth tables. Topics are good, but there is too much information.
- Volkening, Larry: Need one on recap with problems, solving by singles or as groups. Excellent basic level.
- Bergstrom, Rich: Amazed that you could take us so far so quickly. Well paced, only a few times did you get over our heads.
- Gantz, Mike: Expand the lectures to four days - same amount as materials.
- Kemp, William: Excellent.
- Parker, Donald: Very good. Basic enough to use ideas in classroom and in grade school.
- Kurowski, Ed: Challenging and within the range of comprehension of the participants.
- DesArmo, Francis: Good. Very applicable in our high school geometry in our chapter on logic.

5.5.6

- Martell, Charles: Lecturer was very knowledgeable and organized but at times seemed to go too deeply into the material.
- Barton, Evan: I learned some concepts which will be useful in my teaching. I was impressed by Dr. Maki's concern for us.
- Weese, Jim: I liked Gary Maki's enthusiasm and excitement; but the first day was too fast. The second and third days were good.
- Hellman, Walter: An incredible amount of stuff was covered very well. It is hard to believe we could be taken from level zero to real application so quickly. Gary has a great feel for how to select what to teach, as well as what pace to teach at.
- Ovall, Larry: Good overview of logic.
- Pedlar, Craig: Presented very well, but I would have liked more "absorption" time, discussion time, and lab work time.

Comments on the electronics lectures:

- Burton, Fred: Good content. Basic transistor theory might have been helpful. Chip manufacturing process very helpful.
- Aiken, David: Good but got too deep too quickly for me.
- Taylor, John Lynn: Suggest future candidates study associated materials and send to participants for review and study. The given materials were very good and much appreciated.
- Steidley, Perry: Extremely well done!
- Foster, Douglas: Great state of the art formal presentations.
- Wagner, Charles: O. K.
- Gilmore, Tom: Good material on Hubble Space Telescope - hot topic. Cut back 20% Includes more practical examples: chips in toys, cars, appliances, etc. More real world material like telephones and communication satellites would be good.
- Volkening, Larry: Excellent basic level especially on manufacturing process.

- Bergstrom, Rich: Good ideas, the second day it got a little fast and over most of our heads. The other two days were outstanding.
- Gantz, Mike: Expand the lectures to four days – the same amount of time as materials.
- Kemp, William: A bit hard to follow at times but interesting and useful.
- Parker, Donald: Very good. I gained a basic understanding and will be able to use the concepts in my classroom.
- Kurowski, Ed: Challenging and within our range of comprehension.
- DesArmo, Francis: Very informative, although much of it was difficult for me to follow.
- Martell, Charles: Very good presentation but once again much material was too deep; it would have been better to aim a little lower but develop a better understanding.
- Barton, Evan: Appreciated being able to talk with a scientist involved in NASA programs.
- Weese, Jim: Day one was excellent. Days two and three I did not understand. Transistor graphics and overall concepts came through.
- Hellman, Walter: First lecture on transistor was the best explanation on the subject that I have seen or heard. Subsequent lecture covered too much too fast. The final lecture on fabrication was good.
- Ovall, Larry: Good presentation of circuitry and design.
- Pedlar, Craig: Presented very well. Personally, I would have been helped by more "absorption" time, discussion time, and lab work time.

Comments on the logic labs:

- Burton, Fred: Diagrams for our labs would have been helpful. Not enough time to digest theory, technical and practical.
- Aiken, David: Good hands on experience.

5.5.8

- Taylor, John Lynn: Great.
- Steidley, Perry: I was scared to death at the onset, but by the weeks end I had relaxed and learned.
- Foster, Douglas: Good emphasis on topics that were covered in class.
- Wagner, Charles; O. K.
- Gilmore, Tom: About right.
- Volkening, Larry: Need one final complete circuit diagram added after two or three days (sensor/timer/logic to yield an entire project).
- Bergstrom, Rich: First day put the whole day together: outstanding! I got over my fear of electronic chips.
- Gantz, Mike: I got lost at times because of a sometimes hurried pace, but overall it was very educational.
- Kemp, William: Early instructions were weak - I was unsure of where we were going.
- Parker, Donald: Excellent! I can use the information and procedures in my class.
- Kurowski, Ed: Working in teams proved faster and more rewarding (2 per team) under the time constraints. If more time were available then maybe individual work would be best.
- DesArmo, Francis: Great time. I did not think I had learned enough to apply it in lab, but I was wrong.
- Martell, Charles: Well run lab which intertwined well with the lecture materials.
- Barton, Evan: Good hands on experience. Projects were good! Could give a little more time to work on them.
- Weese, Jim: The first two projects were easy to build; the third project was not so successful. There was good student help.
- Hellman, Walter: Great hands on experience which was challenging but possible to do.

Ovall, Larry: Excellent applications of the logic circuitry and the electronics equipment.

Pedlar, Craig: Presented very well. Personally, I would have been helped by more "absorption" time, discussion time, and lab work time.

Comments on the piglet lab:

Burton, Fred: Amy Anderson (NASA scholar serving as Piglet instructor) was very helpful and related well with people older than herself: good skills.

Aiken, David: Very interesting.

Taylor, John Lynn: Great.

Steidley, Perry: This I dug! Also, Amy was wonderful, helpful and patient.

Foster, Douglas: Lots of fun – interesting to learn technologies of how everything is created and manufactured.

Wagner, Charles: Back off to smaller circuits that could be done in high school. Just lab (#2) was just great – the rest were o.k. but usable.

Gilmore, Tom: Cut back 1/3. Use one day to study the components in a personal computer. Troubleshooting.

Volkening, Larry: Helpers were excellent, knowledgeable and had the ability to relate.

Bergstrom, Rich: Good exercise. It was interesting to see how logic designers really work.

Gantz, Mike: Very well set up. Keep the one helper per two participants ratio. A listing of menu commands and what their function is (in layman's terms) would be nice.

Kemp, William: Needed more time but it was very enlightening.

Parker, Donald: Excellent. It was my first chance to use this type of computer. It was an excellent experience which we can bring back and use in the classrooms.

- Kurowski, Ed: Working in teams proved faster and more rewarding due to the time constraints. If more time had been available then individual work would have been best.
- DesArmo, Francis: Too many new commands in such a short time, but with the assistance received things went fine.
- Martell, Charles: Nice equipment to use, but perhaps a slower pace would be good.
- Barton, Evan: I enjoyed the hands on experience with computers and the insight into chip design.
- Weese, Jim: Amy is the greatest. I wish her luck in her current studies.
- Hellman, Walter: Good experience to see how this industrial and commercial application works.
- Ovall, Larry: Nice to see how the CAD program works since I had not used one before.
- Pedlar, Craig: Presented very well. Personally, I would have been helped by more "absorption" time, discussion time, and lab work time.

Comments on the projects lab:

- Burton, Fred: Good ideas that I can use immediately in class. A strong finale!
- Aiken, David: Most directly applicable to high school teaching.
- Taylor, John Lynn: Very very good - I loved the take home materials.
- Steidley, Perry: The circuits and projects were wonderful.
- Foster, Douglas: I would have added some direct interfacing designs. I will be glad to come back and help out. Probably having a IIE or IBM and showing how to actually interface them would be great help.
- Wagner, Charles: O. K.

- Gilmore, Tom: Focus on four specific devices. Make sure that each device gets tested and works.
- Volkening, Larry: Need one final complete circuit diagram added after. two or three days (sensor/timer/logic) to yield and entire project.
- Bergstrom, Rich: Very good. But need to have better equipment so it would work more often. It became frustrating when some of the projects did not work. Gained tremendous ideas to give to kids for hands on experience.
- Gantz, Mike: Excellent. Larry was superb. Materials are greatly appreciated. Very understandable.
- Kemp, William: Excellent - need more equipment so you could save something once it worked.
- Parker, Donald: Excellent. Will use for sure.
- Kurowski, Ed: Working in teams proved faster and more rewarding under the time constraints. If more time had been available then maybe individual work would have been best.
- DesArmo, Francis: Had fun. Actually had success. Nice loose atmosphere in all classes made it enjoyable.
- Martell, Charles: Great stuff. The most successful of all materials.
- Barton, Evan: Would help to have diagrams to do projects.
- Weese, Jim: I will use more of this in class than all the other presentations. A written description of each lab with expected results and possible extensions would really be appreciated - especially two months from now when memory and notes fail.
- Hellman, Walter: Not clear yet - still assembling.
- Ovall, Larry: Again, excellent use of the electronic devices talked about during the week.
- Pedlar, Craig: Presented very well. Personally, I would have been helped by more "absorption" time, discussion time, and lab work time.

Did you gain the skills you expected?

All participants answered "yes."

Suggestions/comments regarding whether or not expectations were met:

- Burton, Fred: None.
- Aiken, David: I really did not know what to expect - but I am not disappointed.
- Taylor, John Lynn: None.
- Steidley, Perry: I saw my limitations and liabilities quickly!
- Foster, Douglas: None.
- Wagner, Charles: A few more basic electronic logic setups.
- Gilmore, Tom: Guest speakers from Apple, IBM, or NASA. Trips to chip manufacturers or a video on actual chip production. Film material on NASA projects that can be used with our classes.
- Volkening, Larry: We need a group meeting to finalize each of our parts and to help each plan if to continue next year.
I gained more insight in how design engineers work and how chips are made.
- Gantz, Mike: the CAD, Logics lab and Projects lab were extremely informative.
- Kemp, William: I am not sure what I expected but it was extremely worthwhile.
- Parker, Donald: My cup runneth over.
- Kurowski, Ed: More than I expected in some cases.
- DesArmo, Francis: You are doing a great job at promoting the sciences. Your enthusiasm for your work is truly invigorating. I wish all universities put forth the effort I have witnessed at the University of Idaho.

- Martell, Charles:** Focus more on certain concepts or skills and allow more time to master these skills. Much went over my head, but I gained much information which will help me.
- Weese, Jim:** I expected more basic component use and testing using testing devices like the oscilloscope and the test probes on the bread board.
- Hellman, Walter:** None.
- Ovall, Larry:** I feel more comfortable with microelectronic technology now. Not so afraid to tackle this area alone now.
- Pedlar, Craig:** I would have liked to have had a two week session – with more time to absorb material presented, work on projects (to actually make them work) and really comprehend their operation and fundamentals regarding that operation: some evenings to discuss this with instructors and other participants. Some free time to use other campus facilities would be nice.

Are you going to be able to use these skills in your classroom ?
All participants answered "yes".

Suggestions/comments regarding skills learned:

- Burton, Fred:** I had not had any training with IC's.
- Aiken, David:** None.
- Taylor, John Lynn:** Most definitely just what I wanted and needed.
- Steidley, Perry:** None.
- Foster, Douglas:** None.
- Wagner, Charles:** I have found two truth table labs that I can put to good use! These labs alone made the institute worthwhile but there was much more and very good.
- Gilmore, Tom:** None.

- Volkening, Larry: I would like a classroom set of equipment as listed in hand-out.
- Bergstrom, Rich: I will use a lot of the information given here to pass on to students. But to actually create some of the circuits I really do not know.
- Gantz, Mike: I would suggest that the lectures be toned down to a high school level. More information based on electronics.
- Kemp, William: None.
- Parker, Donald: Maybe not the specifics but the generalities.
- Kurowski, Ed: PC boards - especially if I can get 25 or so.
- DesArmo, Francis: None.
- Martell, Charles: Projects class has some good activities and I will do some digital work with my AP kids.
- Barton, Evan: Students have very little knowledge in the area of microelectronics. This class will help me to help eliminate this problem.
- Weese, Jim: Breadboarding, truth tables, value of Boolean algebra all helpful.
- Hellman, Walter: None.
- Ovall, Larry: It will take me some time to work it in but I see plenty of places for it.
- Pedlar, Craig: None.

Misc. Suggestions:

- Burton, Fred: Consider a slower pace, more background, more basics, more practice, and extend to two weeks.
- Aiken, David: None.
- Taylor, John Lynn: None.
- Steidley, Perry: None.

- Foster, Douglas: I would have liked to had some time to see what research projects are being staffed in various departments – especially the physics department. I would be glad to teach a course on digital interfacing for Apple 2E computers/MAC computers. Hands on building kits and trying them would be good.
- Wagner, Charles: None.
- Gilmore, Tom: Try to incorporate local students into an Electronic Lab Day. Let the teachers work as T. A.'s one afternoon with students. Let us see if these electronic devices actually work with junior high students.
- Volkening, Larry: 1) Need more time to stream-line and allow free time. 2) Need to give each participant a formal chance to speak to others, if so desired (idea sharing/problem solving session).
- Bergstrom, Rich: I would not change much about the workshop. It was fast pace and fun. For a week long workshop this was very well done and extremely well organized.
- Gantz, Mike: Five full days instead of four and a half. I do not think the text was needed: apply the money used to some other fund.
- Kemp, William: Thursday we needed longer lab periods once we got going.
- Parker, Donald: Continue to give this type of workshop, possibly two times per year. Give a two day workshop for teachers and students during the year.
- Kurowski, Ed: Keep up the great work and try to involve some of us again next year or in a three year master's program.
- DesArmo, Francis: I would have liked just a few more minutes when attempting problems.
- Martell, Charles: Your ideas for on going activities are great. We need to encourage more summer workshops and summer training for science teachers. Keep up the good work!!!
- Barton, Evan: It would be helpful to have more information on our level. A little more time on projects. It would be nice to have time in the evenings to become somewhat familiar with the surroundings.

- Weese, Jim: More applications and uses of electronics – research edge! Money for equipment for classroom use. A slower pace. Begin projects earlier in the week (more time to do lab extensions). Additional speakers could be brought in from industry. “War stories” of design and manufacturing problems are of great interest to students and are great fill in and attention getters for teachers.
- Hellman, Walter: Maybe things will change but logic text was not use much here and I will use it much in the future. A general purpose electronics reference would be more useful.
- Ovall, Larry: It would be possible to be much stronger and better able to implement this technology if this institute were for a longer time.
- Pedlar, Craig: Possible consider a two or three stage program providing 1) a week as we had (maybe less rigorous) 2) following year – a two week session with time for comparing what was done during the year 3) the third year the “work with a scientist” month.

Additional Comments:

- Burton, Fred: It's difficult to critique something that is done so well. Excellent choice of staff! Each related well with us. Staff and participants both wanted to learn and share expertise. All were committed to our task. Well planned and executed. Staff members complemented and supported each other's work.
- Aiken, David: None.
- Taylor, John Lynn: None.
- Steidley, Perry: This workshop opened my eyes to an area of physics that I was not current on and have not used in my curriculum! I will be more aware of these areas in my future teaching.
- Foster, Douglas: I will send you a copy of PSSC Physics Chapter and labs in logic circuits. I suggest you send this and other basic material to all participants and in the future to all participants ahead of the workshop. It would have been nice to have some time to share with other physics teachers some ideas about what we are doing. Plan and announce ahead of time so that everyone can bring some ideas!

- Wagner, Charles: Instructors and helpers were really very good. Introduction - update - future. The way material has to be presented in short session. Very pleased by individuals running course.
- Gilmore, Tom: Good use of graduate students in this program! Judy, you did an excellent job - many thanks!
- Volkening, Larry: Judy and Kelli did an exceptional job of organizing, etc. the entire workshop. I can't say enough good words. Let's consider designing and making parts available for teachers and students to build a complete single function computer from sensing through computation to read-out. Make circuit diagram and piglet print-out part of the package. Example: display T; calculate a; display a from $s = 1/2 \text{ at}(sq)$; as the result of four photogates being trapped.
- Bergstrom, Rich: I think Judy and her staff made the workshop a big success. My hat is off to them. This has been a great energizing time for me. Please run it a lot more times.
- Gantz, Mike: I would like to see more release time in the evening. No one really had a chance to see the area. The instructors, helper, etc., were the best I have ever had. This program is super! I have tried to introduce electronic circuits but have never gotten beyond resistors and capacitors. Now I have a knowledge that I will use a great deal in my classes.
- Kemp, William: The attitudes and interest by all the instructors, especially Gary, was tremendous. You made us feel wanted, needed and treated us as professionals. You listened and helped - thanks for the experience.
- Parker, Donald: Set up field trips for students and teachers, rural area students are not aware of the outside world. Industry as well as college campus life similar to Agriculture and Science Days at Oregon State, Science and Humanities or Science Experience Days. Each time I attend a workshop like Sir Issac Newton I stand on the shoulders of giants and see further than before. This time is no exception; however, the giants were much larger.
- Kurowski, Ed: None.

- DesArmo, Francis: Evenings free. The talks in the evening were very good and should not be dropped. I don't know when they would be best.
- Martell, Charles: A good week but maybe a little too much information overload. Material would be better understood over two weeks with more lab type activities.
- Barton, Evan: None.
- Weese, Jim: A second week of basic electronics using and testing basic electronic components with specific emphasis on the oscilloscope. My electricity (math oriented) was college taught - my electronics and meter use is mostly self taught. Many physics minors or less have had very little electricity.
- Hellman, Walter: I might "free" to see a little bit of the area. This is a great experience. I hope others will have the opportunity to partake of it! I wouldn't change in any major way at all - you've hit on something really good the first time.
- Ovall, Larry: This institute allowed a wide variety of physics teachers to come together for a week and share their experiences in the classroom and beyond. That is an invaluable additional result of getting together.
- Pedlar, Craig: None.

Session 6
Neural Networks and VLSI
Telemetry

Chairman: Gary Maki

Neural Net Diagnostics for VLSI Test

T. Lin, H. Tseng, A. Wu, N. Dogan and J. Meador
Department of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752

Abstract-

This paper discusses the application of neural network pattern analysis algorithms to the IC fault diagnosis problem. A fault diagnostic is a decision rule combining what is known about an ideal circuit test response with information about how it is distorted by fabrication variations and measurement noise. The rule is used to detect fault existence in fabricated circuits using real test equipment. Traditional statistical techniques may be used to achieve this goal, but they can employ unrealistic a priori assumptions about measurement data. Our approach to this problem employs an adaptive pattern analysis technique based on feedforward neural networks. During training, a feedforward network automatically captures unknown sample distributions. This is important because distributions arising from the nonlinear effects of process variation can be more complex than is typically assumed. A feedforward network is also able to extract measurement features which contribute significantly to making a correct decision. Traditional feature extraction techniques employ matrix manipulations which can be particularly costly for large measurement vectors. In this paper we discuss a software system which we are developing that uses this approach. We also provide a simple example illustrating the use of the technique for fault detection in an operational amplifier.

1 Introduction

An integrated circuit test is a combination of input and output signals which characterize some attribute of idealized circuit function. The presence of faults in a fabricated circuit will cause observed output signals to deviate from the simulated ideal. Unfortunately, variation of fabrication process and device parameters as well as measurement noise will also cause a deviation from ideal circuit performance, so something is needed which helps distinguish signal deviations due to fault existence from those due to these other sources. A diagnostic is a decision rule combining what is known about an ideal circuit test with information about how it is distorted by fabrication variations and measurement noise. The rule is used to detect fault existence in fabricated circuits using real test equipment. In this paper we discuss the application of neural network algorithms to the automatic synthesis of diagnostics for integrated circuit test.

Diagnostic synthesis is less concerned with specific aspects of test design than it is with the generation of a decision rule for a given test and process specifications. The focus of most test generation techniques is upon finding an appropriate combination of signals which will properly excite a circuit to reveal the existence of a potential fault. In diagnostic synthesis, a specific test has already been designed (typically under the assumption of a deterministic measurement process), and the job is to find some decision function which accurately reflects the outcome of that test in the real world (where measurements are random). In the event that a good diagnostic cannot be found for a given test, that information can provide feedback to the test designer so that a more robust variation can be created.

2 Statistical IC Diagnostic Synthesis

Diagnostic synthesis can be formulated as a statistical pattern recognition problem. This involves the generation of sample data and the analysis of that data using statistical tools. A priori assumptions about the measurement distribution can be made to simplify the mechanics of the data analysis, or a large number of samples can be used to approximate actual distributions. Feature extraction and pattern clustering techniques can also be used to simplify the discrimination task.

One way to obtain sample data for IC test measurements is simply to fabricate ICs. This is clearly the most accurate way to characterize process dependent performance variations, but it is also an expensive alternative. Monte Carlo simulation of process and device parameter variation is more economical provided circuit simulation requirements do not exceed the capacity of available computational resources. This approach will be less accurate since process and device model limitations may not fully reflect actual circuit performance.

Given a sample of noisy test data, one approach is to assume some measurement distribution, then use sample moments to estimate the joint probability density functions (jpdf) of operational and faulted test results. A threshold discriminant can then be employed as a diagnostic. This method suffers from the inaccuracy of the distribution assumption as well as from the need for some separate feature extraction technique to decide which measurements are useful. Even if fabrication process disturbances are normally distributed, the nonlinear relationship between process variation and measurement means that the measurement data cannot be expected to be distributed in some easily predictable manner. The strongest a priori assumption that can be justifiably made about measurement distribution due to process variation is that it is probably unimodal and asymmetric [11]. Monte Carlo generation of a large number of samples better approximates measurement jpdfs [1], but still suffers from the need for separate measurement feature extraction.

Feature extraction involves the selection of measurement combinations which provide for a more efficient representation of the raw data. A more efficient representation emphasizes combinations which exhibit better discrimination properties. Feature extraction implies data preprocessing which reduces measurement dimensionality as well as clustering

similar measurements. Previously used algebraic methods [11] require the manipulation of large matrices when there are many measurements and provide no solution guarantee. Sometimes human inspection of scatter plots is used to discover correlations between sample data measurements [1] and reduce the number of required measurements.

3 Statistical Properties of Feedforward Neural Networks

A multi-layer feedforward network of the kind currently popular in the neural network literature can be viewed as a statistical pattern recognition algorithm. When trained on random sample data, neural network connection weights effectively form a vector-valued statistic of that data [12]. Feedforward neural networks also serve as universal vector function approximators provided sufficiently many hidden units are available [2] [4]. These properties suggest that a feedforward network can be used to approximate a discriminant function based upon random sample data without the need for a priori knowledge of the sample distribution or an excessive number of samples.

The popular backpropagation training method [9] arrives at feedforward connection weights in a fashion which encourages the automatic extraction of important data features. The gradient descent algorithm associated with backpropagation strengthens connections which contribute to the reduction of error in the approximation of the sample mapping. A feedforward network trained this way will emphasize input combinations which contribute the most to a good approximation, automatically performing feature extraction without the need for data preprocessing. Clustering of similar inputs and dimensionality reduction can both be observed to occur automatically. These characteristics help eliminate the need for cumbersome scatter plot inspection and numerically unwieldy algebraic data preprocessing.

4 Neural Network Based IC Diagnostic Synthesis

The IC diagnostics which we are investigating take advantage of the properties associated with feedforward neural networks. Monte Carlo simulation is used to generate a sample data set modeling ideal test conditions in the presence of process and measurement noise. Part of this sample data set is used to train a feedforward neural network using the backpropagation algorithm. The resulting connection weights define a discriminant function which is then tested for fault coverage performance using the remaining portion of the sample data set. Once an acceptable level of coverage is determined, the connection weights are available for transfer to the automatic test equipment.

The main advantages of this approach are that no measurement distribution assumption is needed to form a discriminant and that features are automatically extracted without complex numerical manipulations. The principle disadvantage is that iterative gradient descent training techniques like backpropagation are subject to solution convergence difficulties which can lead to excessive training times and nonoptimal solutions. It is notable

that contemporary research toward finding ways to overcome such difficulties is in progress.

Recent work in both electronic circuit test and automobile diagnostics lends additional support to this approach. Neural networks have been demonstrated which approximate the relationship between a node voltage measurement space and a six resistor circuit element space, with the goal of detecting out-of-tolerance device parameters [10]. They have also been used to discriminate between automobile engine faults given control CPU signals [6]. Our work includes an additional dimension to these previous results by specifically incorporating the effects of production variations and measurement noise.

Figure 1 shows a diagnostic generation system in its intended context within an overall IC test design strategy. The diagnostic generator combines circuit layout and test specifications with process specifications to generate a diagnostic decision rule. The test specification is expressed as a fault dictionary which relates ideal stimuli and responses to various fault conditions for a given circuit specification. The diagnostic is expressed in terms of a specific feedforward neural network configuration and its associated connection weights. The diagnostic processor corresponds to the hardware which executes the neural network algorithm in conjunction with automatic test equipment. The diagnostic generator also provides some confidence measure which indicates fault coverage. This can help guide potential revisions of the test specification or even the tested circuit.

The internals of the diagnostic generator are shown in Figure 2. Process specifications are translated into a device characteristic sample via Monte Carlo process simulation. The layout specification is translated to a netlist by a circuit extractor, and both are used as input to a circuit simulator. The test stimulus completes the specification of a circuit simulation which when executed, provides a random sample of test results. This sample simulates the measurements which would be made on a batch of fabricated ICs. A measurement simulation then distorts the fabricated response sample, modeling the imperfections of the targeted test equipment. The resulting simulated response sample is then combined with additional information from the original test specification during the training of the pattern recognition algorithms. The results of this training process are then made available to the designer in the way of a diagnostic decision rule and feedback regarding its effectiveness.

A fabrication process simulator has been implemented using the SUPREM-III process simulator [3] and the PISCES-II device characteristic extractor [8] configured with a Monte Carlo process parameters generator. The FABRICS fabrication process simulator [7] is another tool available for implementing the statistical simulation of the fabrication process. In our experimental setup, we are using various Berkeley tools for circuit specification with PSPICE and MCNC CAZM as our circuit simulators. The specific choice of process and circuit simulators is relatively independent of our diagnostic synthesis goal, and is considered a matter of designer preference.

The measured response sample obtained from these simulation steps is then used as input to a feedforward neural network training algorithm. The sample is partitioned into two smaller pieces: one for training and one for evaluation. The network is trained on equal numbers of faulty and fault-free exemplars from one of these sets. The quality of the acquired discriminant is then tested using the previously unseen sample. If a sufficient

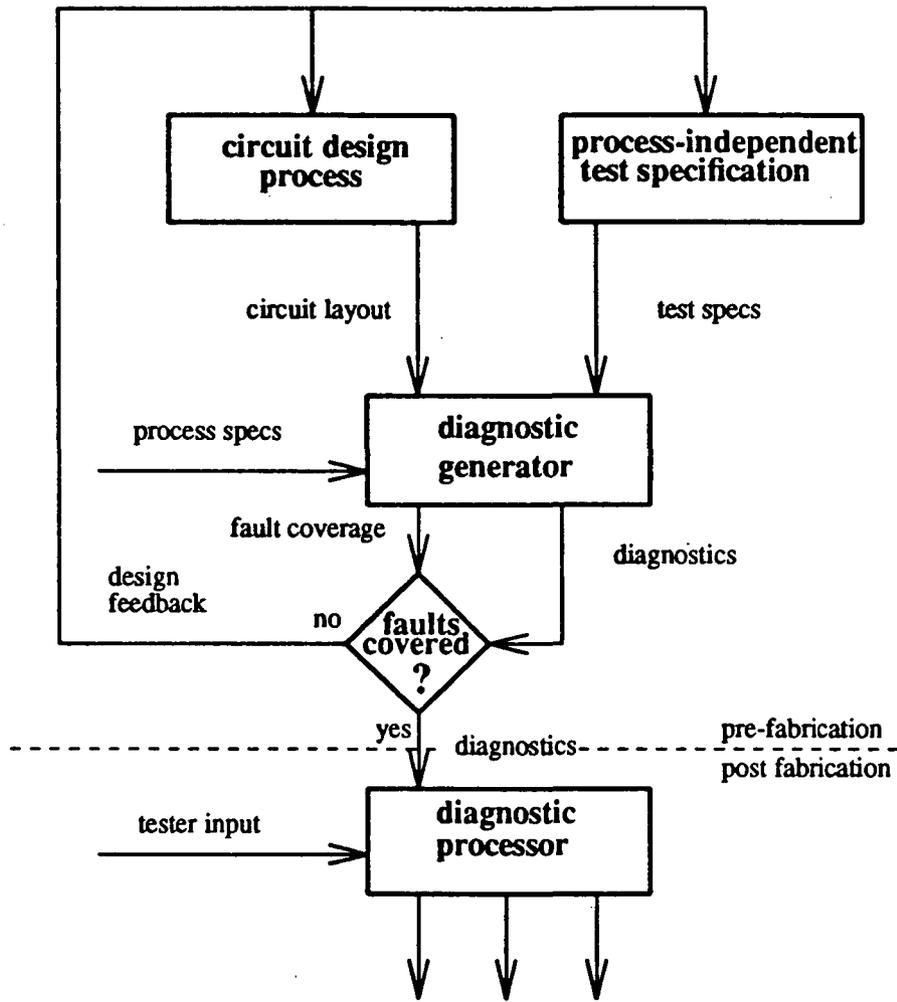


Figure 1: Diagnostic synthesis in a test design process

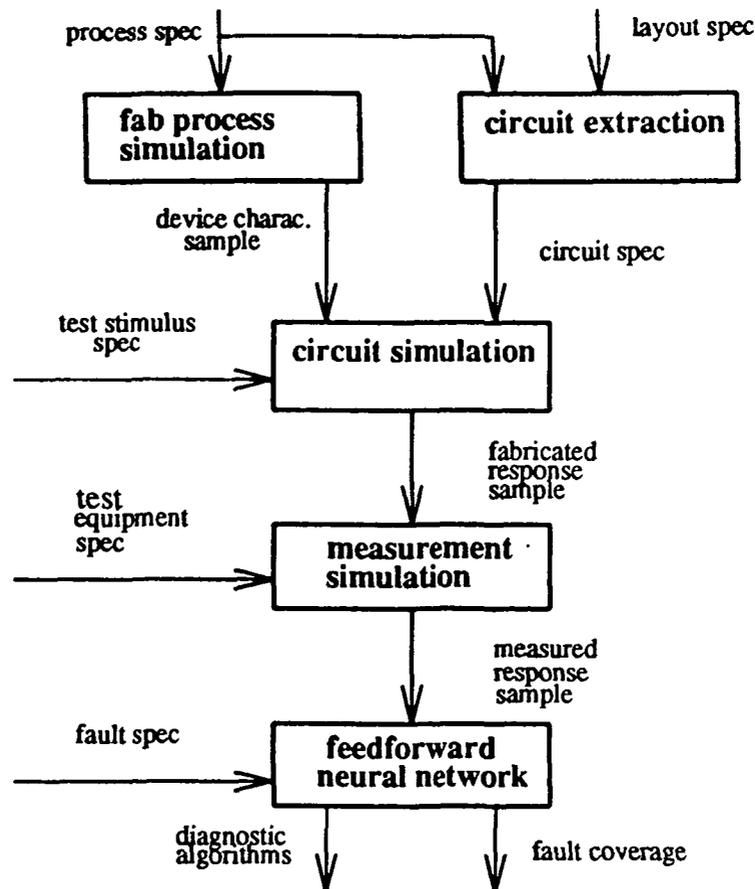


Figure 2: Neural network based diagnostic synthesis method

percentage of these test exemplars is properly classified, then the connection weights corresponding to the generated diagnostic are made available for transfer to the automatic test equipment. A poor discriminant can arise for many reasons however, ranging from an inappropriate test specification to an overly constrained network architecture. We are currently investigating variations of the backpropagation training algorithm which automatically add hidden units as a function of convergence rate. It is expected that a poor discriminant is less likely to result from an insufficient network architecture using such an algorithm.

5 Experimental Approach and the Results

The quality of the pattern classification performance strongly depends on the number of training samples the network is exposed to during the learning stage and how closely the training patterns resemble the actual data with which the network will be confronted during normal operation. Therefore, it is essential to find efficient techniques for fault simulation. In general, fault simulation can be carried out either in a real IC fabrication process, or using computer simulation. The first method has two severe drawbacks:

1. Such experiments are expensive and time-consuming.

- The disturbances introduced in the fabrication process cannot be controlled with sufficient accuracy.

We propose using a statistical process simulator SUPREM-III, a semiconductor device modeling program PISCES-II, and a circuit simulator, e.g., SPICE or CAZM for fault simulation. The relationship among SUPREM-III, PISCES-II, and SPICE or CAZM are as shown in Figure 3. SUPREM-III takes care of the process parameters, the layout parameters, and the process disturbances. The FAB process simulation outputs come from SUPREM-III are fed to PISCES-II for electrical characteristics analysis. The output of PISCES-II is then fed to SPICE or CAZM through an interface software called PICA. PICA is currently being designed at Washington State University. The circuit performances output from SPICE or CAZM will be used for pattern classifications using neural networks.

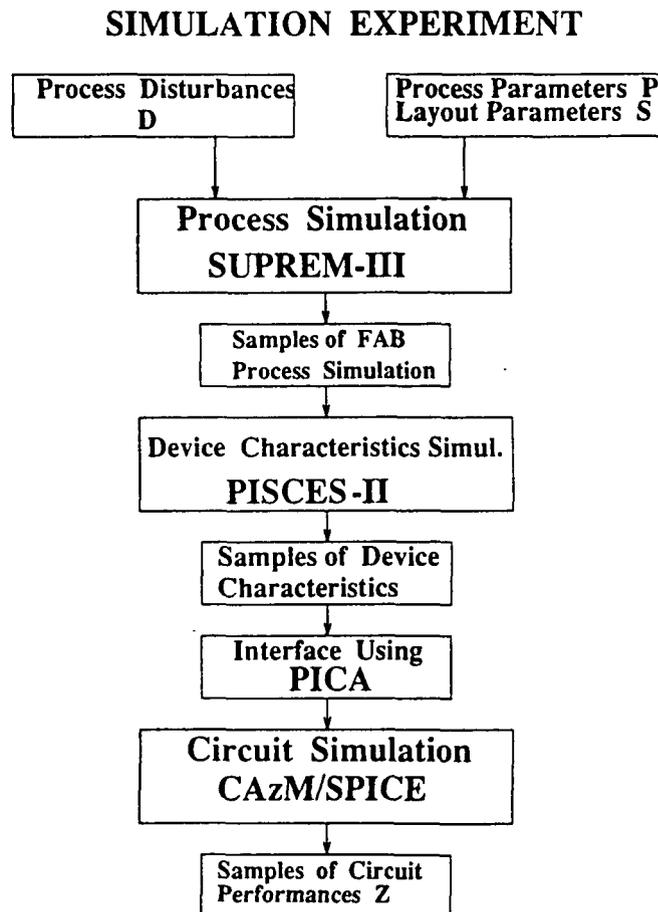


Figure 3: The simulation experiment flow chart for IC tests

The Monte Carlo Method [5] will be used to generate data which resemble the process disturbances. Monte Carlo method is one that involves deliberate use of random numbers in a calculation that has the structure of a stochastic process. By stochastic process we mean

a sequence of states whose evolution is determined by random events. In a computer, these are generated by random numbers. This particular experiment consists of the following steps: creating the fault dictionary using PSPICE, preprocessing the data of the fault dictionary, selecting the neural network architecture and training algorithm, and finally training the neural network for classifying IC failures.

An operational amplifier, shown in Figure 4 and 5, was used in our experiment. Two fault dictionaries of transient analysis were created by using PSPICE with Monte Carlo method. A normal data set was collected by varying device parameters within operational limits. While a faulty data set was collected by making the variation of the junction depth of one of the MOSFETs (M1 in Figure 5) exceeding the functional specification. Input stimulus is a 0.1 volt pulse of 5 micro seconds duration. Each data set has sixty patterns. Each pattern has 101 sampling points in 10 micro seconds. One of our experiments chose 32 samples from each pattern with equal spaces. The other one sparsely chose 11 samples and also evenly spaced. Data in the fault dictionary were normalized to be in the range between 0.1 and 0.9. Figure 6 shows these densely sampled signal waveforms in four sets a, b, c and d.

Using the backpropagation learning algorithm, a two-layer feedforward neural network (one hidden layer) was then trained as pattern classifiers for those data. For the data with 32 samples, we used a 32:5:1 network. For the one with 11 samples, we used 11:5:1 network. Data set used for training has 30 patterns each from both the normal data set and the faulty data set. After the network has been trained, all four data sets were used for testing. The 32:5:1 neural network can classify all 120 patterns after 15000 epochs of training. The 11:5:1 neural network, trained with sparsely sampled data, cannot finish the training phase after 73000 epochs.

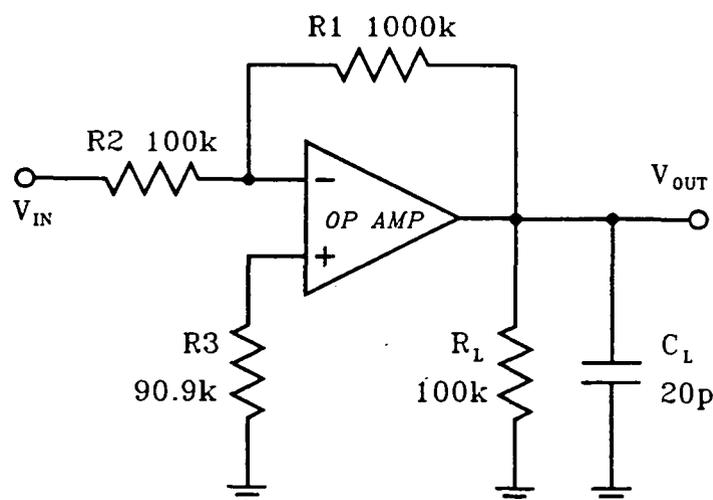


Figure 4: Configuration for SPICE simulation. The OP AMP is shown in Figure 5.

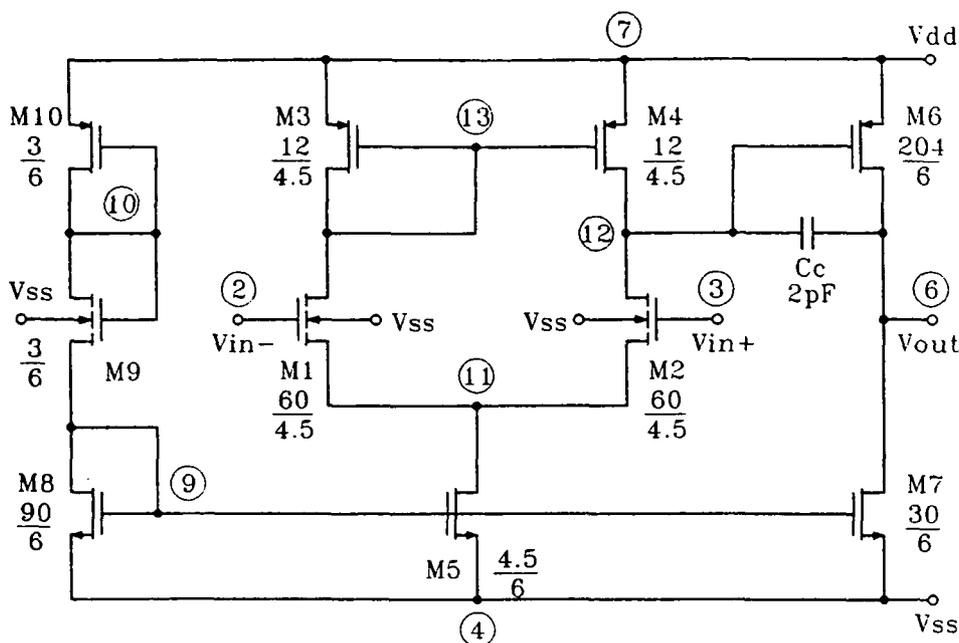


Figure 5: CMOS operational amplifier schematic diagram

6 Summary and Future Work

The possibility of using neural network in IC fault diagnosis problem is demonstrated. Results of the experiment positively showed the capability of the feedforward network in separating the faulty circuit from the normal circuit based on the patterns presented. However, its diagnosis ability depends on the information implicitly in the patterns used in training the network. In the case of sparse sampling, where output signal sampled eleven times, it is unable to train the network to diagnosis the fault from the pattern presented. It is because the essential features of the signals are not presented to the network. Important information is not included in the sparsely sampled pattern. When the output signals were sampled more densely, the network can identify the patterns accurately. Such phenomena is not hard to explain.

Examining the output signals carefully, the distinctive features are in the slew rate and the overshoot of the output signal. Faulty circuit has a slower slew rate and larger overshoot. These distinctive features are not included in the sparsely sampled pattern. Since the output signals of both faulty circuit and normal circuit are very similar in the shape of the waveform. It is hardly to distinguish the signals simply based on their shape. However, when densely sampled, these features present in the pattern used for training and verification. The trained network can positively identify the faulty signals.

Further work will be conducted to investigate the sampling dependent phenomena and to establish techniques to guarantee success in diagnosis using neural network; and also assist how diagnosis measurement should be conducted.

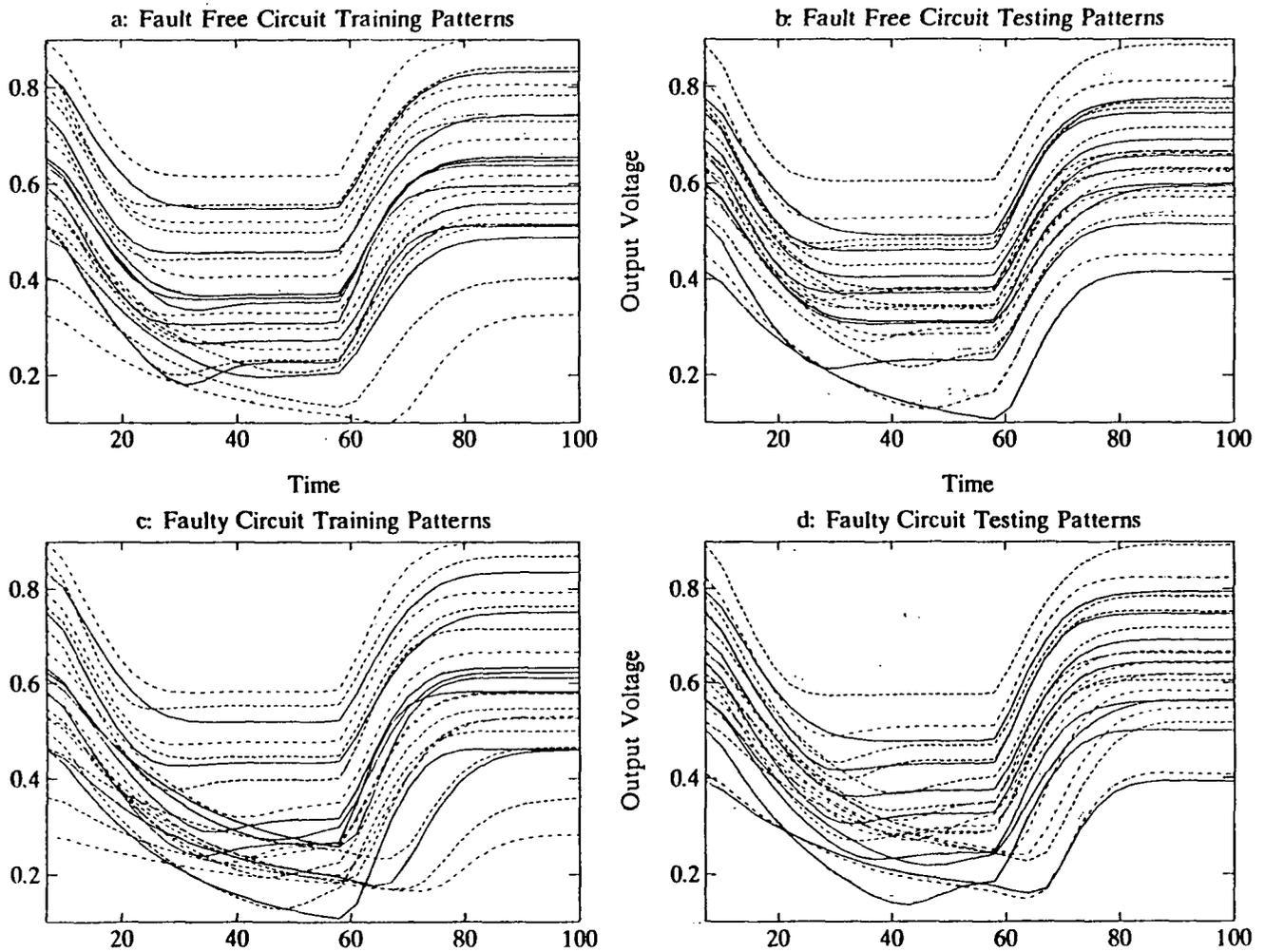


Figure 6: The neural network was trained with patterns in data sets a and c. The properly trained neural network can positively classify any patterns in all four sets. Note that the output voltage and time are normalized.

References

- [1] Brockman, J., and Director, S., "Predictive Subset Testing: Optimizing IC Parametric Performance Testing for Quality, Cost, and Yield," *IEEE Trans. on Semiconductor Manufacturing*, Vol. 2, No. 3, August 1989.
- [2] Funahashi, K. "The Approximate Realization of Continuous Mappings by Neural Networks," *Neural Networks*, Vol. 2, pp. 183-192, 1989.
- [3] Hansen, S., "SUPREM-III User's Manual," Tech. Rep. 8628, Stanford University, 1986.
- [4] Hornik, K., M. Stinchcombe, and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol. 2, pp. 359-366, 1989.
- [5] Kalos, M. H. and Whitlock, P. A., *Monte Carlo Methods Volume I: Basics*, John Wiley & Sons, 1986
- [6] K.A. Marko, L.A. Feldcamp and G.V. Puskorius, "Automotive Diagnostics Using Trainable Classifiers: Statistical Testing and Paradigm Selection," *Proc. Summer 1990 IEEE IJCNN*, San Diego, CA, pp. I-33, I-38, 1990.
- [7] S.R. Nassif, A.J. Strojwas and S.W. Director, "A Methodology for Worst-Case Analysis of Integrated Circuits," *IEEE Trans. on CAD*, Vol. 5, No. 1, pp. 104-112, 1986.
- [8] Pinto, M., et al., *PISCES-II Technical Report*, Stanford University, 1985.
- [9] Rumelhart, D., G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, Vol. 1, pp. 318-362, MIT Press, 1986.
- [10] J.A. Starzyk, M.A. El-Gamal, "Artificial Neural Network for Testing Analog Circuits," *Proc. 1990 IEEE ISCAS*, New Orleans, LA, pp. 1851-1854, 1990.
- [11] Strojwas, Andrzej J., "Pattern Recognition Based Methods for IC Failure Analysis," Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh PA, 1982.
- [12] White, H., "Learning in Artificial Neural Networks: A Statistical Perspective," *Neural Computation*, Vol. 1, No. 4, 1989.

Identification of the Connections In Biologically Inspired Neural Networks

H. Demuth, K. Leung and M. Beale
Department of Electrical Engineering
University of Idaho
Moscow, Idaho 83843
J. Hicklin
The Mathworks, Inc.
South Natick, Massachusetts 01760

Abstract - We developed an identification method to find the strength of the connections between neurons from their behavior in small biologically-inspired artificial neural networks. That is, given the network external inputs and the temporal firing pattern of the neurons, we can calculate a solution for the strengths of the connections between neurons and the initial neuron activations if a solution exists. The method determines directly if there is a solution to a particular neural network problem. No training of the network is required.

It should be noted that this is a first pass at the solution of a difficult problem. The neuron and network models chosen are related to biology but do not contain all of its complexities, some of which we hope to add to the model in future work.

A variety of new results have been obtained. First, the method has been tailored to produce connection weight matrix solutions for networks with important features of biological neural (bioneural) networks. Second, a computationally efficient method of finding a robust central solution has been developed. This later method also enables us to find the most consistent solution in the presence of noisy data.

Prospects of applying our method to identify bioneural network connections are exciting because such connections are almost impossible to measure in the laboratory. Knowledge of such connections would facilitate an understanding of bioneural networks and would allow the construction of the electronic counterparts of bioneural networks on very large scale integrated (VLSI) circuits.

Keywords: neural networks, biological models, neurons, identification

1 Introduction

Brains of living organisms are difficult to understand in part because they can contain billions of extremely complex neurons [6], [1]. However, there is an additional significant difficulty in that the effects of connections between the neurons cannot be measured

effectively or directly in the laboratory. Willows, for instance, reports some 20,000 measurements to determine which regions of a *Tritonia* would evoke responses of the mollusk's tens of neural cells, and goes on to point out that these experiments had to be conducted in reverse to find where the cells sent their messages [12]. His work, absolutely first class in every respect, illustrates the problem of determining the existence of neural connections, let alone evaluating their strengths.

Previous attempts to understand bioneural networks have involved direct measurement of physical properties or partial destruction of networks to observe the effects of such changes. Both approaches are long and tedious procedures. Our research suggests an alternative method of solving the problem of determining the strengths of connections between neurons. An understanding of the structure of bioneural networks would enhance the feasibility of neural repair, neural prosthesis design, and the construction of electronic equivalent circuits such as artificial visual or recognitions systems.

At present there is great interest in artificial neural networks that can recognize objects or patterns in complex data. The current approach of many investigators who wish to build a practical neural network to perform a specific task is to approximate a neuron, constrain the neural network connections to be feedforward, and train the networks by a method such as backpropagation [9]. Such networks bear little resemblance to bioneural systems, and it is not clear in each case whether a network of a specific size can be trained to do the desired job. However, networks of this kind have been trained to do interesting tasks, including some tasks that are not commonly done well by conventional computers or by any neurobiological system [8]. In contrast to the approach of many investigators, we have chosen neuron and network models that capture some of the most important aspects of the behavior of bioneural networks. Note that if one could understand bioneural networks, one could build them directly to solve some difficult problems.

The purpose of this paper is to present the neuron and network models that we have assumed, to show how our identification method proceeds and to tell of new results, including a way to find a robust central solution. The contributions of the paper lie in the novelty, power and simplicity of the method, which allows one to solve the "identification problem" for biologically inspired neural networks which are nonlinear systems with memory; and in various new results presented, including the development of an efficient method for finding a robust solution.

2 Neuron and Network Models

A recent DARPA Neural Network Study [3] defines a neural network as "a computational structure modeled on biological processes." We have chosen our neuron model and network model in that spirit. A schematic of the neuron model that we will adopt is shown in Figure 1.

In biological terms, the soma on the left of Figure 1 is the body of the neuron, and its axon projects to the right. External inputs impinge on the soma. Input signals from presynaptic neurons impinge on the soma and dendrites. Output signals are transmitted

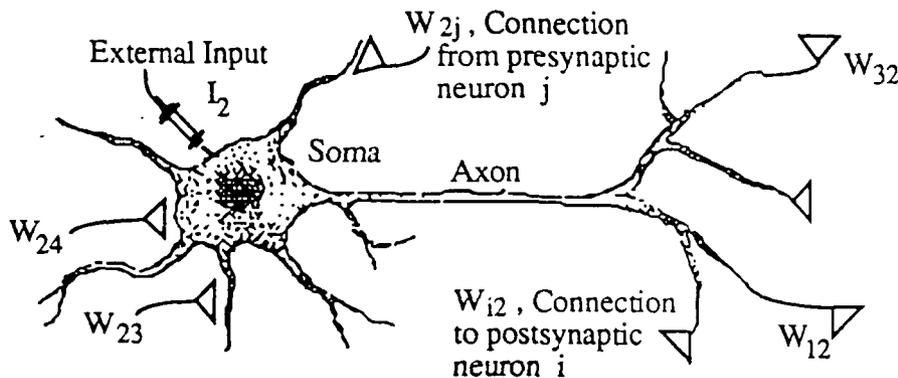


Figure 1: A Single Neuron, Neuron 2 (N2)

through the axon to the postsynaptic neurons on the right.

External inputs to the neuron and signals from other neuron's firings are integrated to form an "activation level." When a neuron's activation exceeds a "threshold" T , the neuron "fires", sending output signals to postsynaptic neurons and setting its own activation to the resting potential, taken here to be zero. The strengths of connections between neurons are given by the element w_{ij} . Thus, when neuron j (N_j) fires, it sends a signal of strength w_{ij} to N_i , and sets its own activation level to the resting potential. The set of all w_{ij} will be referred to as the weight matrix, \mathbf{W} , and the set of initial activations $a_i(1)$ will be referred to as the initial activation vector $\mathbf{A}(1)$. Our objective is to develop a method for finding a weight matrix \mathbf{W} and an initial activation vector \mathbf{A} for a particular neural network whose external inputs and temporal firing pattern are specified.

An equation for the activation level a_i of N_i in a neural network and definitions of the equation's terms follow.

$$a_i(t+1) = I_i(t) + (1 - F[a_i(t)])a_i(t) + \sum_{j=1}^N F[a_j(t)]w_{ij} \quad (1)$$

$$F[a_j(t)] = 1 \text{ if } a_j(t) \geq T \quad F[a_j(t)] = 0 \text{ if } a_j(t) < T \quad (2)$$

$a_i(t)$ is the activation of neuron i at time t ,
 w_{ij} is the strength of the connection from neuron j to neuron i ,
 $I_i(t)$ is the external input to neuron i at time t , and
 T is the threshold at which the neurons will fire.

An example of a two-neuron network with known connections and initial activations is shown in Figure 2.

As this figure shows in the diagram on the top and in the weight matrix \mathbf{W} , the strength of the connection from N_1 to itself is 11, the strength of the connection from N_2 to N_1 is

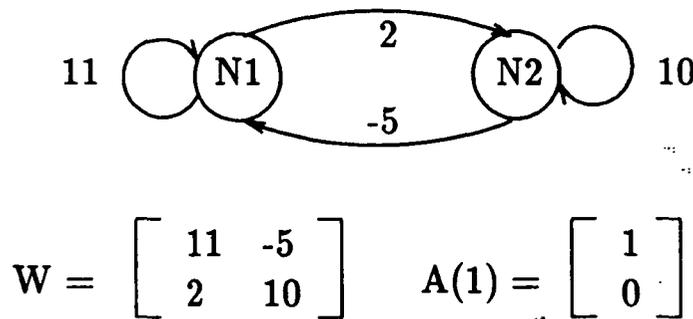


Figure 2: A Simple Two-Neuron Neural Network

-5, etc.. The expression for A shows that the initial activation of N1 is 1, which will cause N1 to fire at $t = 1$.

The networks need not be fully connected, and can be constrained in various ways if desired. For instance, a network can be made feed-forward by setting the upper right coefficients of W to be zero.

In our model, the weight matrix W is not a function of time. However, the method described here can be applied at various times to track changes in weight matrix connections with time.

3 Identification Method

The identification and design method pioneered by Hicklin [5] and used here will be illustrated through a simple example. Let us consider the following firing pattern of length $L = 6$ or a hypothetical two-neuron system with zero inputs:

Time t	1	2	3	4	5	6
Firing of N1	1	1	1	1	1	1
Firing of N2	1	0	1	0	1	0

Table 1: Two-neuron Firing Pattern

What can we deduce about the network producing this behavior? The fact that both neurons fired at $t = 1$ tells us that the initial activation of each neuron was above threshold and thus:

$$a_1(1) \geq T \quad (3)$$

$$a_2(1) \geq T \quad (4)$$

The fact that N1 fired on the second step tells us that its activation at $t = 2$ was also above threshold. What was its activation? From Equation 1 :

$$a_1(2) = I_1(1) + (1 - F[a_1(1)])a_1(1) + F[a_1(1)]w_{11} + F[a_2(1)]w_{12} \quad (5)$$

There were no inputs, so the first term is zero. Both neurons fired on the previous time step ($t=1$) so we know that $F[a_1(1)] = 1$ and $F[a_2(1)] = 1$. When we insert these values in the equation shown above, remembering that N1 fired on the second time step, we have:

$$a_1(2) = w_{11} + w_{12} \geq T \quad (6)$$

We have reached an important point in our development, so let us pause to consider what has been done. As shown in Equation 6 we have established the first weight matrix inequality. It constrains the sum of the two weight matrix elements, w_{11} and w_{12} . Any weight matrix that is to reproduce the specified firing pattern must satisfy this inequality. We can continue extracting inequalities from the firing pattern in this manner. When we have written $N * L = 12$ constraint inequalities we will seek a W and an A that will satisfy all of them. If a solution exists, the network using the W and A solution will produce the desired firing pattern. Now let us proceed with finding additional constraint equations. N2 did not fire on the second time step, so the mathematical process used to obtain the inequality of eq. 6 can be applied here to yield:

$$a_2(2) = w_{21} + w_{22} < T \quad (7)$$

As seen in eqs. 6 and 7, at $t = 2$ each neuron's behavior yielded an inequality relating the threshold and the strengths of the connections to that neuron from firing neurons. Note that the two weight matrix elements w_{11} and w_{12} of eq. 6 are in the first row of the weight matrix, while the two elements w_{21} and w_{22} of eq. 7 are in the second row of the weight matrix. Thus, when we seek a solution, we only need to solve two independent sets of inequalities (eqs. 6 and 7) each involving two elements of the weight matrix, rather than solving one set of inequalities involving all four weight matrix elements.

Proceeding in this way, six inequalities for N1 and six inequalities for N2 can be developed. The most common and well-known method of solving such a system of inequalities is the simplex method, which is described in most introductory linear programming texts [2]. When we apply the simplex method to the linear inequalities for this network we find that the following solution does exist. We have simulated a neural network with the

$$W = \begin{bmatrix} 2.00 & -1.00 \\ 1.00 & -1.00 \end{bmatrix} \quad A(1) = \begin{bmatrix} 1.00 \\ 1.00 \end{bmatrix}$$

initial conditions and weight matrix shown. The network does produce the firing pattern specified in Table 1.

4 Graphical Viewpoint

A weight matrix solution to each of two sets of equations such as those of eqs. 6 and 7 can be obtained graphically in two dimensions for the two-neuron case. This graphical technique can give some insight into methods for solving systems with more variables. In

Time Step	1	2	3	4	5	6	7	8	9
Input to neuron1	2	7	1	2	-2	8	1	-2	3
Input to neuron2	1	-4	1	4	2	1	-3	5	6
Input to neuron3	6	0	2	-2	1	6	9	2	3
Input to neuron4	2	1	-2	7	9	2	0	-3	1
Firing of neuron1	0	1	0	0	0	0	1	0	0
Firing of neuron2	0	1	0	1	0	1	0	0	1
Firing of neuron3	1	1	0	0	0	0	0	1	1
Firing of neuron4	1	1	1	0	1	1	1	0	0

Table 2: The Specified Behavior

Our computer program generated the inequalities and presented them to the simplex method. The solution shown below was found in about 5 seconds on an Apple Macintosh II computer.

$$W = \begin{bmatrix} -3.00 & -3.00 & -1.61 & -0.34 \\ 0.00 & -3.03 & 1.05 & 3.00 \\ -3.00 & -3.00 & -0.98 & 0.20 \\ 1.48 & -0.48 & -0.48 & -0.50 \end{bmatrix} \quad A(1) = \begin{bmatrix} 0.97 \\ -3.00 \\ 1.02 \\ 1.02 \end{bmatrix}$$

Table 3: The Designed Network

Next consider the probability of realization of firing patterns when there are no external inputs and when W and A are free variables. Our method was applied to determine the percentage of patterns realizable for up to ten neurons and ten time steps. The results in Fig. 4 show that the probability of realization increases as the number of neurons increase and as the number of time steps decrease.

6 Application To Biology

Obtaining data in the laboratory about bioneural network connections is extremely difficult. Many biologists, working for decades, have been able to establish some understanding of the neural connections in only a dozen or so crustacean "minibrains", each of which has ten to twenty neurons [10], [11]. Consequently, the prospect of applying our identification method to bioneural systems is exciting and we have begun to make our model more biologically realistic. For instance our method can be tailored so that it can calculate the weight matrix connections of networks having additional features of bioneural networks, as described below.

An individual bioneuron does not have direct connections from its axon back to its soma or dendrites. Thus, the diagonal terms w_{ii} of the weight matrix, W , must be constrained

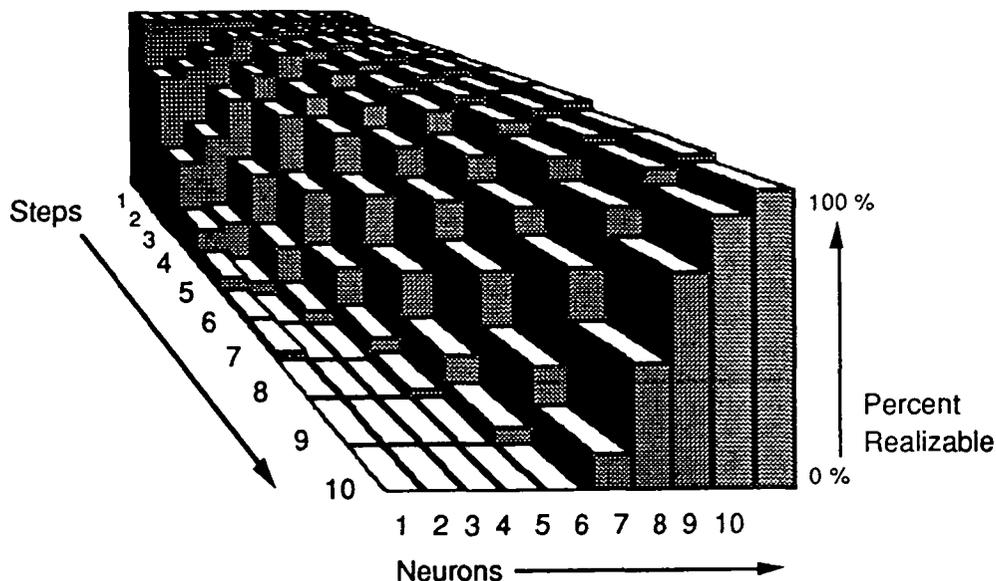


Figure 4: Percent of Patterns Realizable

to the value zero. Also, bioneurons commonly are either excitatory or inhibitory to any particular group of postsynaptic neurons so the columns of W will be either positive or negative.

The N diagonal terms of the weight matrix can be constrained to have the value zero by introducing N such constraints, one for each diagonal term. There are $2N$ possible sets of such column constraints and we can find a solution by using a tree search. The column constraints do not couple the N sets of inequalities associated with N neurons, so we are still dealing with L inequalities for each row. We have applied our method to design a variety of networks having zero diagonal and positive or negative column constraints. As additional constraints are introduced, the probability of realization of a network for a randomly generated temporal firing pattern decreases. We have run tens of thousands of cases on a Cray X-MP to determine the effects of the diagonal zero constraint and the column constraints and combinations of both. Each kind of constraint decreases the probability of realization. The decrease is especially significant in those cases where there are few neurons and/or many time steps. Details of these results will be reported in an upcoming M.S. thesis [7].

Thus, our method can be tailored to include additional biological features. This gives hope that bioneural connection strengths could be obtained from a manageable combination of experiments and calculations based on our method.

Now let us see how some of our viewpoints can be applied in a specific biological example. Consider a lobster's stomatogastric ganglion. This minibrain of some twenty neurons must produce a satisfactory firing pattern if the lobster is to survive. Suppose we were to examine the inputs to this minibrain, measure its firing pattern, apply our (extended and refined) method to identify a weight matrix solution, and map the associated solution space. The lobster could be using any solution in this space satisfactorily. The question is, "Where does he (or she?) live?". Presumably the lobster is not "living on the

edge", because the slightest indigestion might put him/her out of the solution space and thereby out of commission. The species has survived, so it must have solved the problems of establishing a solution space of sufficient size and of finding a robust solution, presumably in the central part of its solution space. We could apply our own inputs to the lobster's minibrain in a laboratory test to produce additional inequalities that would allow us to calculate the lobster's exact weight matrix solution. Knowledge of the lobster's solution space and his/her exact weight matrix solution would help us to predict and understand the effects on the lobster of neural damage, drugs, etc.

7 Robust Solution

Phase one of the simplex method as used above to identify networks produces a solution on the vertex of the solution space. Such a solution is not robust, for a slight change in connection strengths may result in the "network solution" no longer being in the solution space. In other words, the phase one simplex solution is not robust because it is at the edge of the solution space. We would prefer a solution which is in some sense "central" in the solution space.

A solution that is maximally distant from its nearest constraint can be obtained efficiently by using the second phase of the simplex method [4]. To do this, we define a special slack variable δ that is common to all constraints. This variable represents the minimum distance of a trial solution from each constraint. We direct phase two of the simplex method to maximize δ . It does so and produces the desired result directly and efficiently. Robust solutions for many networks have been found. All of the probabilities of realization for the cases involving diagonal and column constraints were obtained with this method. We look forward to using our methods to determine the solution space of a lobster, to finding if a particular lobster's actual solution is the same as our robust solution, and to comparing solution spaces and specific solutions for various lobsters.

Phase two of the simplex method can also be used to deal with bioneural systems having inconsistent firing patterns for which there are no realizable solutions. Such cases may occur due to changing connection weights arising from biological noises. To find a solution in these inconsistent cases, we again use the common slack variable δ , but on this occasion we find the solution associated with the least negative value for δ . Basically we have pushed each constraint boundary back a minimum amount until a point solution is obtained. A solution obtained in this way may be viewed as a central solution in a noisy space, with the value of δ being a measure of the noise inherent to the system under study. The model with such a solution will act much like the original system with the noise removed.

8 Summary of Results

We have found an identification method to determine the connections between artificial neurons of a network from the behavior of the network. We applied the method to identify

neural networks. We found that the probability of realization of firing patterns increases as the number of neurons increases and as the number of time steps decreases.

We identified network connection constraints associated with biological structures and found ways to incorporate them in our design methods. We applied our biologically tailored identification method to identify neural networks having biological network constraints such as zero diagonals.

We have developed a method for finding a central robust solution for neural networks and an analogous procedure for dealing with networks having inconsistent temporal firing patterns.

9 Conclusions

We have developed a new tool for identifying the connections of biologically-inspired neural networks. This tool is important because it is extremely difficult, if not impossible, to measure neural connections in bioneural networks. The availability of such a tool gives hope that a combination of experiments and calculations can be applied to determine the detailed structure of biological neural networks such as visual systems, and that the knowledge obtained would enable us to build their electronic counterparts on VLSI circuit chips. The method that has been developed also might be used to guide the repair of damaged biological neural networks, to develop neural prostheses and to build electronic recognition systems based directly on biological systems.

We recognize that our work is only a beginning and that we need to include more biological features of neurons in our model before it can reasonably approximate living systems. After improvements we hope to interpret firing patterns taken by neuroscience researchers.

10 Acknowledgements

We wish to thank Professor Gary Maki, Director of the University of Idaho's NASA Space Engineering Research Center, Professor Martin Hagan of Oklahoma State University, Professors Mark DeSantis and James Calvert of the University of Idaho, Professors Katherine Demuth and Mark Johnson of Brown University, and Margi Jenks for their suggestions, criticism, and support.

This work was funded originally by a research seed grant from the University of Idaho, and is currently funded by NASA Space Engineering Research Center grant NGW-1406 to the University of Idaho.

References

- [1] T. Bullock, R. Orkland and A. Grinell, *Introduction to Nervous Systems*, CA: W. H. Freeman and Co., San Francisco, 1974

- [2] J. Calvert and W. Voxman, *Linear Programming*, Harcourt Brace Jovanovich, New York, NY, 1989
- [3] "DARPA Neural Network Study, Executive Summary," MIT Lincoln Laboratory, Lexington, MA, 1988
- [4] Personal communication from Martin Hagan, March 19, 1990
- [5] J. Hicklin, "Neural Network Design," University of Idaho Ph.D. Dissertation, November 1988
- [6] E. Kandel and J. Schwartz, "Principles of Neuroscience," Elsevier, New York, NY, 1985
- [7] K. Leung, "Identification of Biologically-Inspired Neural Networks," M.S. Thesis, Computer Engineering, University of Idaho, 1990
- [8] D. Nguyen and B. Widrow, "The Truck Backer- Upper," Proceedings of the International Neural Network Conference (INNC-90- Paris), 1990, pp.399-407
- [9] D. Rumelhart and J. McClelland, *Parallel Distributed Processing*, MIT Press, Cambridge MA, 1986
- [10] A. Selverston and B. Mulloney, "Synaptic and Structural Analysis of a Small Neural System," in F. O. Schmitt and F. G. Warden (eds,) *The Neurosciences, Third Study Program*, pp 389-395, MIT Press, Cambridge MA, 1974
- [11] I. Selverston, "Neural Mechanisms for the Production of Cyclic Motor Patterns," *IEEE Trans on Sys, Man, and Cyb*, Vol smc-5, 1983, pp. 749-757
- [12] A. Willows, "Giant Brain Cells in Mollusks," *Scientific American*, Vol 224, No. 2, 1971, pp. 69-75

Auto-Programmable Impulse Neural Circuits

D. Watula and J. Meador

Department of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752

Abstract-

Impulse neural networks use pulse trains to communicate neuron activation levels. Impulse neural circuits emulate natural neurons at a more detailed level than that typically employed by contemporary neural network implementation methods. An impulse neural circuit which realizes short term memory dynamics is presented. The operation of that circuit is then characterized in terms of pulse frequency modulated signals. Both fixed and programmable synapse circuits for realizing long term memory are also described. The implementation of a simple and useful unsupervised learning law is then presented. The implementation of a differential Hebbian learning rule for a specific mean-frequency signal interpretation is shown to have a straightforward implementation using digital combinational logic with a variation of a previously developed programmable synapse circuit. This circuit is expected to be exploited for simple and straightforward implementation of future auto-adaptive neural circuits.

1 Introduction

Natural neurons compute by performing spacial and temporal integration of incoming impulse trains. In this paper, we describe circuits which implement artificial neural networks that operate in a similar fashion—by integrating trains of discrete pulses. Artificial neurons which work in this manner are desirable not only because they more closely mimic natural neuron functionality than other neural-network paradigms, but also because pulse-based communication inherits positive aspects of both digital and analog communication systems. Data can be represented without quantization effects by encoding information in the time period between identical digital pulses. This combination of analog signaling with digital waveforms is robust; even in a noisy environment, reliable detection of the presence of a pulse is simple. These advantages may have contributed to the natural selection of pulse-based communication in natural neural networks.

2 Information Representation and Interpretation

The impulse neural circuits we are working with communicate via the use of pulse frequency modulation (PFM). There are two readily apparent ways of interpreting the information

6.3.2

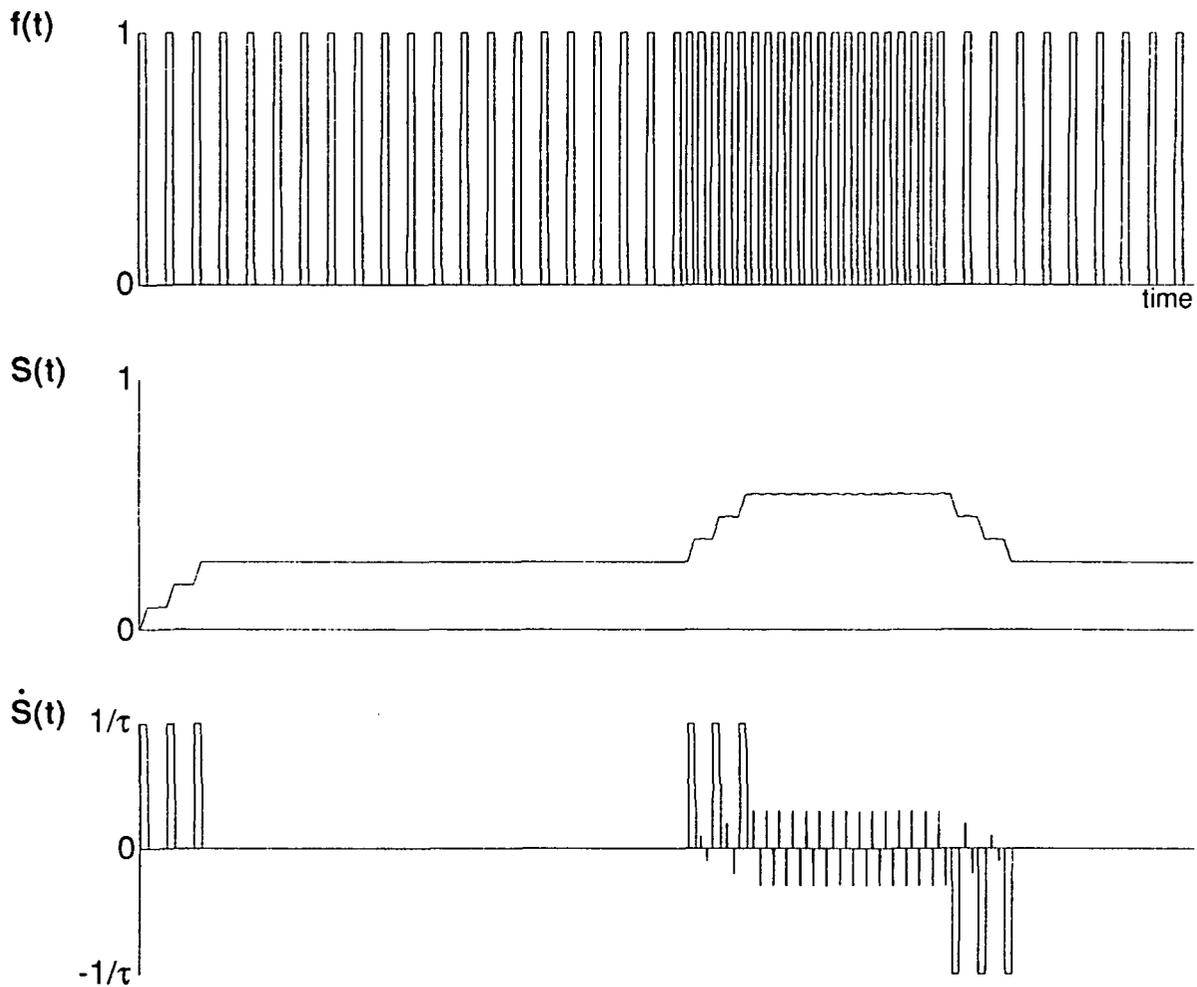


Figure 1: A pulse train $f(t)$, its estimated mean frequency signal $S(t)$, and signal velocity. Waveform $f(t)$, consisting of uniform $10\mu\text{s}$ pulses with $1\mu\text{s}$ rise and fall times, changes from 30kHz to 60kHz and back. The observation period is $\tau = .0001\text{s}$. $S(t)$ and its derivative were evaluated numerically

content of a PFM waveform. Information may be perceived as being contained in the time interval between pulses or in the average of these intervals: equivalently, it is contained in either the instantaneous frequency of the pulse train or in its mean frequency. One advantage of the mean frequency interpretation is that it is easily estimated. The signal value is equivalent to the average frequency as estimated by the reciprocal of the observed average firing period, which is in turn proportional to the number of pulses which occur over an observation period.

Investigations into the nature of PFM communication in neurons often emphasize the mean-frequency interpretation. The integral-pulse-frequency-modulated (IPFM) signal model for natural neural systems provides a case in point. The spectrum of an IPFM signal contains a DC-referenced component proportional to the modulating signal that can be easily recovered by low-pass filtering or averaging. This signal model suggests these operations are the most likely methods employed by natural neurons to recover signal information, as they are the simplest and otherwise the only ways in which distortion-free demodulation can be achieved [1].

The integrate-and-fire neuron model used to generate IPFM signals accurately describes the operation of the impulse neural circuits we are investigating. This makes the time-averaged signal interpretation useful for analyzing impulse neural circuits.

In the remainder of this paper we will assume a time-averaged signal interpretation over some arbitrary period τ . Under the assumption of a continuous pulse waveform, the signal $S(t)$ represents the average pulse firing rate:

$$S(t) = \frac{1}{\tau} \int_{t-\tau}^t f(\sigma) d\sigma \quad (1)$$

Since the signal is represented by the time average of the pulse train, it has been noted that the pulse train itself should contain information regarding the time-differential (or velocity) of the signal [Gluc88]. In fact, simple application of a fundamental theorem of calculus yields:

$$\dot{S}(t) = \frac{1}{\tau} (f(t) - f(t - \tau)) \quad (2)$$

This well-known formulation proves useful in the analysis of the impulse neural circuit to be described as well as in the development of a Hebbian learning rule which is particularly well suited to VLSI circuit implementation. Figure 1 illustrates a PFM coded signal and its derivative. A similar formulation using a low pass filter signal representation also allows for stable numerical computation of the instantaneous mean frequency and its derivative [3]. The application of the low pass signal representation to VLSI implementation is beyond the scope of this paper, but may have future utility.

3 Impulse Neural Circuits

The impulse neural circuits we have been developing incorporate a blend of analog and digital circuit implementation styles [8]. There are two principle subcircuits utilized, namely

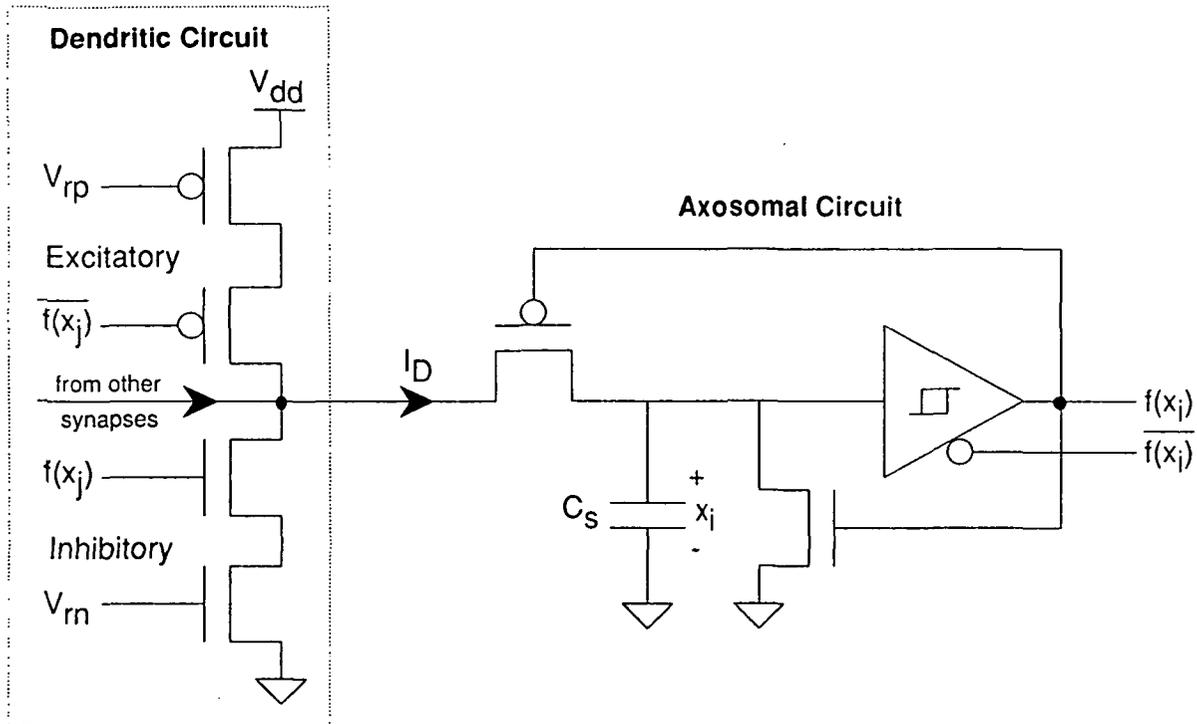


Figure 2: Axosomal and fixed-synapse circuits

axosomal and synapse circuits. The axosomal circuits compute a nonlinear summation of input signals whose magnitudes are weighted by the synapse circuits. Together they implement an integrate and fire mechanism which corresponds to one of several commonly used models of natural neuron function [6],[2],[1].

3.1 Axosomal Circuit

The axosomal circuit (Figure 2) is a simple current controlled relaxation oscillator. The circuit has two discrete states of operation, the input integration state and action potential generation. The Schmitt trigger establishes two threshold potentials which determine when the neuron moves between these states. Feedback to two FETs, one in series with and one shunting capacitor C_s , determines whether it is being charged by net input excitation during the integration state or being discharged during an action potential. Instantaneous membrane potential is represented by voltage x_i across the capacitor. The upper Schmitt trigger threshold voltage (V_{th}) emulates the natural neuron action-potential threshold while the lower (V_U) emulates the membrane potential at which the voltage-gated ion channels close.

During the integration state, $x_i < V_{th}$ and the series PFET conducts charge to or from C_s at a rate dependent upon the net result of excitatory and inhibitory stimuli (expressed by I_D in Figure 2). Over this period, the shunting NFET conducts no current. When x_i reaches V_{th} , $f(x_i)$ switches from 0 to 1, switching off the series PFET and driving the shunting NFET into the active operating region. This is the action potential state of the circuit. The capacitor discharges toward ground at a rate determined by C_s , V_{th} , and the

NFET characteristics. When $x_i = V_u$, the circuit switches back to the accumulation state and one firing cycle is complete. Circuit operation continues in this manner at a rate which is dependent upon the net excitation from the dendritic current.

3.2 Synapse Circuits

Two dendritic circuits have been developed, a fixed synapse and a programmable synapse. Using a parallel-connected combination of these synaptic cells, networks with any combination of inhibitory and excitatory connections can be constructed. Additionally, these circuits can emulate additive or shunting connections.

Figure 2 depicts the inhibitory and excitatory fixed synapse circuits. In each two-transistor circuit, one FET behaves as a voltage-controlled switch for converting uniform pulses into weighted, discrete charge packets that are conducted to or from an axosomal summing capacitor. A second FET determines the synaptic weight, or size of the charge packet. Depending on the choice of reference voltage (V_{rp} or V_{rn}), the pair acts as either a switched current source in an additive synapse or a switched resistor in a shunting connection. Further control over synaptic efficacy is attained by customizing FET W/L ratios. Thus, fixed networks may be implemented simply by tailoring device geometries and selecting appropriate bias voltages.

Unfortunately, experimental work has shown that it is difficult to select these parameters based on weight values obtained from conventional neural-network training techniques. More traditional paradigms, such as error backpropagation, employ abstract models which do not exhibit the nonlinear response or process variations found in these electronic circuits. One solution to this problem is the application of auto-adaptive circuitry which compensates for circuit imperfections.

The programmable synapse constitutes a portion of that auto-adaptive circuitry. Specifically, it stores the current adaptive state (connection weight) in a non-volatile fashion as the threshold voltage (V_{th}) of a floating-gate FET. The programmable synapse circuit (Figure 3) consists of five transistors, two of which share a common floating gate. In combination with reference FET Q_1 , the circuit acts as a switchable current source controlled by reference voltage V_{rn} and the impulse gating signal $f(x_j)$. With V_{rn} adjusted such that Q_4 operates in the subthreshold region, a small shift of V_{th} induces a significant change in the Q_4 source current. With Q_1 properly biased, this current controls the amount of current flowing into the dendritic summing node (I_D). With V_{rn} fixed, variations in V_{th} for Q_4 will vary I_D over several orders of magnitude when Q_3 conducts.

The circuit is programmed by the application of voltage pulses V_{pp} to the control gate of Q_5 , causing a high electric field to form in the dielectric between the gate and the substrate. This will induce charge tunneling between the substrate and the floating gate, where the carriers will become trapped. Any trapped carriers will redistribute themselves between Q_4 and Q_5 since they share the conductive polysilicon floating gate. As a result, the V_{th} of both transistors will be shifted slightly. A positive V_{pp} causes negatively charged electrons to become trapped in the floating gate, which in turn causes I_{ds} to drop below I_{ref} . There will then be an increase in the current flowing into the summing node. The

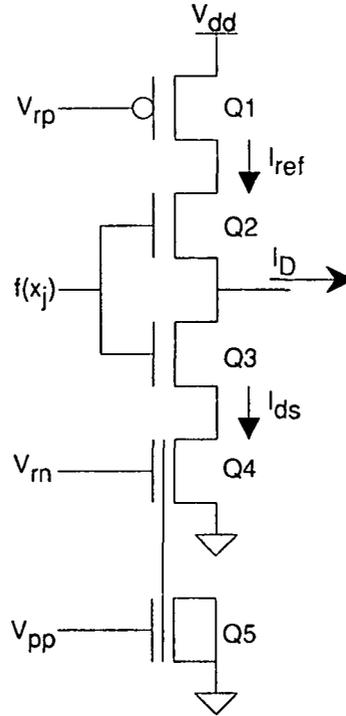


Figure 3: Programmable synapse circuit

opposite happens for a negative V_{pp} . Note that, unlike the fixed synaptic cells, a single programmable synapse may change from excitatory to inhibitory or vice versa.

3.3 Signal transfer function

The transfer function computed by an impulse neural circuit can be derived under the assumption that the chief information carrying component of the impulse train is its average value. The impulse neural circuit which we have described can be shown to use time-averaging to demodulate input signals. It can also be shown that the average output signal is a nonlinear function of a weighted summation of averaged input signals.

The averaged output of an impulse circuit which integrates input signals during both the integration and action potential state is given by:

$$S_o(t) = \frac{1}{\tau} \int_{t-\tau}^t f(\sigma) d\sigma = \frac{AT_0}{C(V_{th} - V_u)} \sum_{j=1}^{N_I} \beta_j \frac{1}{\tau} \int_{t-\tau}^t h(f_j) d\sigma = \frac{AT_0}{C(V_{th} - V_u)} \sum_{j=1}^{N_I} \beta_j S_j \quad (3)$$

for observation interval τ . A, T_0 are the output pulse amplitude and width, C is the integration capacitance, h is the nonlinear transconductance relation associated with synapse j , β_j is the transconductance factor of synapse j , and V_{th}, V_u are as defined previously. N_I is the total number of inputs.

This general result shows that the output mean firing rate is a linear weighted summation of input mean firing rates for a circuit which integrates inputs over the entire observation period τ , regardless of the circuit state. Such is not the case, however, in

the actual neuron circuit. Input signals are not integrated during the action potential state. This means that the circuit output will eventually saturate with increasing net excitation. A nonlinear activation function can be derived for the circuit using a special case of (3) where the observation interval is $\tau = T_0 + T_1$, with T_1 representing the integration period for a single output pulse:

$$S_o(t) = \frac{AT_0}{C(V_{th} - V_{tl})} \sum_{j=1}^{N_I} \beta_j \frac{1}{T_0 + T_1} \left(\int_{t-T_0-T_1}^{t-T_1} h(f_j) d\sigma + \int_{t-T_1}^t h(f_j) d\sigma \right) \quad (4)$$

Since no input signals register an effect during the action potential state, the first integral term evaluates to zero. Algebraic manipulation of the resulting equation yields the following expression for T_1 :

$$T_1 = \frac{C(V_{th} - V_{tl})}{NET} \quad (5)$$

where

$$NET = \sum_{j=1}^{N_I} \beta_j \hat{S}_j$$

where \hat{S}_j represents an approximation of the input signal over the (shorter) observation interval T_1 and NET is the net input excitation (weighted sum of input signal approximations). The actual input/output relationship computed by the circuit becomes:

$$S_0 = \frac{1}{T_0 + T_1} = \frac{1}{T_0 + \frac{C(V_{th} - V_{tl})}{NET}} \quad (6)$$

Here, S_0 expresses the instantaneous firing rate of the neuron output signal as a nonlinear weighted sum of mean firing rates observed over the integration period. This function is sigmoidal, saturating as NET approaches infinity and dropping to zero with $NET = 0$. This result agrees precisely with that obtained via the more specific analysis reported in [8], and concurs with circuit simulation results.

A series of SPICE simulations (Figure 4) of an axosomal circuit extracted from a 2μ CMOS layout shows very close agreement with the form of (6) over at least six orders of magnitude of input current. Preliminary measurements from a prototype of the axosomal circuit are also presented (Figure 5). Although the data exhibits the same exponential slope and saturation characteristic as the simulation, it also manifests an awkward transition into saturation that has not yet been explained. The large offset in the data before the saturation region is attributable to the input capacitance of the chip, which is significant since the extracted value of C_{subs} is less than $.2pF$.

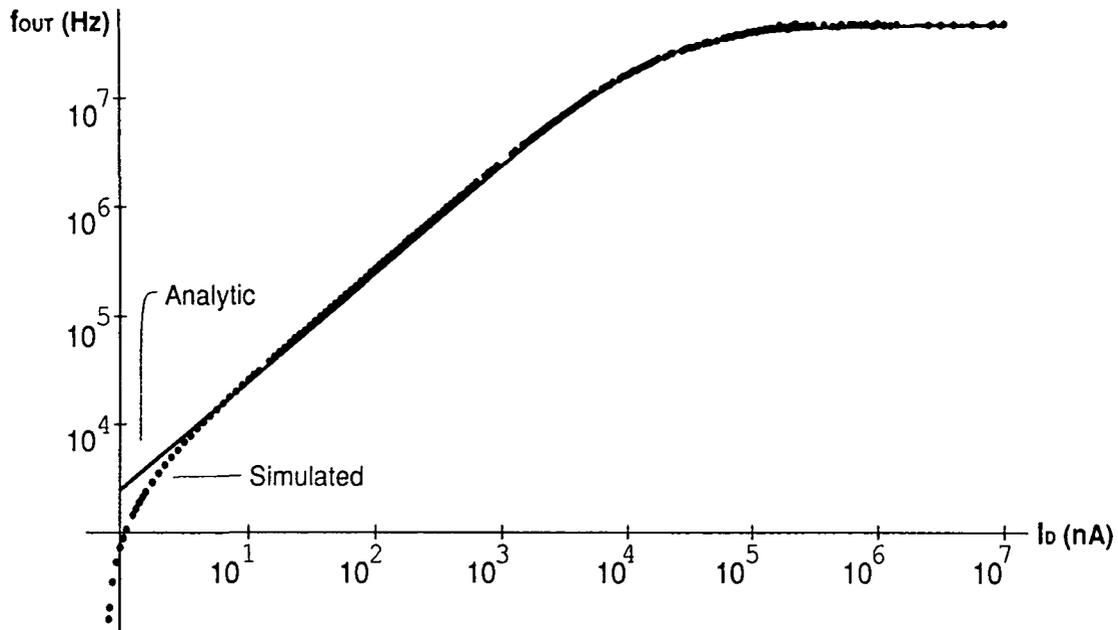


Figure 4: Comparison of simulated transfer characteristic with Equation 6 for $C = .18pF$, $T_0 = 2.2 \times 10^{-8}s$, and $V_{th} - V_{tl} = 2.2v$

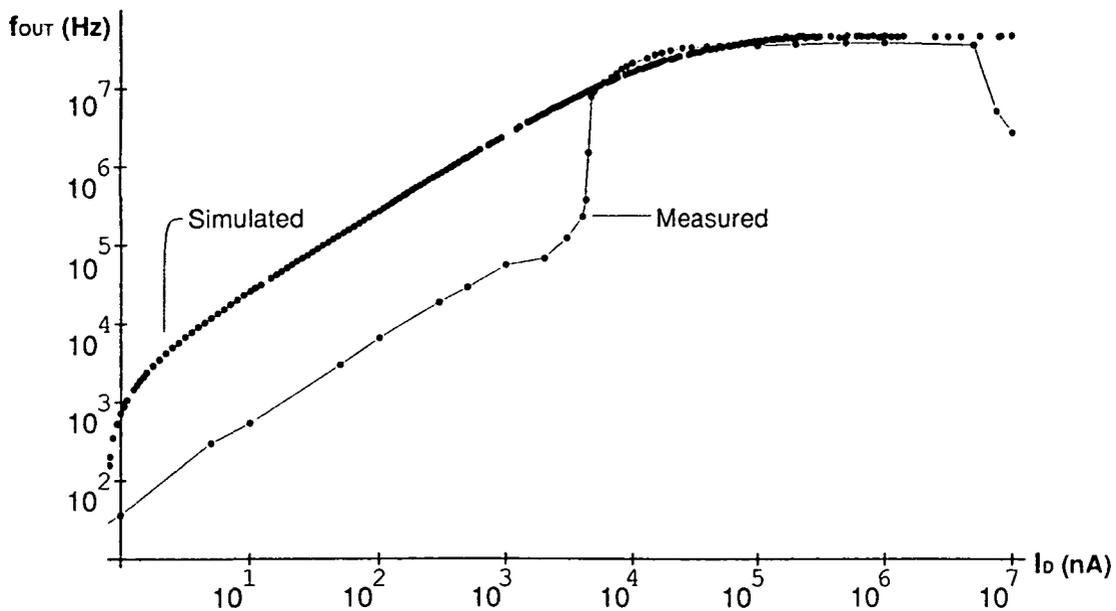


Figure 5: Comparison of simulated transfer characteristic with measurements from a prototype neuron

4 Auto-Adaptation in an Impulse Neural Circuit

There exist three general neural network adaptation strategies. Neural networks can be trained using supervised, unsupervised, or reinforcement methods. Any algorithms which update connection weights as a function of some error between an output and target vector are supervised. Error backpropagation is one of the best known examples of a supervised learning algorithm. Supervised adaptation not only updates the connections when errors occur, but also gives some indication about how to update the connections to reduce error. Unsupervised training methods use no error feedback whatsoever to adjust connection weights. Their chief advantage is the exclusive use of local information from pre- and post-synaptic neurons. The reliance on only local information helps simplify VLSI implementation of the learning rule. Reinforcement methods utilize output feedback to update connection weights, but without specific error gradient information. Reinforcement learning can use either global or local information, and provides a good model for classical conditioning phenomena which have been observed in natural neurons.

Natural neural systems adapt to the environment while processing trains of pulses in the apparent absence of direct performance feedback, so unsupervised learning provides one model for adaptation in such systems. It is our goal to implement auto-adaptive VLSI neural circuits, so a natural alternative is to begin with algorithms that use locally available information to update connection strengths. Unsupervised learning is a logical candidate for this.

At least four types of unsupervised learning have been described, including signal Hebbian, differential Hebbian, competitive, and differential competitive learning. This algorithm class exhibits a simplicity and locality not present in more complex training algorithms such as backpropagation and ART. Our work thus far has focused upon Hebbian rules exclusively, even though extensions based on competitive learning may also be possible. Hebbian style adaptation rules are found in adaptive bidirectional associative memories (ABAMs) [Kosk89], and in reinforcement generation for classical conditioning models. Hebbian rules have also been used in principle component analysis for feature extraction in lieu of computationally complex matrix algebra [9],[7],[5].

Signal Hebbian adaptation is expressed as a correlation of pre-synaptic and post-synaptic neuron firing rates:

$$\dot{w}_{ij} = \beta S_i(t) S_j(t) \quad (7)$$

where $S_i(t)$ represents a mean firing rate of neuron i at time t and β is an adaptation rate constant. It is important to note that this expression is independent of the type of frequency averaging used, so its meaning can vary with the signal interpretation. Even though the signal Hebbian rule can account for learning, it cannot account for "unlearning", since connection strengths monotonically increase with signal correlations.

The differential Hebbian adaptation rule can be expressed as the correlation between the rate-of-change (velocity) of pre- and post-synaptic signals:

$$\dot{w}_{ij} = \beta \dot{S}_i \dot{S}_j \quad (8)$$

When used for reinforcement generation, differential Hebbian has been shown to account for several aspects of classical conditioning behavior which go beyond the simple acquisition accounted for by signal Hebbian learning.

A differential Hebbian learning rule for the mean frequency signal representation expressed in (1) can be implemented using simple digital logic in conjunction with a variation of the adaptive synapse circuit of Figure 3. Given the relation described in (2) it can be shown that the product of two signal velocities can be computed using the relation:

$$\tau^2 \dot{S}_1(t) \dot{S}_2(t) = f_1(t)f_2(t) - f_1(t)f_2(t - \tau) - f_2(t)f_1(t - \tau) + f_1(t - \tau)f_2(t - \tau) \quad (9)$$

Since all $f_i(t)$ are binary signals having the value "1" when an action potential is occurring at time t , much of the computation can be carried out by simple combinational logic. This representation is naturally compatible with impulse neural circuits, as the binary pulse train signal representation allows for the reduction of the multiply operations to simple logical AND functions.

Using (1) to establish a signal interpretation based upon inverse mean period allows for the design of a simple auto-adaptive synapse circuit. Figure 6 diagrams an implementation of an auto-adaptive differential Hebbian synapse cell which is currently being developed. The binary products are computed by the AND functions while their arithmetic combination is computed by charge summation in the floating gate circuit. Delayed versions of the input signals are derived from simple binary delay cells. Only one delay cell is required per neuron circuit. The configuration shown will exhibit some features of classical conditioning if f_1 is the output from the post-synaptic neuron and f_2 is any input signal. As such, the circuit computes reinforcement signals which contribute to both connection acquisition and extinction.

5 Conclusion

We have shown that a differential Hebbian adaptation rule can be implemented in a simple and straightforward manner in an impulse neural circuit. The results of this work are expected to contribute to the implementation of classically conditioned neural circuits and adaptive bidirectional associative memories as well as any other network training algorithm based upon a differential Hebbian learning rule.

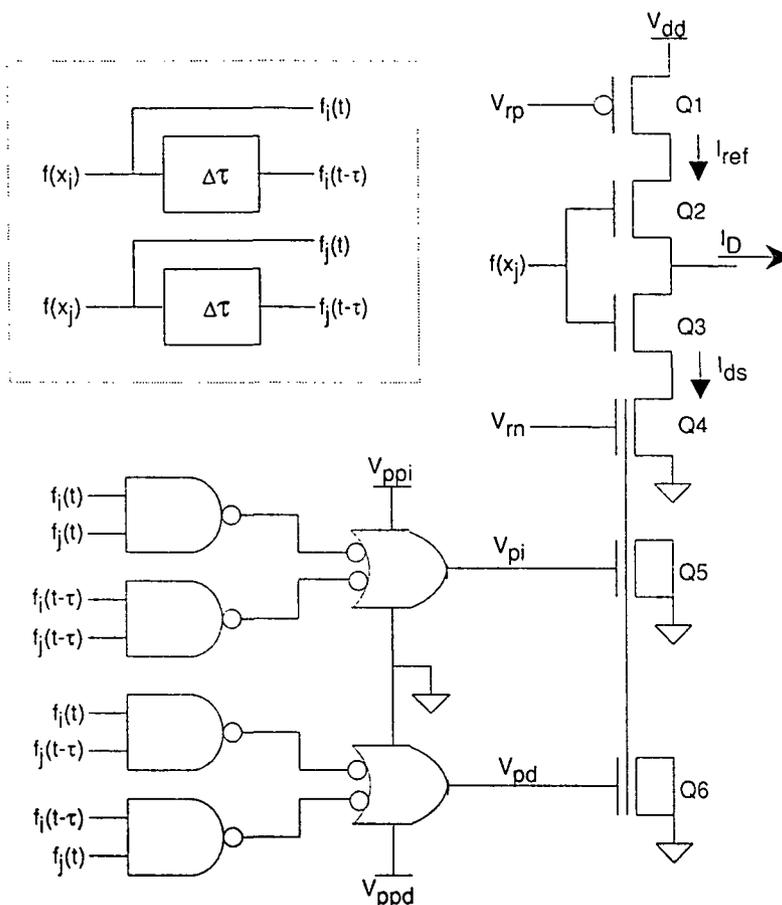


Figure 6: An autoadaptive differential Hebbian synapse cell representing the weight from neuron j to neuron i

References

- [1] Bayly, E., "Spectral Analysis of Pulse Frequency Modulation in the Nervous Systems," *IEEE Trans. on Bio-Medical Engineering*, Vol. BME-15, No. 4, pp. 257-265, 1968.
- [2] Gestri, G., "Pulse frequency modulation in neural systems: a random model," *Biophys J.*, Vol. 11, pp. 98-109, 1971.
- [3] Gluck, M.A., D.B. Parker, E. Reifsnider, "Some Biological Implications of a Differential Hebbian Learning Rule," *Psychobiology*, Vol. 6 No. 3, pp. 298-302, 1988.
- [4] Kosko, B., "Unsupervised Learning in Noise," *Proc. IJCNN*, Washington D.C., pp. I-7, I-17, 1989.
- [5] Kung, S.Y., "Orthogonal Learning Network for Constrained Principal Component Problem," *Proc. IJCNN*, San Diego CA., 1990.

6.3.12

- [6] Hoppensteadt, F.C., *An Introduction to the Mathematics of Neurons*, Cambridge University Press, Cambridge, 1986.
- [7] Leen, T., M. Rudnick, and D. Hammerstrom, "Hebbian Feature Discovery Improves Classifier Efficiency," *Proc. IJCNN, San Diego CA.*, 1990.
- [8] Meador, J, et al., "Programmable Impulse Neural Circuits," *IEEE Trans. on Neural Networks* (to appear).
- [9] Oja, E., "A simplified neuron model as a principal component analyzer," *J. Math. Biology*, Vol. 15, pp 267-273, 1982.

High Performance VLSI Telemetry Data Systems

J. Chesney, N. Speciale, W. Horner, S. Sabia
Data Systems Technology Division
Mission Operations and Data Systems Directorate
NASA, Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract - NASA's deployment of major space complexes such as Space Station Freedom (SSF) and the Earth Observing System (EOS) will demand increased functionality and performance from ground based telemetry acquisition systems well above current system capabilities. Adaptation of space telemetry data transport and processing standards such as those specified by the Consultative Committee for Space Data Systems (CCSDS) standards and those required for commercial ground distribution of telemetry data, will drive these functional and performance requirements. In addition, budget limitations will force the requirement for higher modularity, flexibility, and interchangeability at lower cost in new ground telemetry data system elements. At NASA's Goddard Space Flight Center (GSFC), the design and development of generic ground telemetry data system elements, over the last five years, has resulted in significant solutions to these problems. This solution, referred to as the *functional components approach* [1], includes both hardware and software components ready for end user application. The hardware functional components consist of modern data flow architectures utilizing Application Specific Integrated Circuits (ASICs) developed specifically to support NASA's telemetry data systems needs and designed to meet a range of data rate requirements up to 300 Mbps. Real-time operating system software components support both embedded local software intelligence, and overall system control, status, processing, and interface requirements. These components, hardware and software, form the superstructure upon which project specific elements are added to complete a telemetry ground data system installation.

This paper describes the *functional components approach*, some specific component examples, and a project example of the evolution from VLSI component, to basic board level functional component, to integrated telemetry data system.

1 Introduction

To insure the "routine" nature of space activities in the future, many essential science and engineering developments must occur over the next ten years. One of the most crucial of these developments is the evolution of current space telemetry systems. The widespread

6.4.2

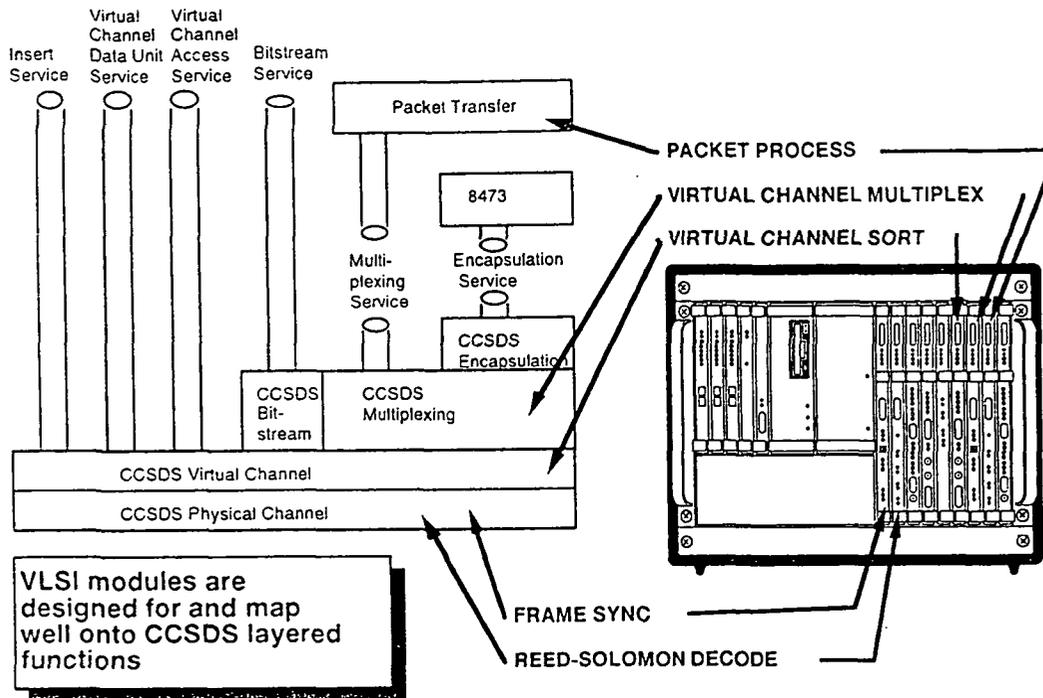


Figure 1: VLSI Systems Mapping To CCSDS Layered Architecture

development and/or adoption of various international standards affecting space telemetry data (CCSDS, CCITT, ISO etc.) promises a great potential to acquire, exchange, process, and distribute space telemetry data with fewer unique system designs. This step, however, is only one part of the required solution. In fact, the adoption of these standards without significant advancements in ground and flight system elements has caused some measure of concern on the part of developers attempting to provide the added functionality required to meet new standards at today's higher data rates (up to hundreds of Mbps).

The goals of today's space telemetry system developers are very similar to those of early television pioneers. Technology had provided a revolutionary system for the transmission and reception of information. The great goal of those individuals was the very broad distribution of this technology to many potential providers and consumers. Three essential elements were required (from a purely developmental perspective) to meet this goal. First, standards were required from which a broad range of commonality in performance and functionality could be insured. Second, development of essential engineering capability to meet the performance and functional standards. Third, the development of low cost, easy to use, and reliable consumer products that apply the fundamental engineering advances to the standards required to support the television industry.

Standards, the first of these elements, are well underway in most areas of space telemetry system development. With so many advances in basic hardware and software engineering capability today, the second element, the development of essential engineering capability, has, in most regards, been met. The development of low cost, easy to use, and reliable consumer telemetry products is becoming more of a reality today and is the basic subject of this paper.

2 Functional Components

The Data Systems Technology Division (DSTD), at NASA's Goddard Space Flight Center (GSFC) has, over the last five years, applied state of the art technology to enhance the performance and reduce the costs of NASA ground telemetry data system design and development. As part of this effort, the MicroElectronic Systems Branch of the DSTD, has designed and developed a variety of generic hardware and software processing elements used to capture, process, and distribute space telemetry data. The functional components approach includes a general philosophy which attempts to combine these basic elements into larger functional components which are easily integrated into a high performance, configurable, low cost, and high reliability space telemetry system. The key elements in this approach are Very Large Scale Integrated (VLSI) circuits and advanced, highly integrated, real time software system environments. The VLSI devices meet the high performance, small size, and low cost associated with the more standard functions required in modern telemetry systems, while the real-time software systems provide the necessary flexibility to meet project specific needs. The high densities and clock speeds now available from a variety of commercial programmable logic devices allow these devices to meet nearly all other combinational logic needs. The general requirements for control, status, and data exchange/processing are supported by extensive use of commercially available:

1. modules (e.g. CPUs, memory & I/O, etc.);
2. system environments (e.g. VME, Multibus, NuBus, etc.);
3. communication protocols (Ethernet/FDDI); and
4. software languages, real-time operating systems, and support utilities.

These systems provide easy access to next stage computer processors often required for higher level data processing of space telemetry data. Figure 1 shows a mapping of a functional component system to the CCSDS layered system architecture. The actual line between the various hardware, firmware, and software implementations is actually quite variable but on a steady course to higher VLSI integration providing ever more performance, functionality, small size, lower cost, and higher degrees of reliability than thought possible just a few short years ago.

3 System Overview - Platforms

The VME open bus system and the Apple Macintosh II NuBus system are the two main platforms presently being used to support telemetry data system development and operation under the functional components approach. These systems presently support data rates up to 20 Mbps and a broad range of space telemetry functionality such as synchronization, Reed/Solomon decoding, packet processing, Virtual Channel sorting/multiplexing, data simulation, and high density (256x256) cross-point switching. Architectural stability over the spectrum of supported functionality is required in order to minimize design

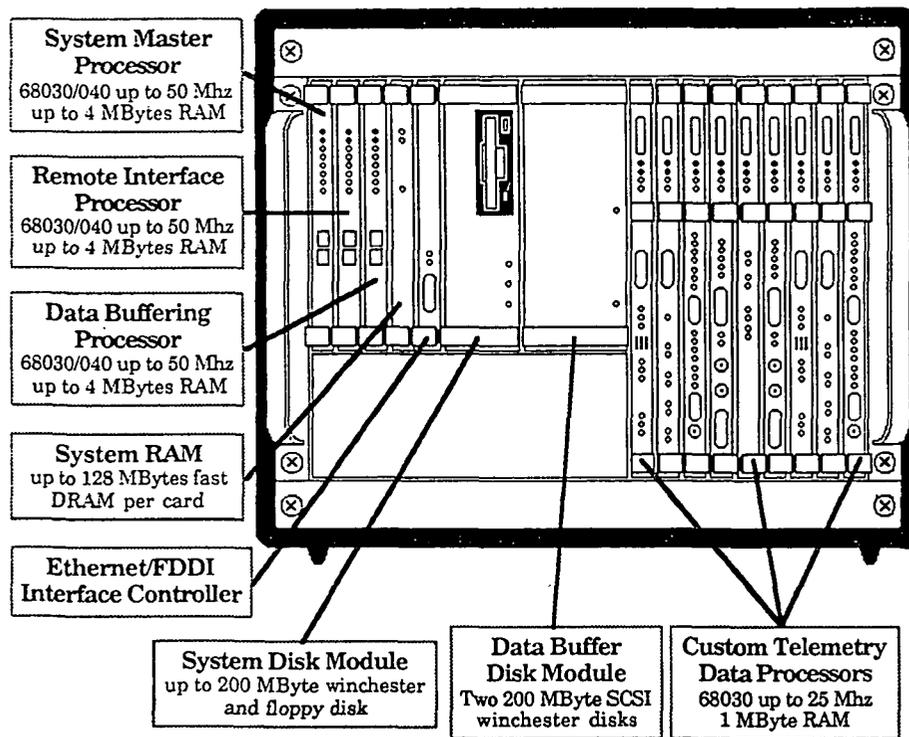


Figure 2: Typical VME platform

changes and error introduction when providing various degrees of performance for the same function. This implies that once a functional component design has been completed and tested, a new design for that same function to increase bit rate (for example) will use the same architecture to the maximum extent possible. As a result of this policy, the architecture and the actual logic design for the Gallium Arsenide (GaAs) frame synchronizer VLSI device (300 Mbps rate) are nearly identical to the Complementary Metal Oxide Semiconductor (CMOS) technology device used to support much lower bit rates (to 50 Mbps). Current activity utilizing both Emitter Coupled Logic (ECL) and GaAs logic gate arrays will provide a general capability to 300 Mbps for virtual channel sorting and multiplexing by June 1991 [2].

The basic configuration of a typical VME system generally consists of:

1. VME standard open bus enclosure, power supply, and a 21 slot backplane including a VME bus (J1 & J2) and a custom telemetry bus (J3);
2. a selection of commercial VME card modules;
3. a selection of GSFC custom cards (A/B/C channel cards);
4. an operator's console;
5. a real-time, multi-processor operating system environment; and

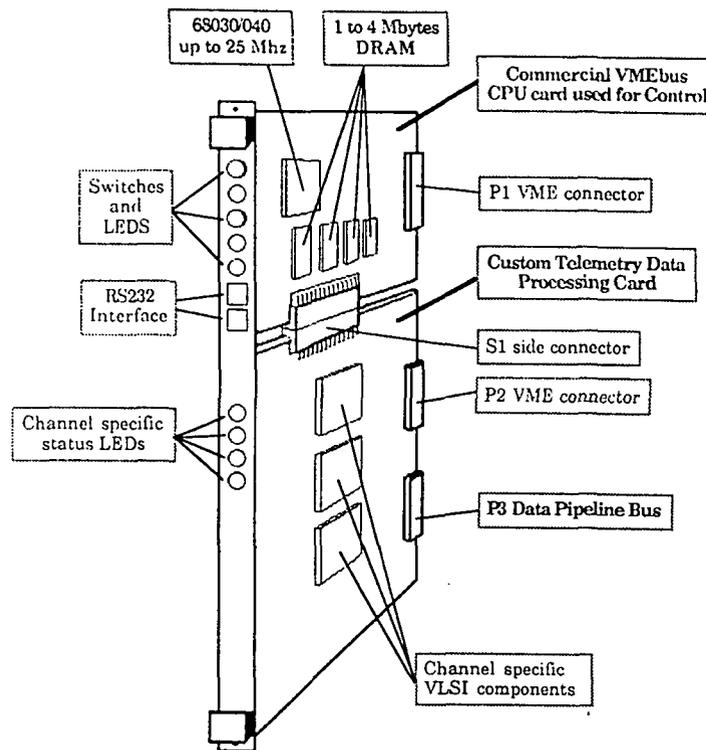


Figure 3: Custom card diagram

6. application programs supporting specific application requirements [3,4].

The physical diagram of a typical VME configuration is shown in Figure 2.

The VME environment and its commercial cards are used to support the general exchange and storage of control, status, and quality data required for local and remote operator console and control operations. These elements provide the general CPU processing, storage, and I/O functions required by the overall system application including the network (e.g. Ethernet, FDDI etc.) or host interface. They also initiate diagnostic tasks and format test results for presentation as required.

Custom cards and the third bus connector (9U VME card cages are used) comprise the telemetry data pipeline. This pipeline allows both incoming and outgoing telemetry data to be processed and transferred through the system without overburdening the VME bus with every data word transfer. This "pipelined" technique allows great flexibility in the processes performed on the telemetry data and as well as for increased data bandwidth. The custom cards directly support the CCSDS standards for telemetry data acquisition and processing through the extensive use of the NASA VLSI Application Specific Integrated Circuits (ASICs) developed at GSFC. Each custom card is composed of a custom hardware section especially designed to implement a particular NASA communication function(s) and a commercial CPU section used for local and global control/status exchange (Figure 3). Some very high performance cards (e.g. packet processor card) require up to three 68020/30 CPU mezzanines (custom built) over a full 9U card area of support hardware. These

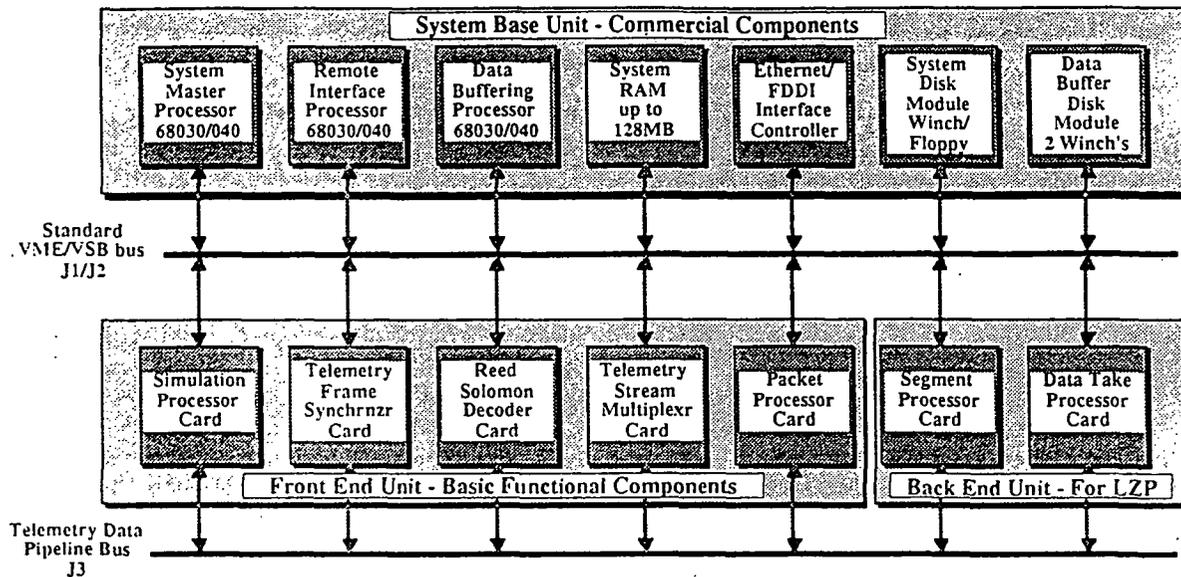


Figure 4: Typical VME block diagram

components and card systems provide programmability to support format or operation variations normally encountered during data capture and process operations.

The block diagram of this general VME implementation shown in Figure 4 demonstrates the flexibility and power available to the developer and to the end user. Because the custom cards adhere to VME/VSB standards and because of the flexibility of the telemetry software environments supporting this system (see SOFTWARE COMPONENTS), virtually any commercial VME card or card system can be easily ported to this environment. The separate telemetry pipeline bus offers great flexibility at the telemetry channel interface. The processing of telemetry data is actually a pipeline process by which telemetry data moves from custom card to custom card as needed to complete the processing requirement. Movement of data into (or out of) the VME/VSB environment for reasons of status, control, or data exchange/processing is possible at every card stage.

The general Macintosh based telemetry system [5], referred to as the Transportable Telemetry Workstation or TTW (see Figure 5), consists of:

1. a custom telemetry backplane bus;
2. a telemetry system software environment including user interface;
3. a Macintosh II/IIx host computer;
4. a MC68020 CPU NuBus card (real-time system controller);
5. a NuBus custom telemetry Synchronizer Card (same GSFC VLSI design as VME);
and
6. Additional cards for such functions as Reed/Solomon decoding, Packet Processing, Sorting, and Multiplexing of telemetry data.

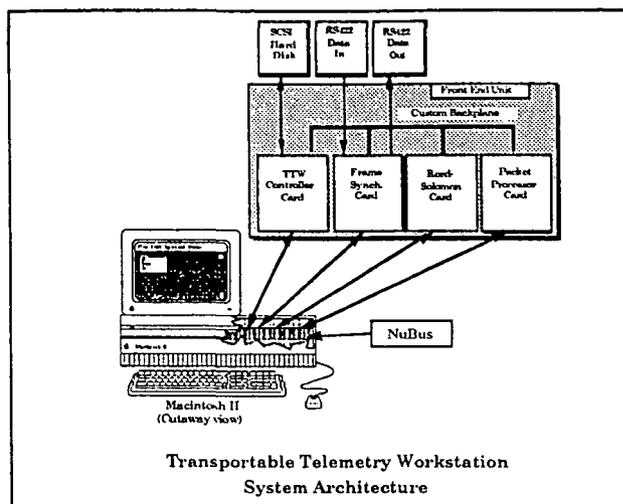


Figure 5: Macintosh Platform

Many of the general comments relating to the VME platform apply also to the TTW system.

The specific hardware (e.g. synchronization, Reed/Solomon decoding, packet processing etc.) associated with a particular functionality is defined within the Computer Aided Engineering (CAE) environment used at GSFC to design, develop, and test these components. The porting of this functionality to yet another platform such as the IBM PS/2 or RS6000 (Micro Channel) is similarly achievable.

4 Hardware Components

All hardware components support NASA's requirement to acquire, process, and distribute space telemetry data. These components include some 12 VLSI Application Specific Integrated Circuits (ASICs) and more than 14 card level components supporting some 11 different projects. These components were developed in-house at GSFC utilizing modern Computer Aided Engineering (CAE) tools. These tools also provide such capability as state-of-the-art Printed Circuit Board (PCB) place and route. The DSTD also supports an in-house fabrication capability including Surface Mount Technology (SMT) to meet limited production of system components. Above the card level, the actual performance level of a particular component (chip, card, or subsystem) is transparent. This implies that the look and feel of the user application and development environments is largely unchanged by bit rate performance requirements.

5 VLSI Components

The VLSI components provide the backbone of the functional component approach and define the actual potential of this approach to provide high performance, low cost telemetry systems. These components currently range in density from 4,400 to 15,000 gates and in-

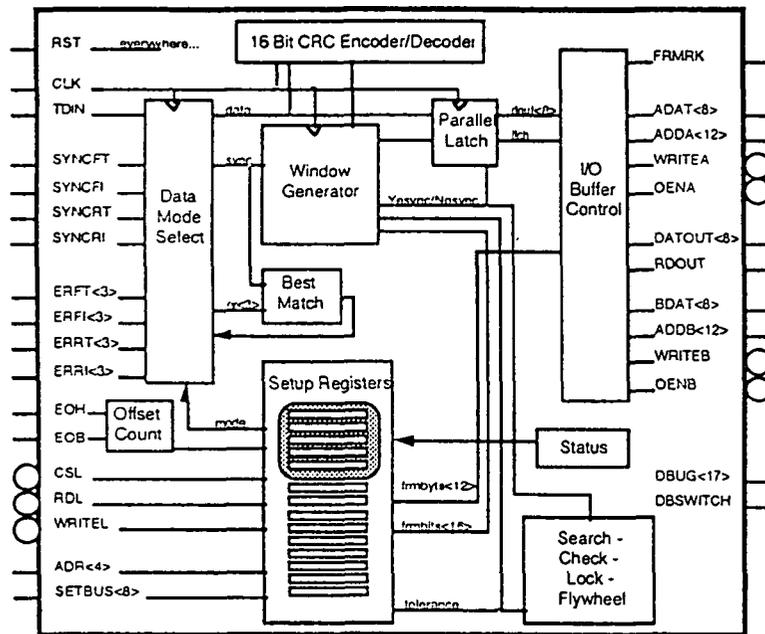


Figure 6: Telemetry frame synchronizer chip II block diagram

clude CMOS, ECL, and GaAs semi-custom arrays and a few full custom chips. To enhance each component's flexibility and reusability (in other designs), all components include such features as a microprocessor interface, a comprehensive set of internal read/write registers, hardware/software resets, and external mode control pins. Figure 6 show the component block diagram of the Telemetry Frame Sync II chip (TFS II). Figure 7 shows the associated register model used to program the TFS II.

The following are typical VLSI components currently used to support general system development:

- **Telemetry Frame Sync Chip (CMOS)**

- Programmable search, check, lock, and flywheel strategy.
- Synchronization of frames up to 32 kbits in length.
- Inversion correction, reversal correction, and slip correction.
- Double buffering of data.
- 16 bit CRC check.
- Microprocessor Controlled.
- 4,400 gate - 2 micron CMOS.
- Operation up to 20 Mbits/sec.

• Telemetry Correlator Chip (CMOS)

- Correlation to any synchronization pattern up to 32 bits in length.
- True and inverted sync indications.
- Programmable error tolerance for Search, Check, and Lock modes.
- Independent 22 bit CRC encoder/decoder.
- Microprocessor Controlled.
- 4,400 gate - 2 micron CMOS.
- Operation up to 20 Mbits/sec.

• Telemetry Frame Sync Chip (GaAs) ¹

- Programmable search, check, lock, and flywheel strategy.
- Synchronization of frames up to 32 kbits in length.
- Inversion correction, reversal correction, and slip correction.
- Real Time Quality Trailer generation.
- Double buffering of data.
- Programmable CRC check.
- Microprocessor Controlled.
- 15,000 gate - GaAs.
- Operation up to 300 Mbits/sec.

• NASCOM Block Processor chip

- Extraction of telemetry data from 4800 bit NASCOM block.
- Storage of entire 144 bit header.
- CRC and sequence check.
- Double buffering for both telemetry and non-telemetry data.
- Timing signals for end of header and end of block.
- Microprocessor Controlled.
- 4,400 gate - 2 micron CMOS.
- Operation up to 20 Mbits/sec.

Other key VLSI components include a NASA - 36 time decoder (CMOS), Random Access Memory Controller (CMOS), Support Chip for MC68020/30 CPU (CMOS), Correlator Chip (ECL), Test Pattern Generator chip (CMOS & GaAs), and the Tri-buffer Controller Chip (CMOS).

¹Note: prototypes of this part expected by 10/90.

Over the next two years, the second generation of VLSI components will be designed, developed, and tested. With new array gate counts up to 200,000 for CMOS and 100,000 for GaAs, these components promise the ultimate in integration of functionality, a single chip solution for much of the combinational logic now used on *all* functional component card subsystems. In addition, RAD hard, JAN 38510 gate arrays in the 50,000 plus densities could provide 'ready for flight' chip sets meeting much of NASA's up and down link functional and performance requirements.

6 Board Level Components - The Synchronizer Card

The VME Synchronizer Card is one of 14 card level components designed at GSFC and is an example of the potential power and flexibility of this approach. A general functional block diagram of this card is shown in Figure 8. This 9U VME card consists of a commercial single board computer referred to as the Synchronizer Card Channel Controller (SCCC) and a custom logic card connected via a side connector to the SCCC. The SCCC is a dedicated processor (68020/30 class) which provides setup, self-test, hardware diagnostics, debug, and control over the custom logic card. Extensive software to control the card and provide complete status information has been written and integrated on the SCCC.

The Synchronizer card develops most of its performance from four VLSI chip set s. While operational, these custom chip sets provides the hardware functions necessary to perform high speed NASCOM block processing, telemetry frame synchronization, real time frame trailer appendage, and cumulative quality generation. It can also perform data simulation for self-checking purposes.

The NASCOM Block Processor Chip (NBPCS) Set can be enabled to accept data from either the Data Simulation Chip Set or from the RS422 interface. Data formatted in 4800 bit NASCOM blocks is synchronized to the programmed sync pattern. Two output paths from this subsystem exist. For both paths the header (entire 144 bits) of each NASCOM Block can be read by the SCCC. If data is determined to be telemetry data, the NASCOM Block header and trailer is stripped away and a serial stream of only telemetry bits (fill bits ignored) is output to the Telemetry Frame Synchronizer Chip Set (TFSCS). If the block is determined to be non-telemetry (i.e. command block) the entire block can be transported off the Synchronizer Card. Additionally, the NBPCS performs quality checks on each block received including cyclic redundancy code (CRC) and sequence checks.

The TFSCS can be programmed to select data from either the RS422 interface, the DSCS, or the NBPCS. A complete synchronization strategy can be programmed for the selected data stream. Complete search, check, lock strategies, and slip windowing is provided. Additionally, the TFSCS has the ability to correct inverted or reversed frames. A status word for each frame is reported back to the SCCC. This status word contains sync mode, sync errors, slip indication, CRC errors (if applicable), and data type (forward, reverse, inverted, or true) of frame received. The TFSCS provides complete double buffering of frames and automatic output with framing control signals. Output control strobes allow for automatic status counting.

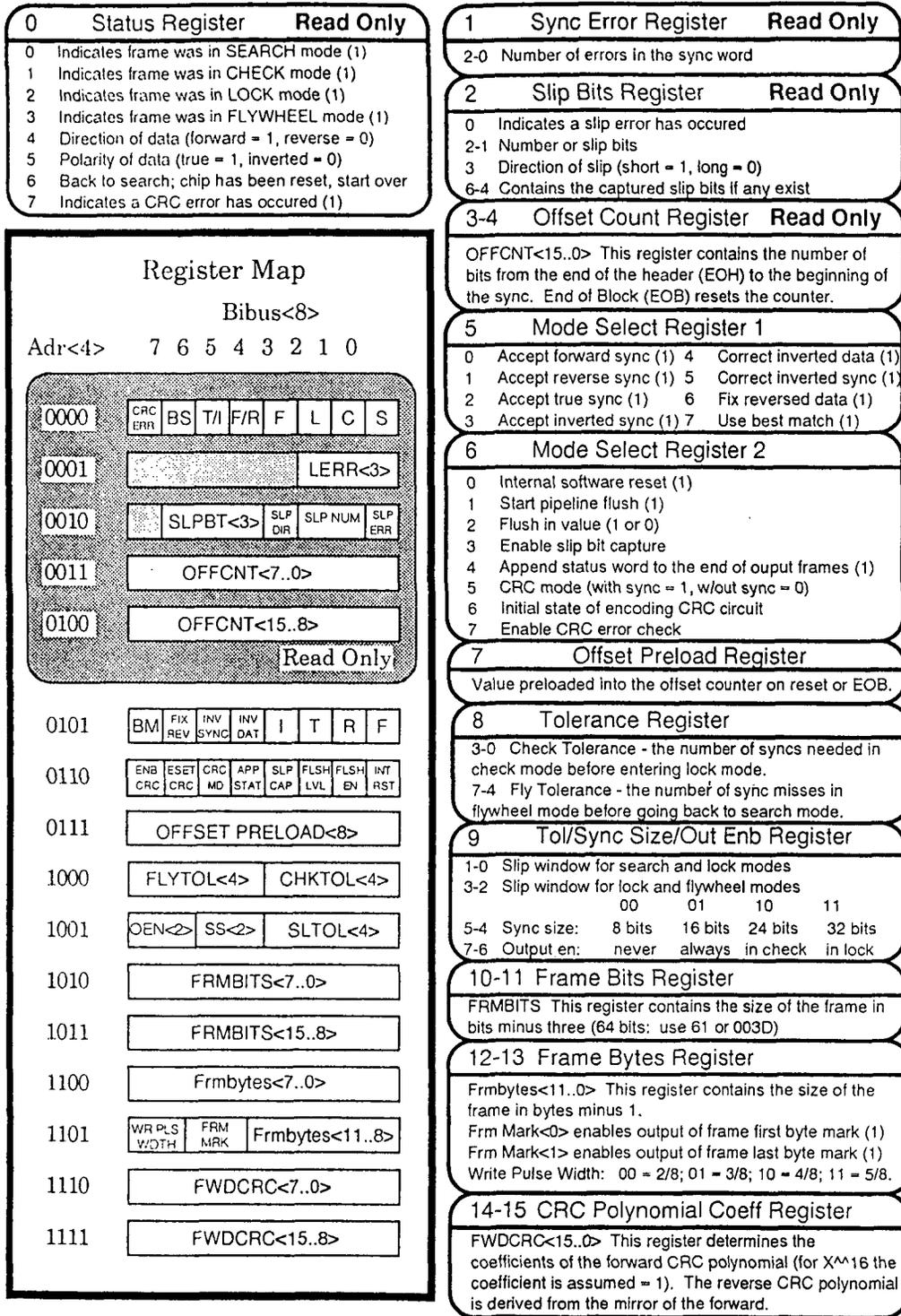


Figure 7: Telemetry frame synchronizer Chip II register model

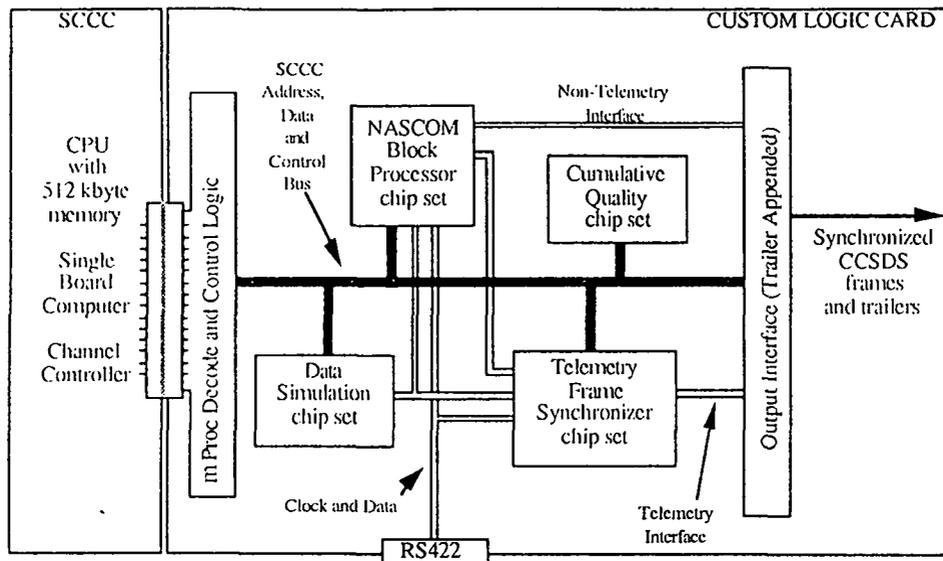


Figure 8: Telemetry frame synchronizer card block diagram

The Output Interface can be programmed to select either the telemetry or non-telemetry data path. Quality data generated by the SCCC is appended to the selected output path. The output interface then controls the transfer of this annotated data to the next processing system.

The Cumulative Quality Chip Set (CQCS) accepts control strobes from the NBPCS, the TFSCS, and from the SCCC and accumulates 32 status counts for up to 24 million events. These counts are read by the SCCC periodically and formatted into a complete status block. The status block can be formatted on a display terminal connected to the SCCC or communicated to a higher level system controller.

The Data Simulation Chip Set (DSCS) provides complete self-testing of all functions on the Synchronizer Card with high speed (up to 20 Mbps) simulation data. Once set up for output, the DSCS can provide independent output of a serial telemetry data stream. This stream can be programmed to provide any type of data (forward, inverted, reverse, or true) in any format for a known number of repetitions. By comparing the status results for the test run against known correct results, the SCCC can determine if the card is functioning correctly before activating it for operational data.

7 Software Components

The design and development of highly functional and flexible telemetry data systems requires the use and application of state-of-the-art real-time software techniques and approaches that are tightly coupled with the high performance VLSI based hardware systems [1,6]. The automated data driven operation of NASA's next generation telemetry data handling systems will require that standard system functions be virtually turnkey in every aspect of their operation. These system characteristics for the VME platform

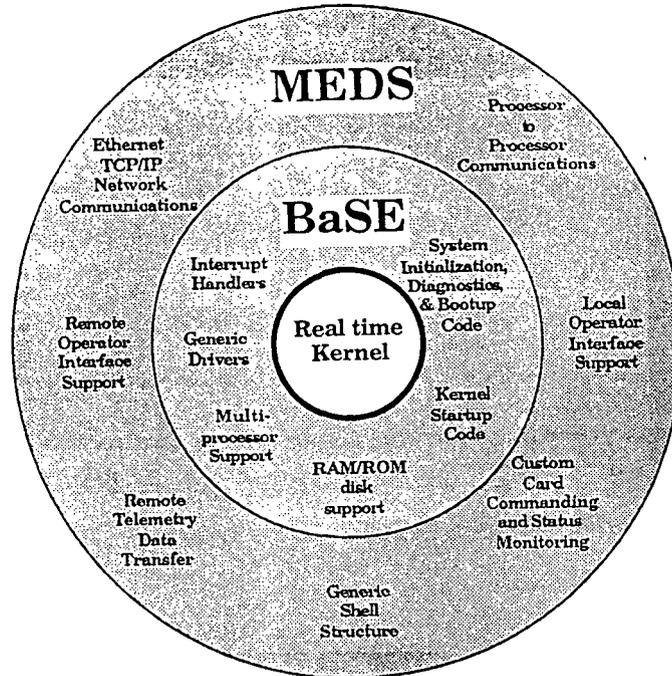


Figure 9: Software Environment

are provided by a commercial high performance real-time Operating System, and by three software systems environments referred to as the Base System Environment (BaSE), Modular Environment for Data Systems (MEDS), and the Network Environment for Telemetry Systems (NETS) (see Figure 9).

The BaSE allows system interactions between various manufacturer's cards to be prototyped, explored, and tested before being placed into operational use. Using the MEDS software package, application specific real-time code has a strong modular foundation that begins with a generic multiprocessing shell and supports the basic software functions needed by all multiprocessor based telemetry data systems. While still in the conceptual stage of development at GSFC, NETS is intended to ease the design and development effort required to integrate various telemetry data system nodes (e.g. VME capture system, workstation data processor, workstation host center controller etc.) using wide and local area networks.

Current development work with the Macintosh II NuBus system will also include the basic 'look and feel' of the original VME Base and MEDS environments.

8 BaSE Software Environment

NASA's next generation telemetry systems must provide a fairly simple and fast path to future enhancements. For this reason, the electronics hardware is based around widely supported open bus systems such as VME and NuBus. Likewise, a versatile software environment that can support the flexibility of these systems is also required. The telemetry system environment must contain enough intelligence to automatically configure the sys-

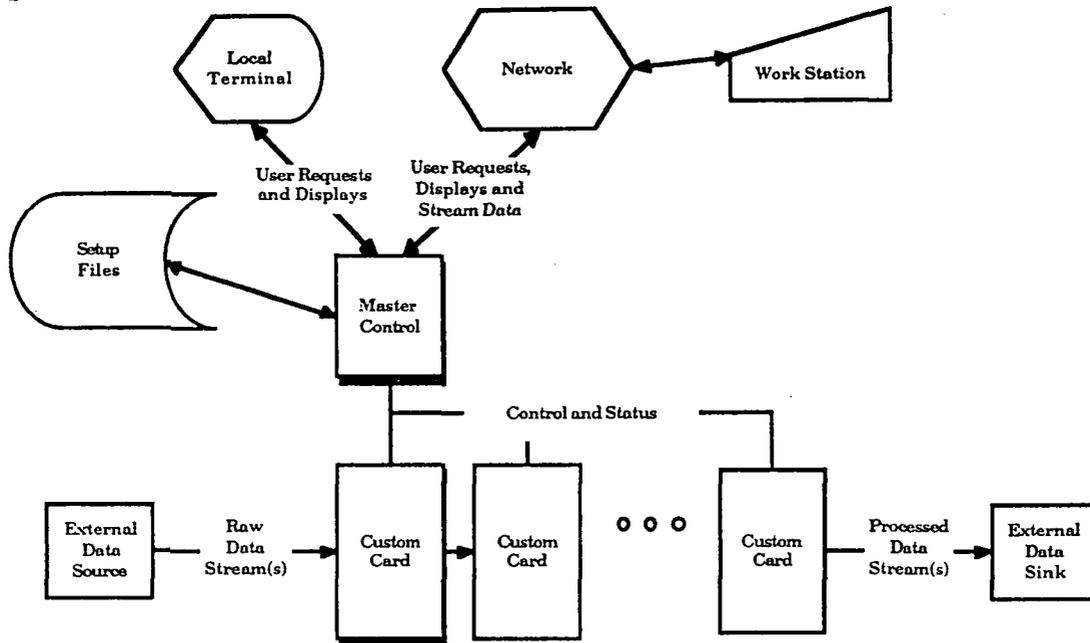


Figure 10: MEDS General Flow Diagram

tem based on a (often) changing hardware environment. It is the goal of BaSE to provide this type of functionality for the VME system and to provide the 'basic' interface at the bus hardware/software interface level. Through the use of BaSE, the systems developer can easily apply (or port) any of the hundreds of commercial cards or card systems to his particular telemetry system configuration. Once incorporated into the functional components BaSE environment, this new card(s) is compatible with and potentially part of any functional components system. BaSE allows the designer to pick and choose those VMEbus based products that best fit the application at hand and it ensures that products from different manufacturers can be used together in a *plug and play* fashion. BaSE allows seamless integration of NASA's custom VMEbus based telemetry processing cards into the operational environment and provides a single environment that is used for all phases of system development. BaSE is used from initial hardware test and checkout, to system software development, to final operational system deployment. The goal of the BaSE environment is to provide a cost effective telemetry platform environment that is generic in both hardware and software, can be used for both development and operations, and can be quickly and easily tailored to meet changing system needs.

9 MEDS

The systems based on the functional components approach all have a similar pipelined, multiprocessor, and dual bus hardware architecture, as a platform on which to build application specific hardware. When designing software for such a system, there are many questions which need to be answered. What data and parameters will each processor need to accomplish its job? How will the processors communicate with each other? And with the operator? How can the total job be subdivided into tasks? On which processors will

they run? What data and parameters will each task need to accomplish its job? How will each task get data? The Modular Environment for Data Systems (MEDS) was developed to help answer these questions and give an application programmer a starting point for designing a system based on the standard hardware platform. While BaSE provides a generic commonality at the bus hardware/software interface level, MEDS provides the knowledge of a "space telemetry data system". This implies the knowledge that a particular card system, say a Reed/Solomon decoder card, is not just a set of command, status, and data registers mapped in a specific memory space on the VME bus. Indeed, MEDS provide system wide knowledge that this card is in fact a Reed/Solomon decoder card with specific data in specific location with specific rules for formatting, debugging, controlling, user interface etc.

The Modular Environment for Data Systems (MEDS) is designed as a general purpose software platform which is expanded and customized by application programmers to suit their particular requirements. It supports the basic software functions needed in all systems, namely, the ability to setup application specific hardware and software, process the telemetry data based on the setup style parameters, monitor the processing and supply network support for remote operator interface and data transfer. MEDS supplies an infrastructure to pass data between systems, processors and tasks as well as support for operator interface development. A complete system is built by adding custom code to the general purpose MEDS code. Therefore MEDS spares the application developer from the burden of creating an infrastructure for each new system and adds consistency in all system design, implementation and maintenance.

A MEDS based system unites and manages the standard multiple processor hardware platform. The processors are organized as a single master processor directing multiple subordinate application specific custom cards (Figure 10). The master processor is the single point of control within the system; it interfaces with the operator, on either a local terminal or a remote workstation. Using a set of operator defined setup files, the master processor will initialize the custom cards and monitor their processing on various status pages. Telemetry data may enter and exit the system via the remote interface as well. In any case, it is the pipeline of custom cards that actually process the telemetry data.

The MEDS software resides mainly on the master with cooperating software running on each custom card. The basic MEDS functions include:

- Setup system and subsystems for processing (e.g. setup VLSI chip registers).
- Control the application specific processing (e.g. enable, disable, reset a card).
- Monitor the system and subsystems (e.g. gather and display card processing status).
- Stream data transfer over network (e.g transfer telemetry data to/from a workstation).

10 NETS

The Network Environment for Telemetry Systems (NETS), although currently in the conceptual design phase, is foreseen as one of the basic software environments supporting the functional component concept. NETS will provide for the control and management of multiple telemetry processing systems in a data processing facility. Similar to MEDS in architecture, NETS will provide a set of standard functions for communicating, scheduling and configuring telemetry processing systems distributed in a facility through a shared commercial network such as Ethernet or FDDI.

11 Level Zero Processor Project

The Level Zero Processor (LZP) now under development at GSFC [7] is an excellent example of the real benefits of the VLSI functional components approach. A primary goal of NASA's space and ground network system is to make the telemetry data transportation transparent to the customer so that it seems like the experimenter's instrument is within his own facility. This requires a process to remove from delivered data products all artifacts and disturbances introduced during data transport. This type of processing is a key part of what is usually referred to as the Level Zero Processing. In addition to error decoding and correcting functions to eliminate bit errors, the LZP system restores the order of data for a given observation or collection period. Such a data set is called a datatake and is delivered to the customer as a LZP product. Restoration of a datatake requires one or all of the following four basic LZP functions to be performed:

1. reassembling user packets from Virtual Channel Data Units (VCDUs);
2. reversing "backward" playback data;
3. merging together real-time data and playback data with proper time order; and
4. deleting redundant data due to the overlap between real-time and playback data.

To perform these tasks a new processing algorithm and a new architecture for the LZP system utilizing VLSI technologies has been proposed by the DSTD at GSF C. The new LZP system is based on the functional components approach and utilizes the VMEbus with multiple microprocessors running concurrently. The telemetry data will be processed for datatakes by various microprocessors and custom VLSI controllers while flowing through a data pipeline. Disk farms will be used as mass storage to buffer data for up to three orbits. This LZP will use Consultative Committee for Space Data Systems (CCSDS) Recommendations for format standards. The LZP system will provide standard services for taking data either NASCOM blocked or in synchronous VCDU format into a time ordered data take for delivery to the experimenter or customer.

The system will operate in three non-exclusive operation modes: real-time, quick-look, and production processing. In the real-time processing mode, customer's packets are

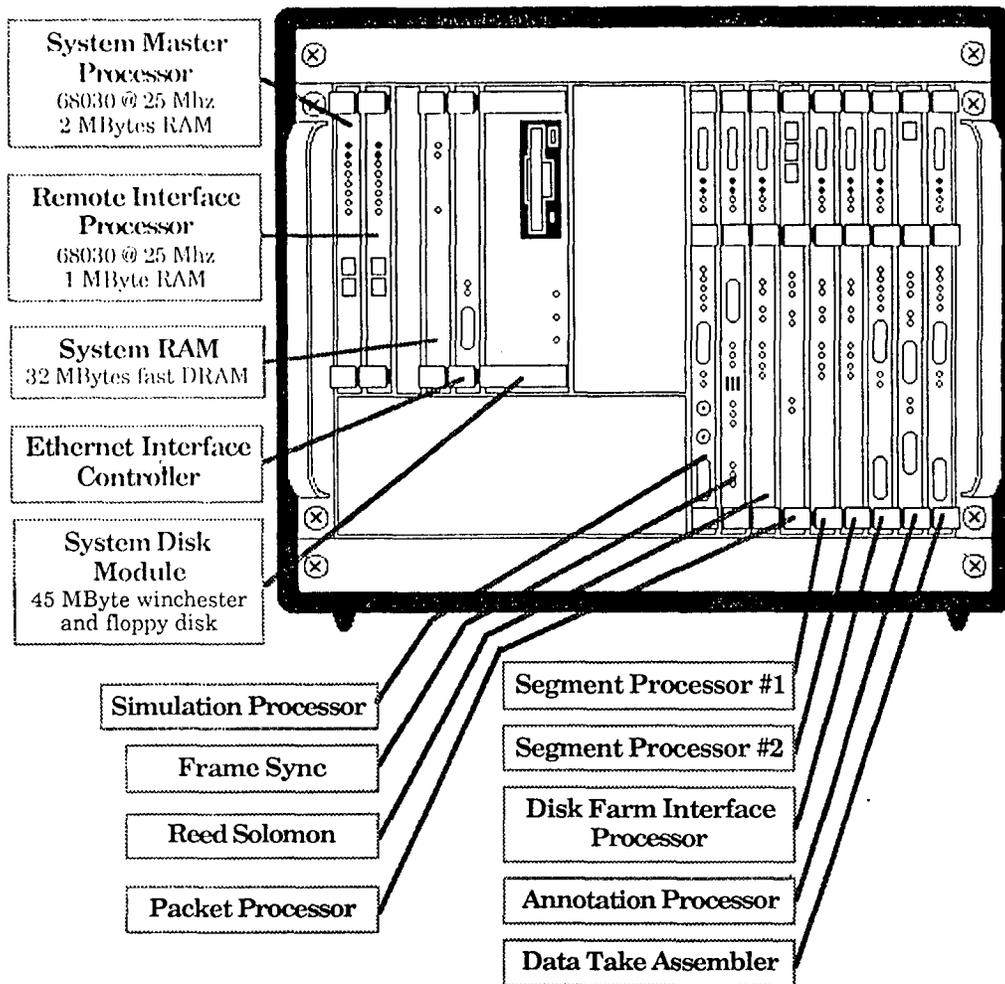


Figure 11: Level Zero Processor Physical Diagram

transmitted as soon as each packet is received and reassembled. Also, the data is retained for normal production. In the quick-look processing mode, a higher priority subset of the datatake will be made available to the customer. No redundancy deletion is performed. Again, the data is retained for normal production. The final and most important mode is the production processing mode. In this mode, data is processed and grouped into datatakes specified by the customer through scheduling tables. The completed datatake could be available to the customers within 90 minutes after receiving the last source packet from the datatake. (Additional factors in the actual throughput of the system include the time required to completely dump on-board recorder data and the performance achievable in the output product management and distribution.)

The LZP system can perform annotation of packets and datatakes, including data quality and accounting functions. Moreover, the catalog files for system operation, quality and production are maintained.

The following assumptions have been made for the processing environment in which the proposed LZP system will operate:

- All data complies with the Consultative Committee for Space Data Systems (CCSDS) Recommendations [2]. This implies that packetized data of multiple sources are transferred through multiple virtual channels.
- Maximum data rate of any virtual channel is 150 Mbps and total data from one virtual channel over one orbit does not exceed 15 Gbytes.
- Orbit time of the platforms is 90 minutes, 2/3 of which being daylight and 1/3 of which darkness and all data from one orbit will be received by the end of following orbit.
- Data packets bearing the same source ID have fixed length and the average packet size is 1 Kbytes, or 8 Kbits.
- The number of real-time and playback data segments is less than 1000 for each source per orbit.

Level Zero Processing is a two stage process. In the first stage, a serial data stream is assembled into user packets and then stored in a mass storage buffer. The bit ordering of playback data within each packet is corrected. In the second stage, packets are sorted according to their source IDs and time sequence, and grouped together to form datatakes.

In order to achieve high speed and low cost, the functional VLSI component approach was used in designing the LZP system. Figure 4 depicts the general system functional block diagram for the LZP system (mass storage system not shown). Implemented with commercial microcomputer modules and custom telemetry data processing cards, the LZP system consists of three subsystems: the front-end VLSI data capture system, the Datatake Processor, and the Mass Storage Subsystem. Figure 11 shows the actual physical diagram of the telemetry data system portion of the LZP without the required mass storage components.

12 Conclusion

To fully develop the potential of future space activities, NASA's telemetry data systems must do more than simply meet specific technical requirements. They must provide for reliable, low cost, and modular systems which NASA and its user community can tailor in size and performance to particular needs. These systems must allow for growth and expansion in the years to come. Also, with the push toward an automated data driven operation of NASA's next generation telemetry data handling systems, it is important that standard system functional components be virtually turnkey in operation. Even though these systems are a tightly integrated mix of hardware and software elements, they are but single elements in a large, highly complex NASCOM telemetry data handling system. The functional component approach was designed to meet these needs.

References

- [1] Hand, S. and Sabia, S., "Functional Component Approach to Telemetry Data Capture Systems," VOL XXIV, *Proceedings of the International Telemetry Conference*, Las Vegas, Nevada, October 1988.
- [2] Shi, J., Grebowsky, G., Horner, W., "Prototype Architecture for a VLSI Level Zero Processing System," VOL XXV, *Proceedings of the International Telemetry Conference*, San Diego, CA, October 1989.

Nomenclature

ASIC	Application Specific Integrated Circuit
BaSE	Base System Environment
CAE	Computer Aided Engineering
CCITT	Consultative Committee for International Telegraph and Telephone
CCSDS	Consultative Committee for Space Data Systems
CMOS	Complimentary Metal Oxide Semiconductor
CQCS	Cumulative Quality Chip Set
CRC	Cyclic Redundancy Check
CVCDU	Coded Virtual Channel Data Unit
DSCS	Data Simulation Chip Set
DSTD	Data Systems Technology Division
ECL	Emitter Coupled Logic
FDDI	Fiber Data Distribution Interface
GaAs	Gallium Arsenide
GSFC	Goddard Space Flight Center
ISO	International Organization for Standards
LZP	Level Zero Processor
MEDS	Modular Environment for Data Systems
NASCOM	NASA Communications
NBPCS	NASCOM Block Processor Chip Set
NETS	Network Environment for Telemetry Systems
PCB	Printed Circuit Board
SCCC	Synchronizer Card Channel Controller
SMT	Surface Mount Technology
TFSCS	Telemetry Frame Synchronization Chip Set
TFS II	Telemetry Frame Synchronization Chip II
VCDU	Virtual Channel Data Unit
VLSI	Very Large Scale Integration

Session 7 Systems

Chairman: John Purviance

Data Storage Technology Comparisons

Dr. Romney R. Katti
Jet Propulsion Laboratory
4800 Oak Grove Drive
California Institute of Technology
Pasadena, California 91109

1 Introduction

The role of data storage [1] and data storage technology is an integral, though conceptually often underestimated, portion of data processing technology. Data storage is important in the mass storage mode in which generated data is buffered for later use. But data storage technology is also important in the data flow mode when data are manipulated and hence required to flow between databases, datasets and processors. This latter mode is commonly associated with memory hierarchies which support computation.

VLSI devices can reasonably be defined as electronic circuit devices such as channel and control electronics as well as highly integrated, solid-state devices that are fabricated using thin film deposition technology. VLSI devices in both capacities play an important role in data storage technology. In addition to random access memories (RAM), read-only memories (ROM), and other silicon-based variations such as PROMs, EPROMs, and EEPROMs, integrated devices find their way into a variety of memory technologies which offer significant performance advantages. These memory technologies include magnetic tape, magnetic disk, magneto-optic disk, and vertical Bloch line memory. In this paper, some comparison between selected technologies will be made to demonstrate why more than one memory technology exists today, based for example on access time and storage density at the active bit and system levels.

2 Role of Integrated Devices in Data Storage Technology

The memory element that is most commonly considered as a highly integrated memory is semiconductor RAM. This memory can be implemented in many ways, but generally is configured to consist of a memory element, such as a single capacitive transistor or a multi-transistor flip-flop, replicated in a two-dimensional matrix in which each element is accessed through a row and column address. Research and developmental investigations are looking at cell sizes which can achieve 256 Megabits on a single chip, but the currently available commercial chips typically provide 64 kilobits to 4 Megabits. Because of radiation immunity, reliability, power consumption, and other spaceflight requirements, chip

capacities which have recently been launched in spacecraft are in the 1 kilobit to 4 kilobit range. Efforts are underway to develop and qualify 64 kilobit chips for spaceflight.

Though the memory portion in RAM memory cells appears to offer high density, because each memory element requires access to various power, ground, blocking, precharge, column address, and row address lines, the effective, realizable storage density at the system level is degraded. Overall data transfer rates can also be poor for certain applications because the data rate equals the access rate and a single bit is accessed per unity cycle time.

To improve density and data transfer rate, memory elements which reduce the access and support overhead per stored bit are considered. Block access memories such as magnetic disk systems, magneto-optic disk systems, and vertical Bloch line (VBL) memory are intended to fill this void. For example, disk systems use a single read/write transducer to access sector of data arranged in tracks, and VBL memory propagates magnetic information within a crystal to integrated sensors. Storage density is increased because data bits are placed together as close as is physically possible without adding area around each bit for reading and writing each bit. The average data transfer rate can be increased for contiguous records by accepting an access time delay, usually around 100 *musec* for magneto-optic disks, 10 to 40 *musec* for magnetic disks, and 100 *musec* to 1 *musec* for VBL memory, but achieving high data rate once the access is complete.

The block access systems just described achieve high a real density, but their structures do not maximize volumetric storage density. To achieve greater volumetric storage density, tape systems are used. Tape systems are mechanical systems which essentially wrap data bits into rolls made from ribbons of material that are made to be as compact as can reliably be manufactured. The increase in volumetric storage density, however, comes at the cost of increased access times, which are typically in the tens of seconds. Data transfer rates can be kept high if lengthy data streams at high data rate are recorded for lengths of time which exceed access times. In certain applications, the advantage of storage density may overwhelm the disadvantage of increased access times.

Examples of integrated devices are now provided. In magnetic disk drives, the recording head is an integrated magnetic device made using thin film magnetic material deposition. Such an inductive head is shown in Figure 1. The figure has been extracted from the book "Magnetic Recording" by C.D. Mee and E.D. Daniel [2]. The active magnetic material is commonly made from sputtered and/or plated permalloy (Ni_{0.8}Fe_{0.2}). VBL Memory is an integrated, solid-state, nonvolatile, radiation hard memory in which the active storage film is an epitaxially-grown garnet, which is a magnetic oxide. A layout of a VBL memory chip is shown in Figure 2.

3 Results and Discussion

Comparing widely differing technologies is typically a challenging enterprise. There are also many issues which need to be addressed in a data storage system for spaceflight, such as environmental issues, radiation, mass, volume, power, reliability, durability, cost,

etc. However, certain assumptions can be made which produce trends which are observed in actual systems. Persons interested in this subject are encouraged to make their own assumptions and conduct their own analysis. In this analysis, the technologies of RAM, magnetic disk, magneto-optic disk, VBL, and magnetic tape will be compared at various levels of advancement. A comparison will be made of a real and volumetric bit densities at the active memory element level, and this will be compared to a real and volumetric bit densities at the memory system level. The active memory element refers to the memory cell unit which includes active media, while the system memory element includes overhead including media substrates and read/write transducers.

Shown in Figure 3, for a range of technologies, is a graph indicating the surface area of a bit of data at the active bit level, the volume of a bit at the active bit level, and the volume of a bit at the system level. The numerical values assumed for the bits are summarized in Table 1. Several trends are apparent and are now discussed.

First, the volume of a bit at the system level, after accounting for example for media substrates and read/write transducers, is two to four orders of magnitude greater than the volume of a bit at the active media level. This indicates that the mechanical trappings in data storage systems consume a considerable amount of volume. Further, because of this, high a real density values do not always automatically yield volumetrically efficient systems. Second, at the system bit volume level, RAM systems tend to be volumetrically inefficient when compared to block access and serial access systems. Third, this trend correlates with the trend that volumetrically efficient systems tend to have longer access times. Fourth, while technological progress is also being made in block access and serial access systems. Thus, block and serial access systems are continuing to offer system performance advantages over RAM systems.

Data from existing systems can also be reviewed and compared to the observations made above. Shown in Figure 4 is a graph of access time vs. capacity. The trend observed here, in which access time increases as capacity increases, is consistent with the conclusion drawn above.

Other conclusions can be drawn from existing data storage systems. Figure 5 shows that as capacity increases, data rate increases. Since the time duration for which data needs to be recorded tends not to decrease, capacity must then increase as data rate increases. This trend is apparent in Figure 6, in which capacity per unit of data rate actually increases as capacity increases.

As the capacity of systems increases, the efficiency of data storage systems increases, in terms of capacity achieved per unit of power and unit of volume. These trends are shown in Figures 7 and 8, respectively. For reference and to facilitate comparisons, graphs of capacity as a function of power and volume are also provided, as shown in Figures 9 and 10.

Figure 11 shows the mass of data storage systems plotted against volume. It is observed that as mass increases, volume increases. This is interpreted as stating that mechanical and electronics issues are dominant in existing systems. Thus, improvements in future data storage systems are needed not only in data storage technology components, but also in the mechanical and electronic components.

The reliability of data storage systems is plotted against capacity in Figure 12. One observation indicates that tape systems achieve high capacity, but that the existing formats, involving fixed head and rotary head recorders, achieve high and low levels of durability respectively.

4 Conclusion

Comparison of data storage technology and systems have been made using postulated data storage values as well as actual system values. Several trends have been observed with the data, and three are now summarized. First, as storage density increases, average access time also increases. This indicates that present and upcoming technologies tend not to have the ability to satisfy both high capacity and rapid access to data. Second, while advances are being made in RAM technology, advances are also being made in the block access and serial access memories. Thus, for the foreseeable future, efficient data systems will consist of multiple memory technologies, and not just RAM. Third, storage density at the active bit level tends to be a thousand to a million times greater than storage density at the usable system level. This suggests that features such as packaging, and VLSI control and channel electronics, which include signal conditioning, coding, and EDAC/compression functions, are very important to data storage systems.

5 Acknowledgements

The work performed in this report was conducted at the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored under a contract by the National Aeronautics and Space Administration.

References

- [1] R.R. Katti and H.L. Stadler, "Survey of Data Storage Systems for Spaceflight Applications," Report to NASA/OAST, Civil Space Technology Initiative, JPL D-6924, Internal Report, May 31, 1989.
- [2] C.D. Mee and E.D. Daniel, eds. "Magnetic Recording, Volume I: Technology," McGraw-Hill, New York, 1987.

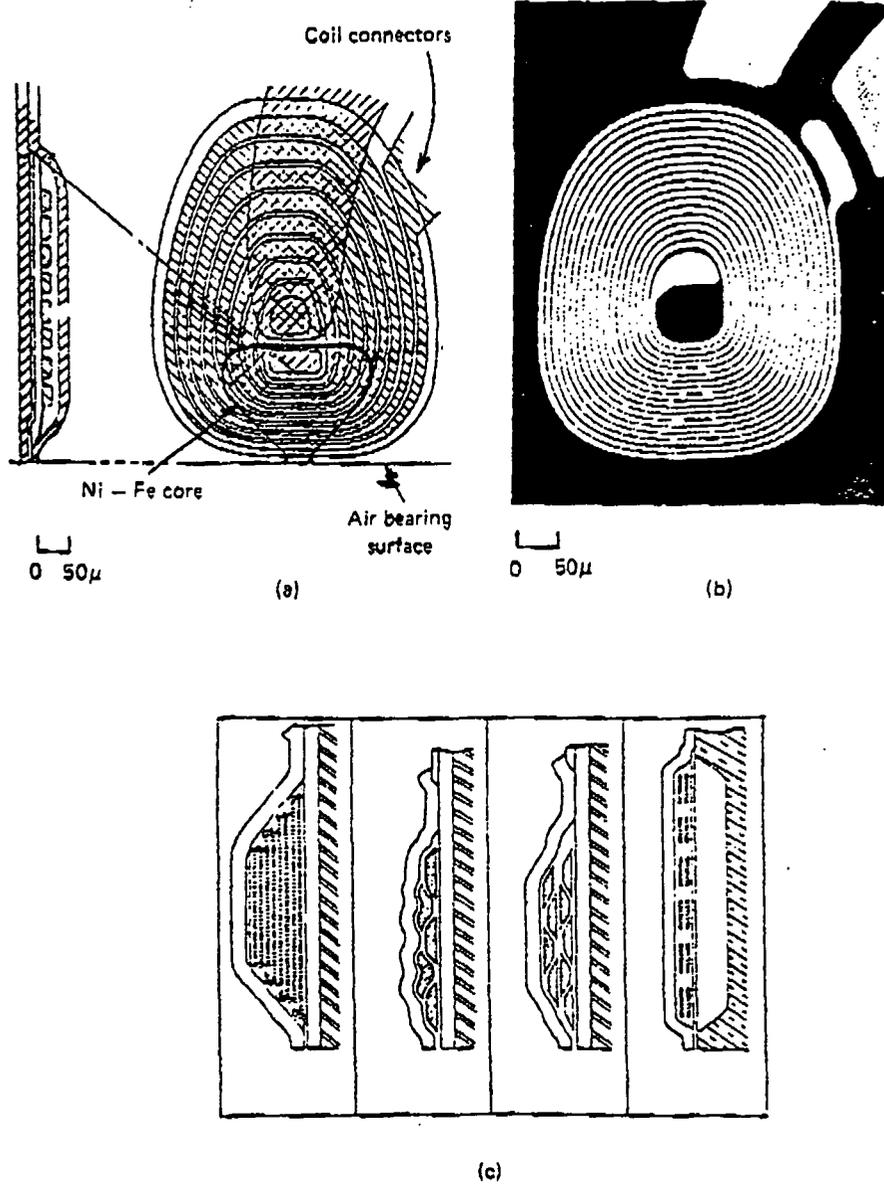


Figure 1: Design, Micrograph, and Cross-Sectional Views of Inductive, Thin Film Magnetic Recording Heads.

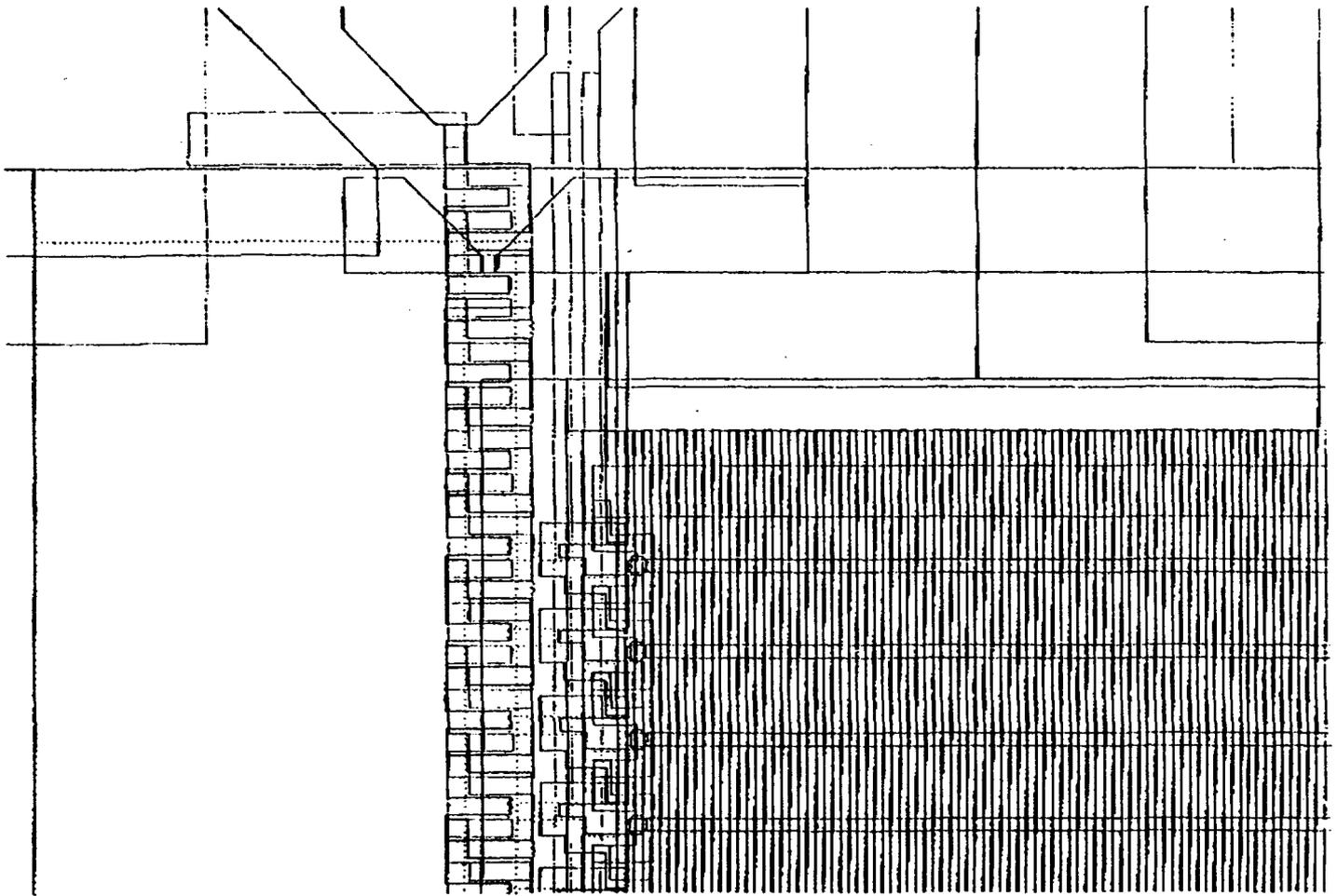


Figure 2: Layout of Vertical Bloch Line Memory.

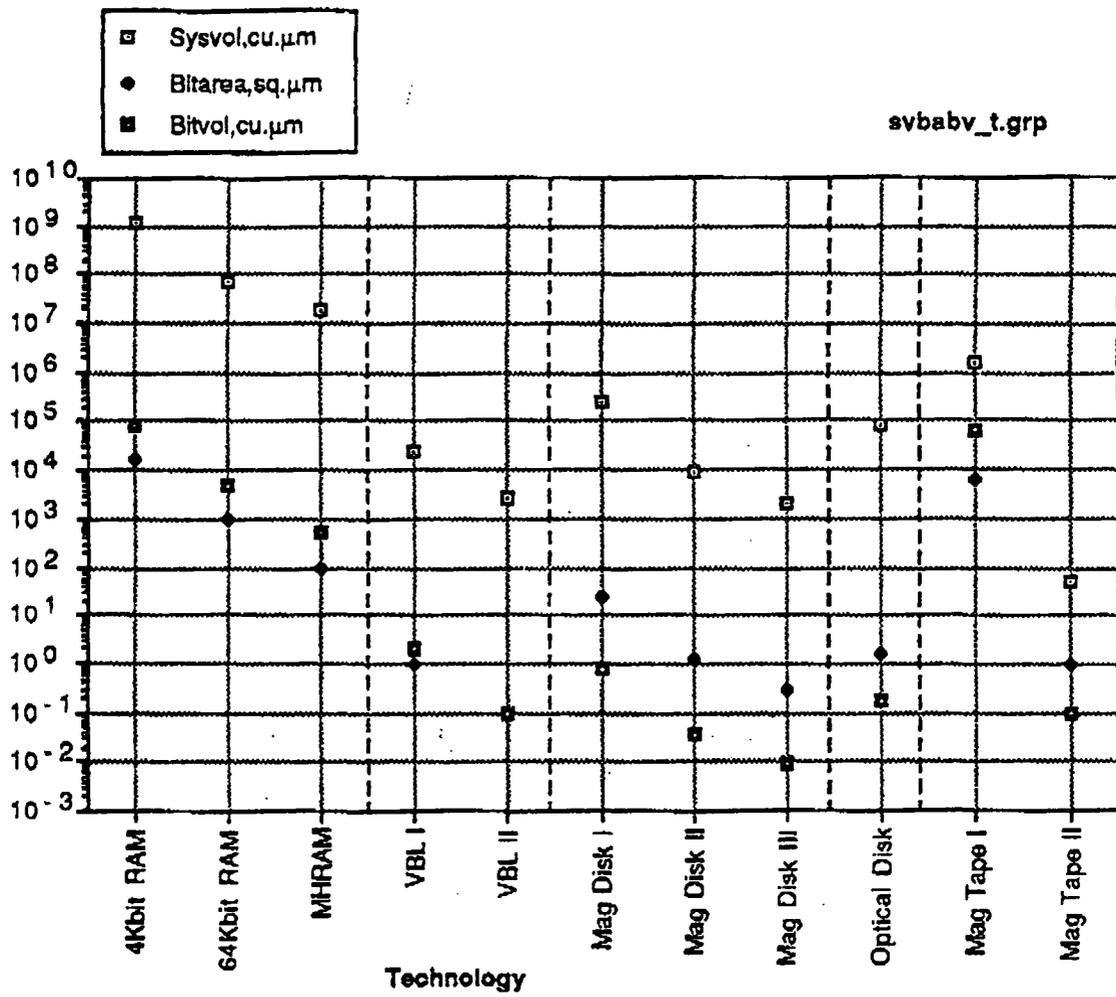
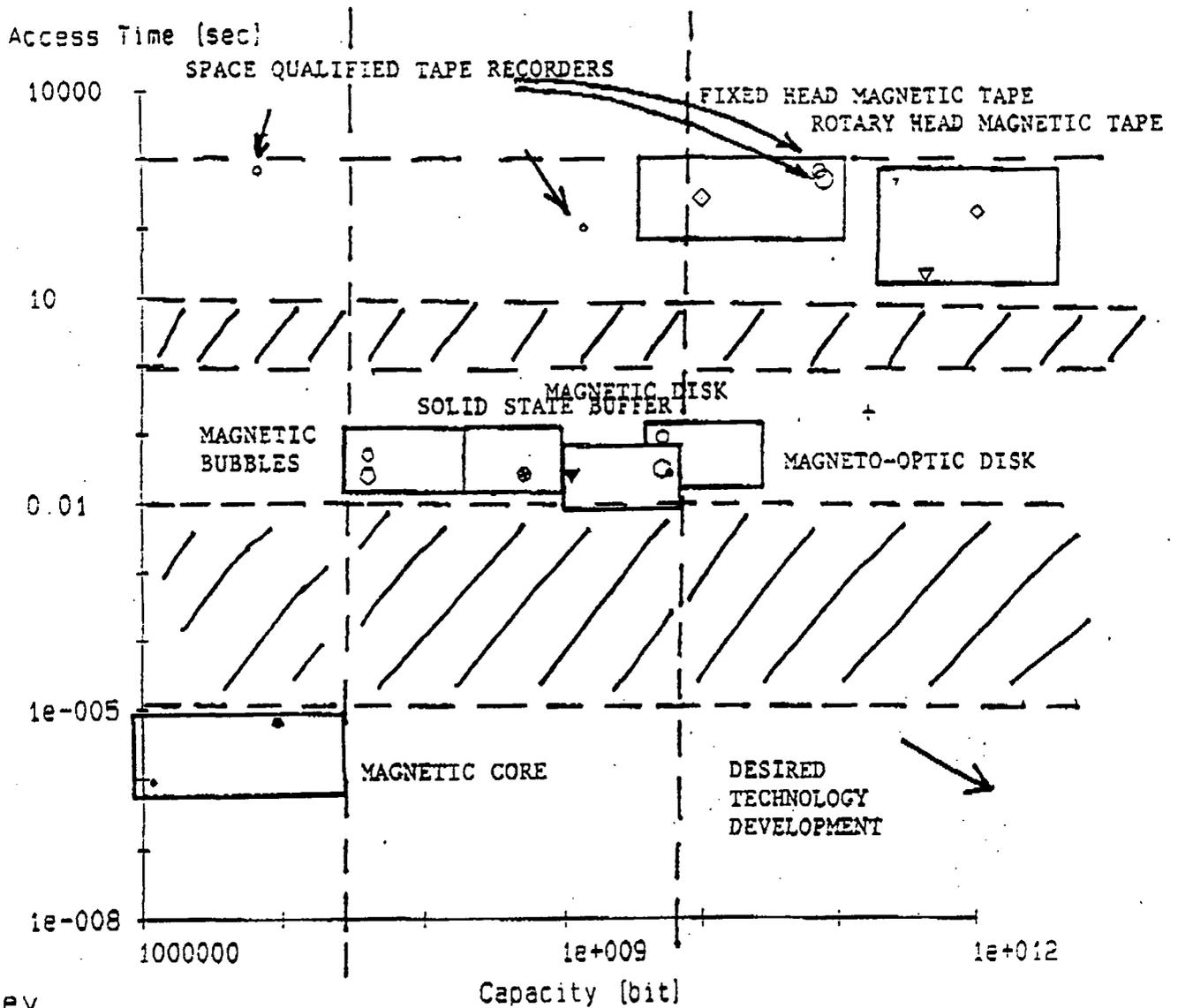


Figure 3: Graph of Bit Performance vs. Technology.

Technology	BitLength,um	Systlength,um	Bitwidth,um	Sysewidth,um	Blidepth,um	Sydepth,um	Bitarea,ca,um	Sysarea,sq,um	Bitvol,cu,um	Sysvol,cu,um
1 4Kbit RAM	128.400	343.630	126.490	343.630	3.000	10000.000	15999.720	118081.577	79998.601	1180815769.0
2 64Kbit RAM	31.800	85.910	31.600	85.910	3.000	10000.000	998.560	7380.529	4982.800	73805281.000
3 MHRAM	10.000	49.700	10.000	37.200	5.000	10000.000	100.000	1848.840	500.000	18488400.000
4 VBL I	1.000	1.000	1.000	1.000	2.000	25000.000	1.000	1.000	2.000	25000.000
5 VBL II	0.316	0.316	0.316	0.316	1.000	25000.000	0.100	0.100	0.100	2496.400
6 Mag Disk I	0.500	0.500	50.000	50.000	0.030	10000.000	25.000	25.000	0.750	250000.000
7 Mag Disk II	0.250	0.250	5.000	5.000	0.030	7000.000	1.250	1.250	0.037	8760.000
8 Mag Disk III	0.100	0.100	3.000	3.000	0.030	7000.000	0.300	0.300	0.009	2100.000
9 Optical Disk	1.000	1.000	1.700	1.700	0.100	60000.000	1.700	1.700	0.170	85000.000
10 Mag Tape I	4.000	4.000	1600.000	1600.000	10.000	250.000	6400.000	6400.000	64000.000	1600000.000
11 Mag Tape II	0.100	0.100	10.000	10.000	0.100	50.000	1.000	1.000	0.100	50.000

Table 1: Summary Table of Assumed Bit Dimensions.



Key

- | | |
|--|--------------------------------------|
| ○ DDS-5000EC (SPOT-1, 2) Tape Recorder | ○ 1990 Magnetic Bubble System |
| ○ DDS-6000EC (JERS-1) Tape Recorder | ○ DR650RE Magneto-Optic Disk System |
| ○ DDS-6000EC (Landsat) Tape Recorder | ○ Tahiti-I Magneto-Optic Disk System |
| • QDS-9000 Tape Recorder | + Spaceflight Optical Disk Recorder |
| ▽ PV6410 Rotary Tape Recorder | • Ruggedized Advanced Disk System |
| ▽ PV6210 Rotary Tape Recorder | • Militarized Advanced Disk System |
| • NASA STR-108 Tape Recorder | ▼ Mobile Mass Storage Disk System |
| ◇ ROR-300 Tape Recorder | • MESA-64RH Core Memory System |
| ◇ 20-MBS Tape Recorder | • MEGA-9 Core Memory System |
| ○ 1986 Magnetic Bubble System | ● SSR1-512B Solid State Recorder |

R. Katti JPL
Data Storage Survey

Figure 4: Graph of Access Time vs. Capacity for Existing Data Storage Systems.

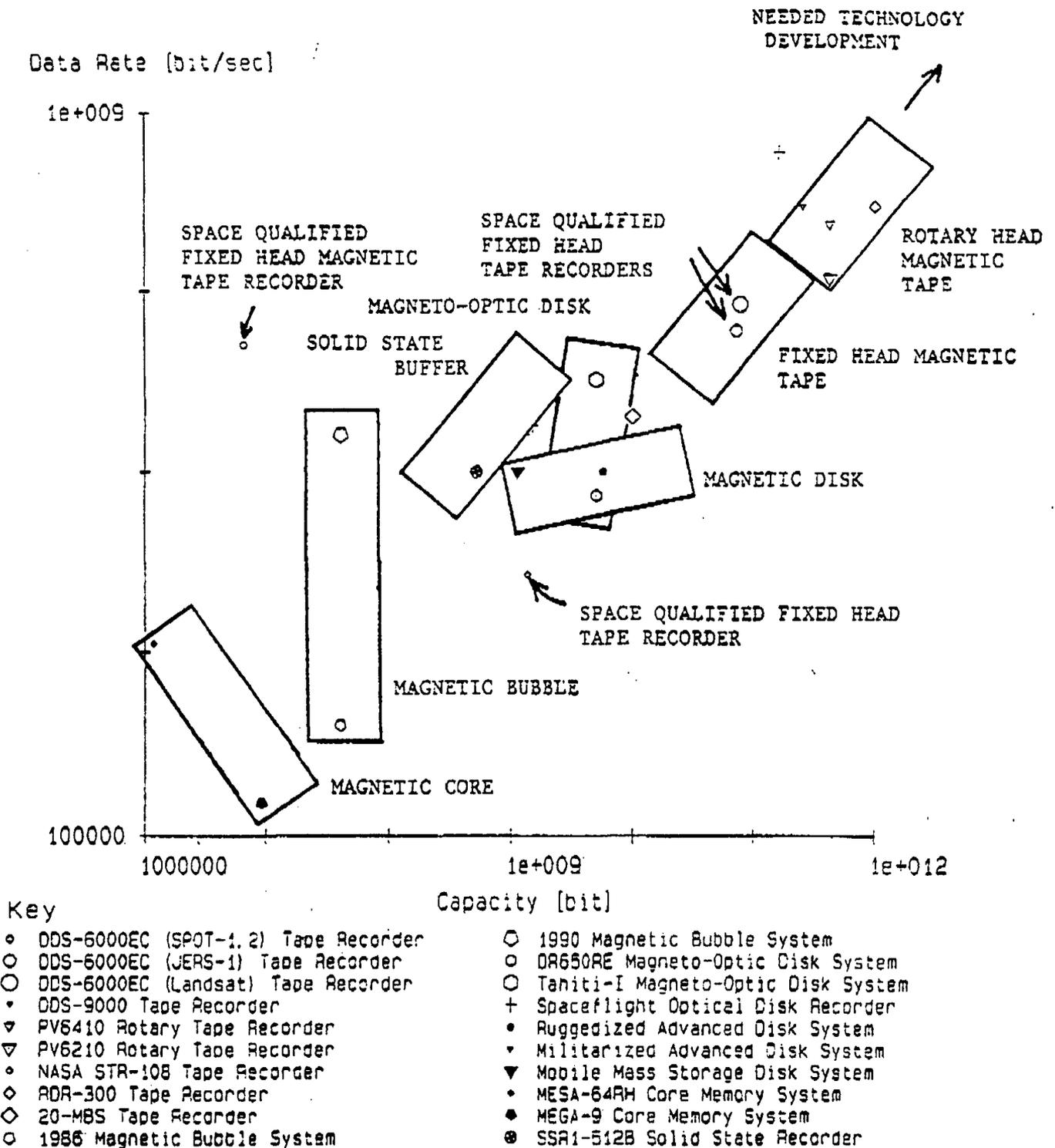
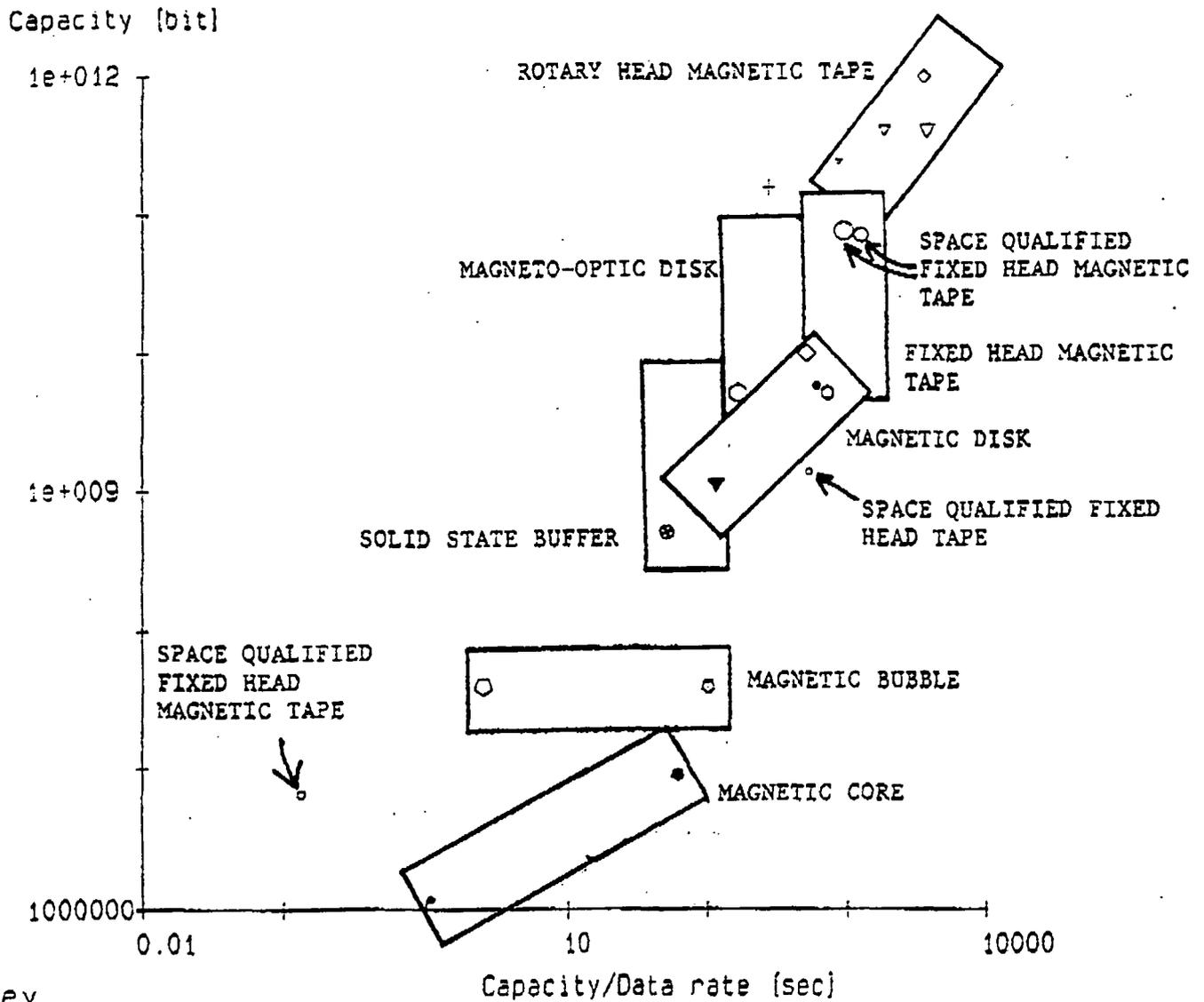


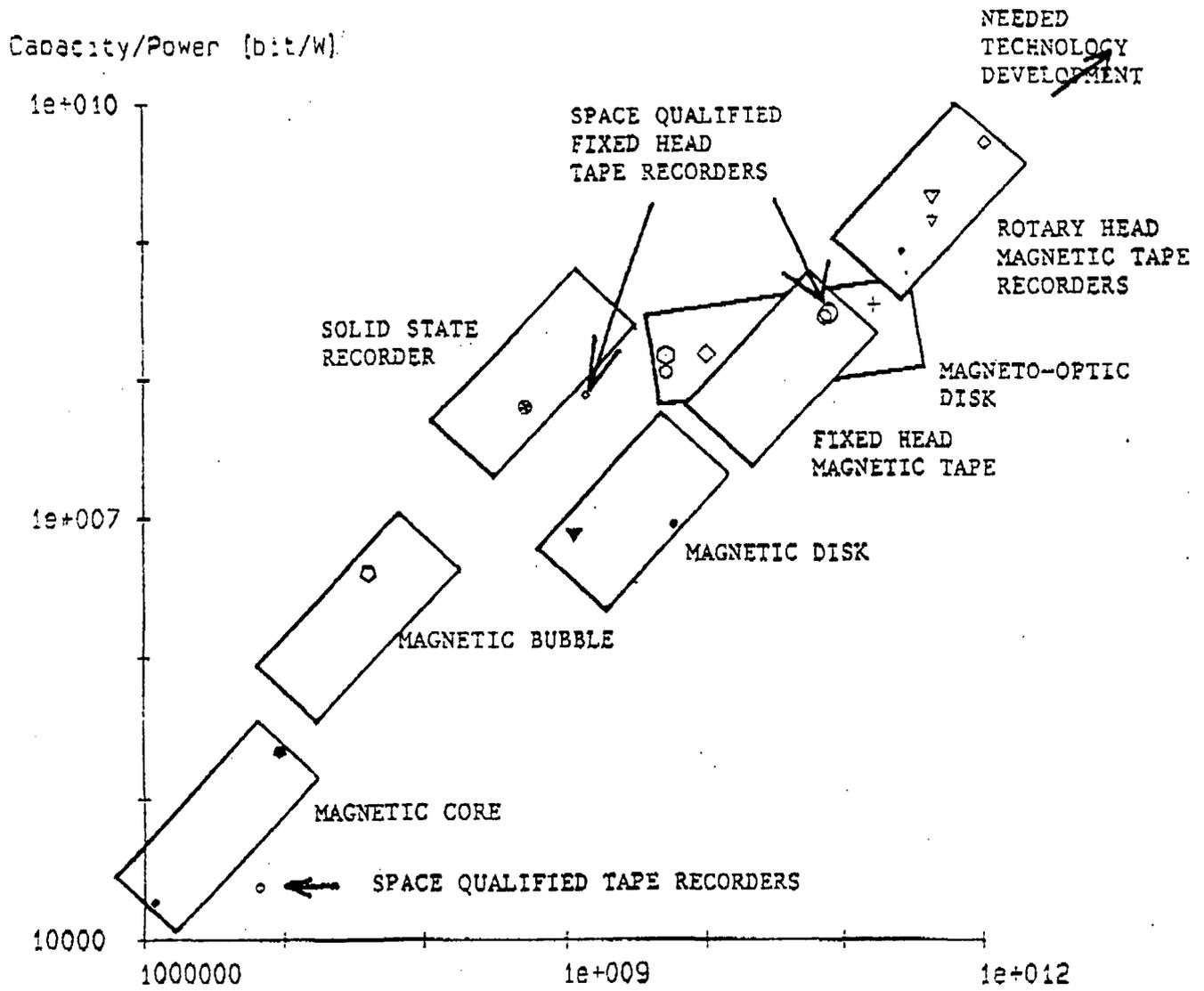
Figure 5: Graph of Data Rate vs. Capacity for Existing Data Storage Systems.



Key

- | | |
|--|--------------------------------------|
| ◦ DDS-6000EC (SPOT-1, 2) Tape Recorder | ⊙ 1990 Magnetic Bubble System |
| ○ DDS-5000EC (JERS-1) Tape Recorder | ⊙ DR650RE Magneto-Optic Disk System |
| ○ DDS-6000EC (Landsat) Tape Recorder | ○ Tahiti-I Magneto-Optic Disk System |
| • DDS-9000 Tape Recorder | + Spaceflight Optical Disk Recorder |
| ▽ PV6410 Rotary Tape Recorder | • Ruggedized Advanced Disk System |
| ▽ PV6210 Rotary Tape Recorder | • Militarized Advanced Disk System |
| ◦ NASA STR-108 Tape Recorder | ▼ Mobile Mass Storage Disk System |
| ◇ RDR-300 Tape Recorder | • MESA-64RH Core Memory System |
| ◇ 20-MBS Tape Recorder | • MEGA-9 Core Memory System |
| ○ 1986 Magnetic Bubble System | ● SSR1-512B Solid State Recorder |

Figure 6: Graph of Capacity vs. Capacity per Unit Data Rate for Existing Data Storage Systems.



- Key
- | | |
|--|--------------------------------------|
| ◊ DDS-6000EC (SPOT-1, 2) Tape Recorder | ⊙ 1990 Magnetic Bubble System |
| ○ DDS-5000EC (JERS-1) Tape Recorder | ⊙ DR650RE Magneto-Optic Disk System |
| ⊙ DDS-6000EC (Landsat) Tape Recorder | ⊙ Tahiti-I Magneto-Optic Disk System |
| ▼ DDS-9000 Tape Recorder | + Spaceflight Optical Disk Recorder |
| ▽ PV6410 Rotary Tape Recorder | • Ruggedized Advanced Disk System |
| ▽ PV6210 Rotary Tape Recorder | • Militarized Advanced Disk System |
| ◆ NASA STR-108 Tape Recorder | ▼ Mobile Mass Storage Disk System |
| ◆ RDR-300 Tape Recorder | • MESA-64RH Core Memory System |
| ◆ 20-MBS Tape Recorder | • MEGA-9 Core Memory System |
| ⊙ 1985 Magnetic Bubble System | ● SSR1-512B Solid State Recorder |

Figure 7: Graph of Capacity per Unit Power vs. Capacity for Existing Data Storage Systems.

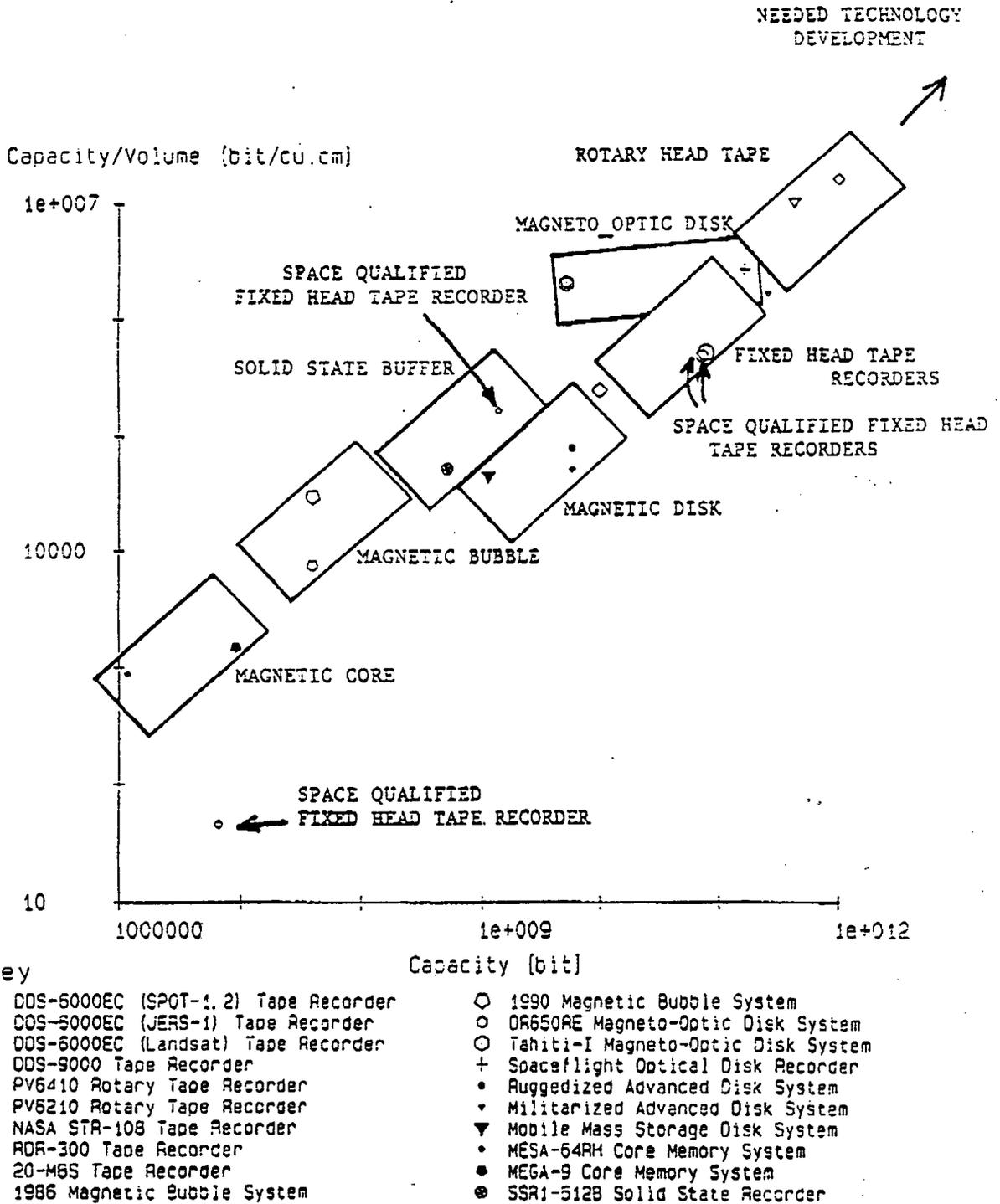


Figure 8: Graph of Capacity per Unit Volume vs. Capacity for Existing Data Storage Systems.

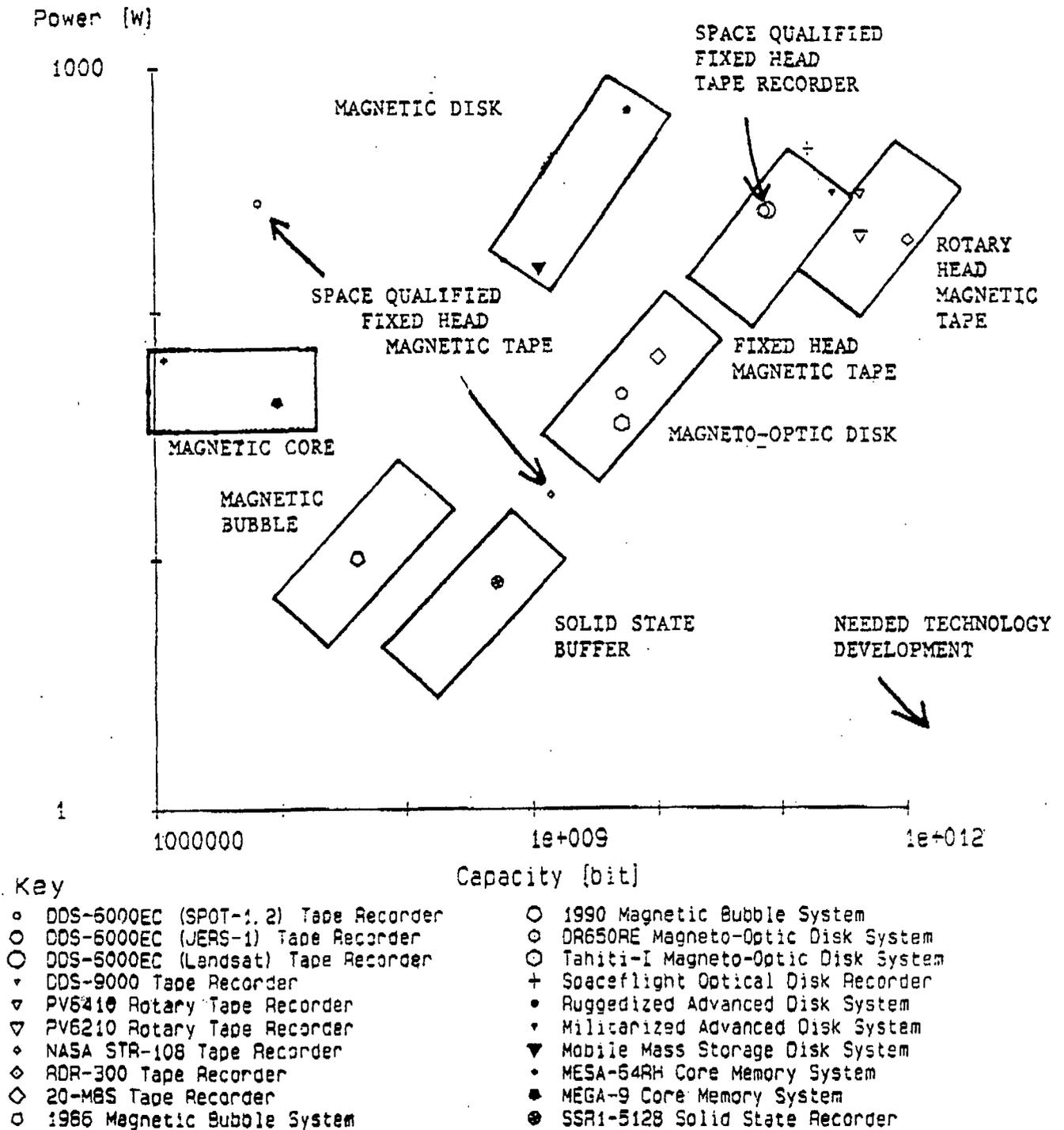
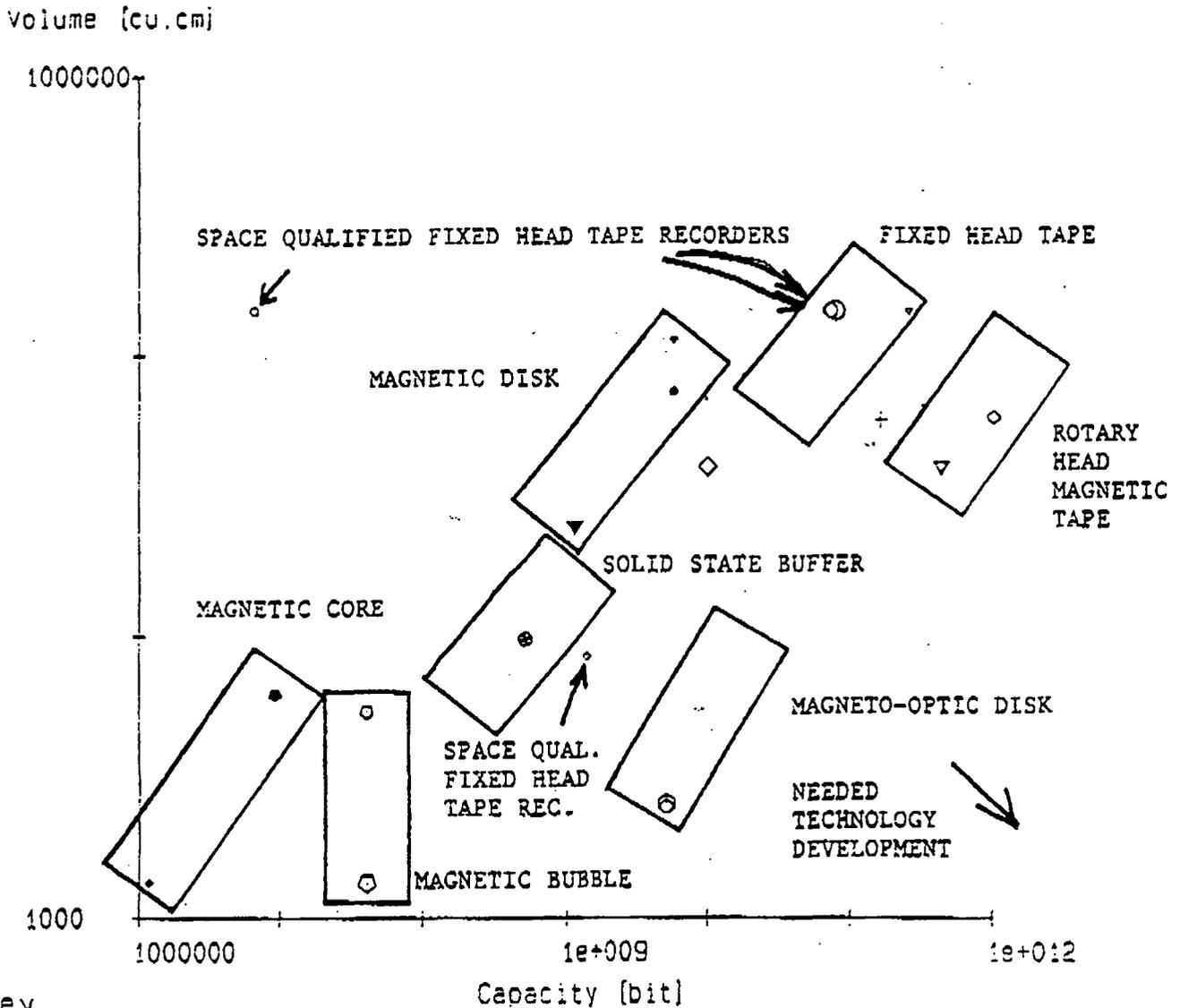


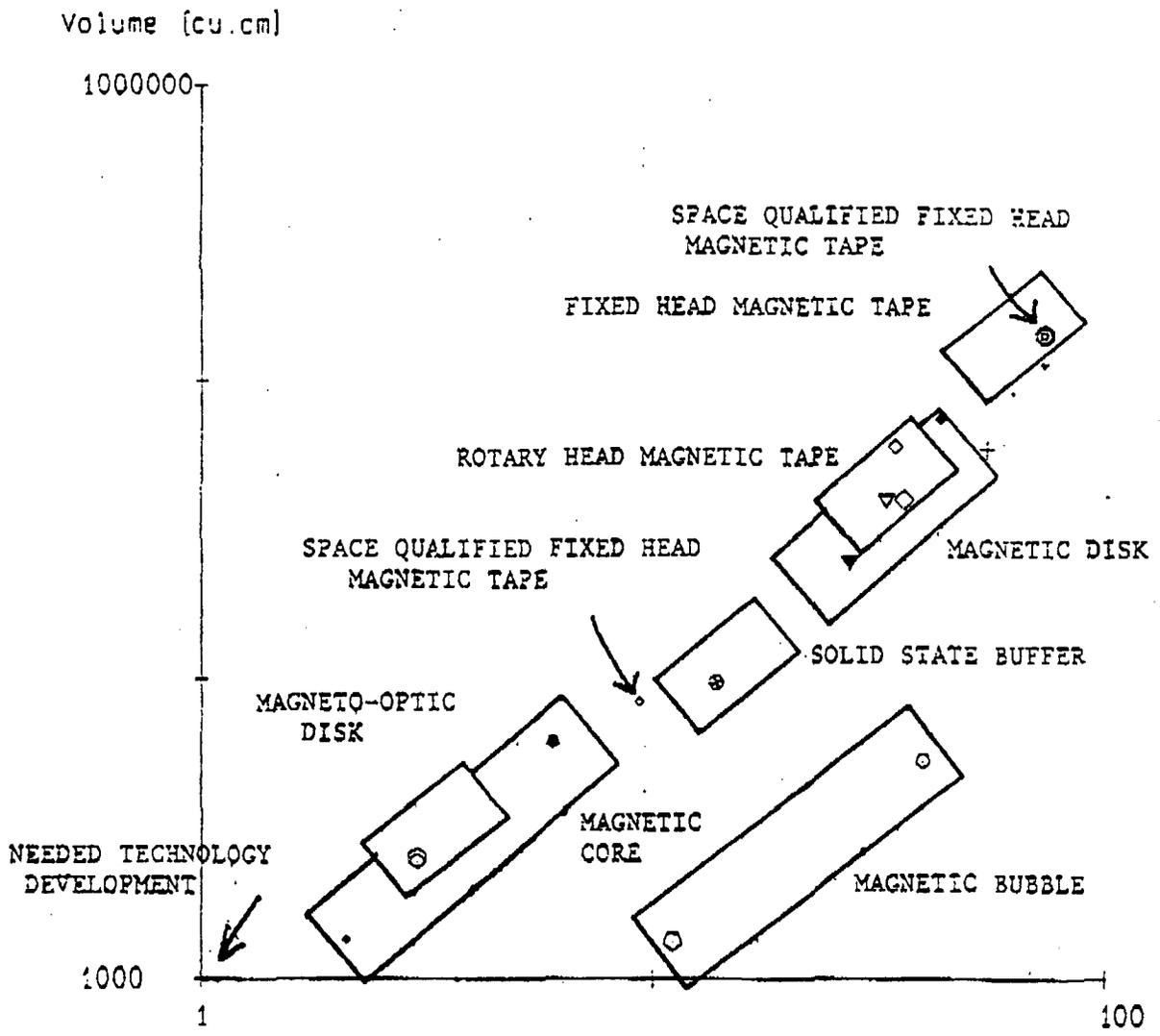
Figure 9: Graph of Capacity vs. Power for Existing Data Storage Systems.



Key

- DDS-6000EC (SPOT-1,2) Tape Recorder
- DDS-6000EC (JERS-1) Tape Recorder
- DDS-6000EC (Landsat) Tape Recorder
- DDS-9000 Tape Recorder
- ▽ PV6410 Rotary Tape Recorder
- ▽ PV6210 Rotary Tape Recorder
- ◊ NASA STR-108 Tape Recorder
- ◊ RDR-300 Tape Recorder
- ◊ 20-MBS Tape Recorder
- 1986 Magnetic Bubble System
- ◊ 1990 Magnetic Bubble System
- DR650RE Magneto-Optic Disk System
- Tahiti-I Magneto-Optic Disk System
- + Spaceflight Optical Disk Recorder
- Ruggedized Advanced Disk System
- Militarized Advanced Disk System
- ▽ Mobile Mass Storage Disk System
- MESA-54RH Core Memory System
- MEGA-9 Core Memory System
- SSR1-5123 Solid State Recorder

Figure 10: Graph of Capacity vs. Volume for Existing Data Storage Systems.



- | Key | Mass [kg] |
|--|--------------------------------------|
| ◦ DDS-6000EC (SPOT-1, 2) Tape Recorder | ⊙ 1990 Magnetic Bubble System |
| ⊙ DDS-6000EC (JERS-1) Tape Recorder | ⊙ DR650RE Magneto-Optic Disk System |
| ⊙ DDS-6000EC (Landsat) Tape Recorder | ⊙ Tahiti-I Magneto-Optic Disk System |
| ▽ DDS-9000 Tape Recorder | + Spaceflight Optical Disk Recorder |
| ▽ PV6410 Rotary Tape Recorder | • Ruggedized Advanced Disk System |
| ▽ PV6210 Rotary Tape Recorder | ▽ Militarized Advanced Disk System |
| ◦ NASA STR-108 Tape Recorder | ▽ Mobile Mass Storage Disk System |
| ◊ RDR-300 Tape Recorder | • MESA-64RH Core Memory System |
| ◊ 2C-MBS Tape Recorder | • MEGA-9 Core Memory System |
| ◊ 1986 Magnetic Bubble System | • SSR1-512B Solid State Recorder |

Figure 11: Graph of Mass vs. Volume for Existing Data Storage Systems.

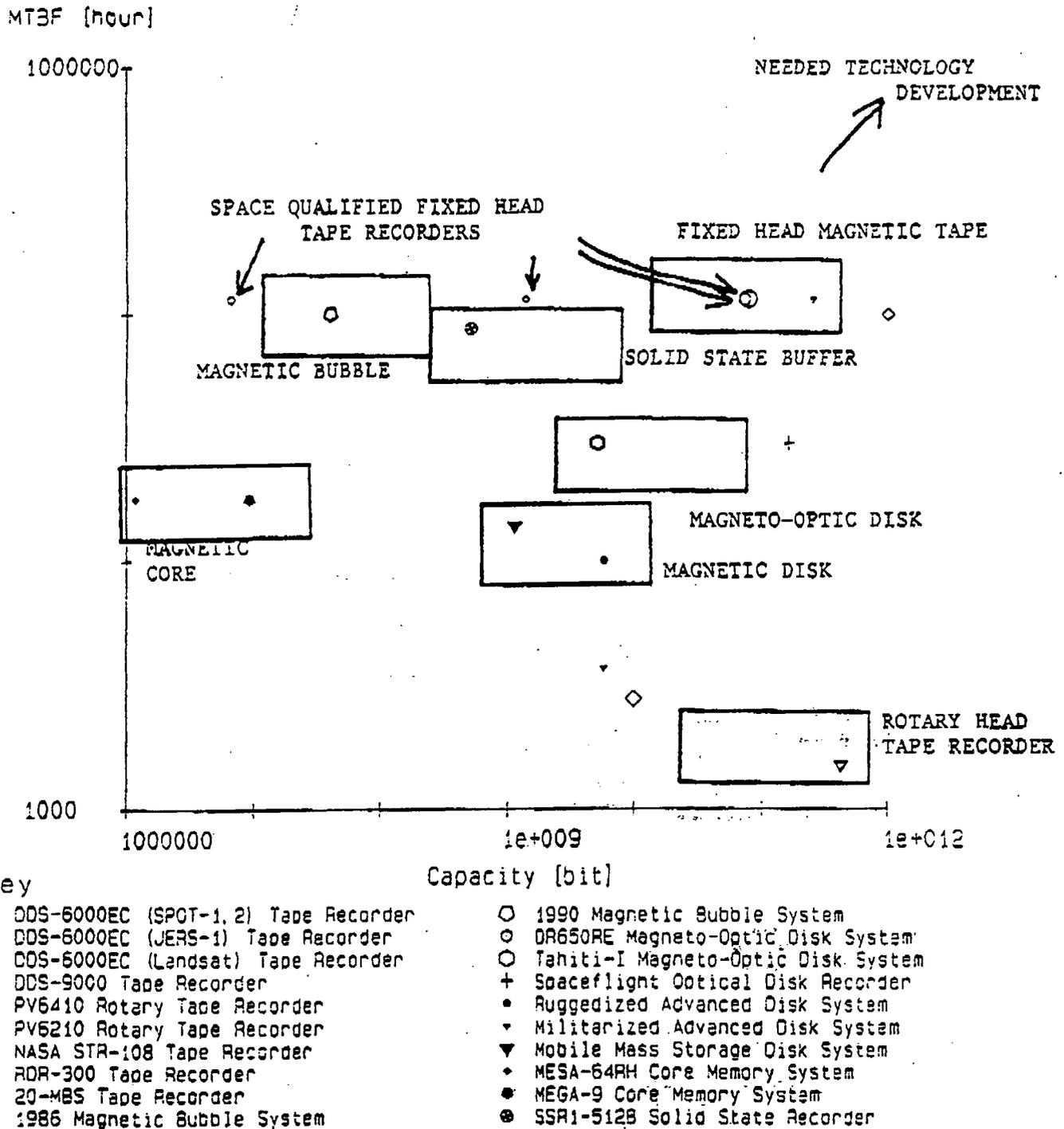


Figure 12: Graph of System Mean Time Before Failure vs. Capacity for Existing Data Storage Systems.

Digital Control of Magnetic Bearings in a Cryogenic Cooler

J. Feeley, A. Law and F. Lind
Department of Electrical Engineering
University of Idaho
Moscow, Idaho 83843

1 Introduction

This paper describes the design of a digital control system for control of magnetic bearings used in a spaceborne cryogenic cooler. The cooler was developed by Philips Laboratories for the NASA Goddard Space Flight Center and is described in detail in Reference 1. Six magnetic bearing assemblies are used to levitate the piston, displacer, and counter-balance of the cooler. The piston and displacer are driven by linear motors in accordance with Stirling cycle thermodynamic principles to produce the desired cooling effect. The counter-balance is driven by a third linear motor to cancel motion induced forces that would otherwise be transmitted to the spacecraft.

An analog control system is currently used for bearing control. The purpose of this project is to investigate the possibilities for improved performance using digital control. Areas for potential improvement include transient and steady state control characteristics, robustness, reliability, adaptability, alternate control modes, size, weight, and cost. The present control system is targeted for the Intel 80196 microcontroller family. The eventual introduction of Application Specific Integrated Circuit (ASIC) technology to this problem may produce a unique and elegant solution both here and in related industrial problems.

2 System Description

The long life and tight clearance requirements of the NASA cooler motivated the choice of frictionless magnetic bearings to magnetically levitate the piston, displacer, and counter-balance shafts. A schematic side view of a typical magnetic bearing assembly is shown in Figure 1. It shows the principal system components for vertical position control and illustrates their relationship within the overall magnetic bearing system. The optical sensors measure the shaft position relative to the housing. Dual sensors and signal conditioning circuitry are used to produce a highly accurate position signal for the system controller. The controller responds to the difference between the desired and actual positions and sends an output signal to a voltage amplifier. The amplifier adjusts the voltage applied to the bearing coil and produces an electromagnetic field between the pole pieces to change the vertical force applied to the shaft. The force moves the shaft to maintain the desired clearance between the shaft and the housing. Operation of the horizontal bearing control system is similar.

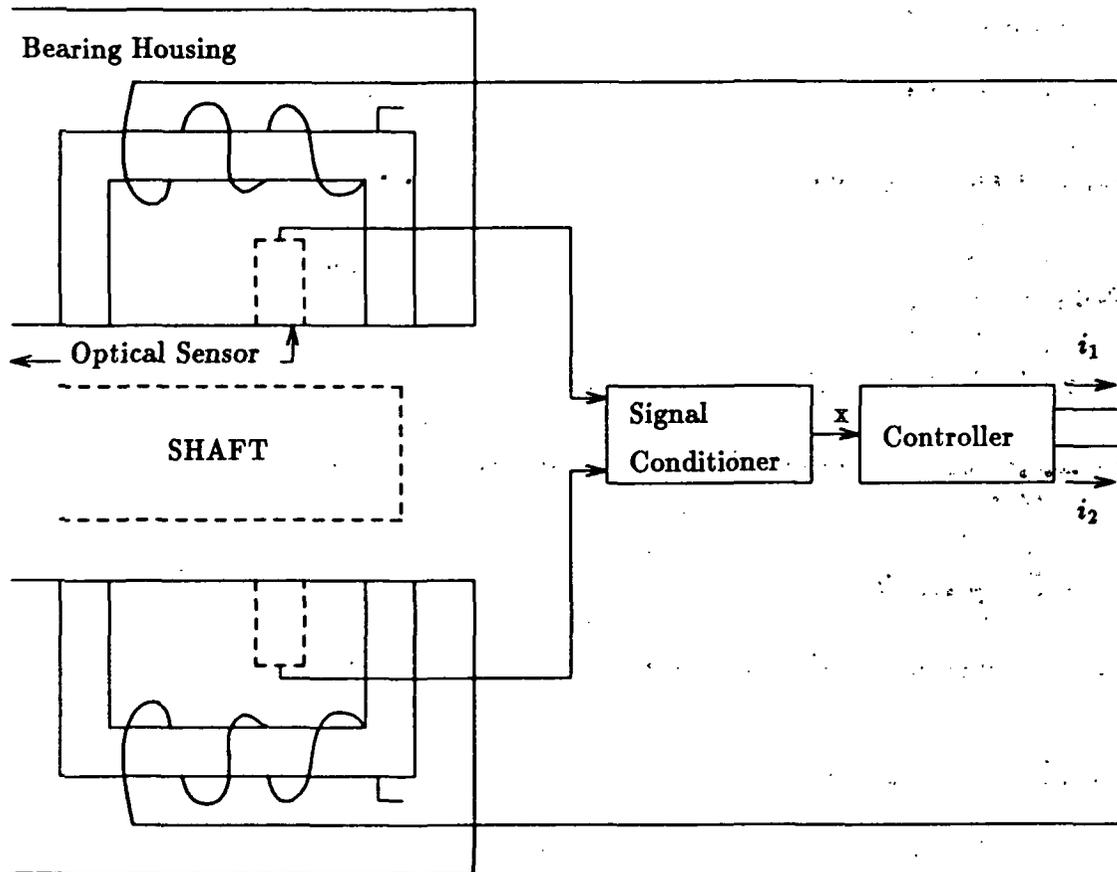


Figure 1: Typical magnetic bearing assembly.

3 Mathematical Models

Traditional control system design is based on a mathematical model of the system to be controlled. Relatively simple, linear, low order models are sought because of the availability of linear design tools and the insight and understanding provided by simple models. Once a candidate controller has been designed using linear methods, its performance is checked in a detailed nonlinear computer simulation of the closed loop system. Several iterative steps between linear system design and nonlinear simulation analysis usually produce an acceptable controller. In this section mathematical models are developed for the principal components in the magnetic bearing system.

3.1 Magnetic Bearing and Levitated Shaft

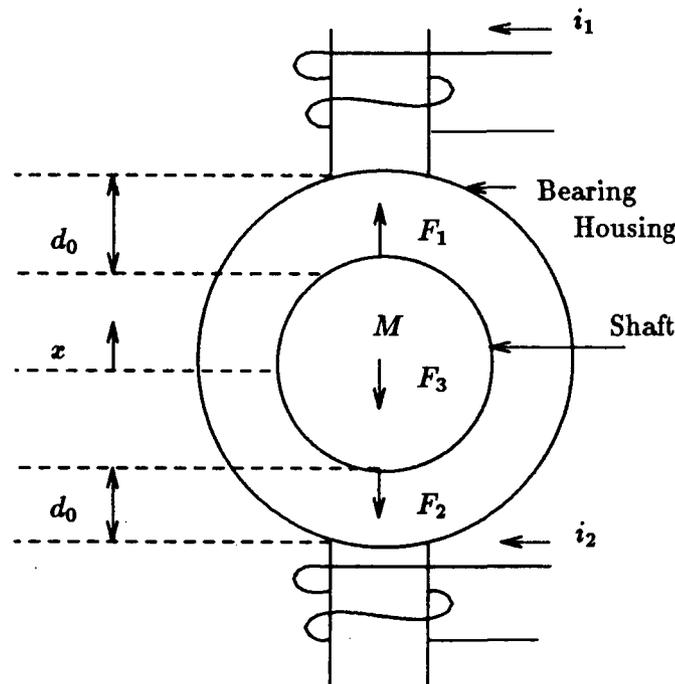


Figure 2: Free body diagram of levitated shaft

A free body diagram of one end of a levitated shaft is shown in Figure 2. The forces F_1 and F_2 are proportional to squares of the coil currents i_1 and i_2 in the respective pole piece coils and inversely proportional to the squares of the distances between the shaft and the pole pieces. F_3 is a viscous damping force due to the motion of the shaft through the helium working fluid. The vertical acceleration of the shaft is given by

$$\frac{d^2 x(t)}{dt^2} = \frac{1}{m}(F_1 - F_2 - F_3) \quad (1)$$

Substituting for the forces in terms of more basic variables gives

$$\frac{d^2 x(t)}{dt^2} = \frac{K_b}{m} \left[\frac{(i_0 + i)^2 (d_0 + x)^2 - (i_0 - i)^2 (d_0 - x)^2}{(d_0 - x)^2 (d_0 + x)^2} \right] - \frac{B}{m} \frac{dx(t)}{dt} \quad (2)$$

where K_b is the bearing constant of proportionality, m is the effective mass of the shaft supported by the bearing, i_0 is a constant bias current, i is the adjustable part of the current used for dynamic control, d_0 is the nominal bearing gap, x is the position of the bearing, and B is a viscous damping coefficient. Expanding Equation 2 in a Taylor series about the point $i = 0$, $x = 0$ yields the linear expression

$$\frac{d^2 x(t)}{dt^2} + K_1 \frac{dx(t)}{dt} - K_2 x(t) = K_3 i(t) \quad (3)$$

where the following definitions have been made:

$$\begin{aligned} K_1 &= \frac{B}{m} \\ K_2 &= \frac{4K_b i_0^2}{m d_0^3} \\ K_3 &= \frac{4K_b i_0}{m d_0^2} \\ K_b &= \frac{K_{eff} d_0^3}{4 i_0^2} \end{aligned}$$

and K_{eff} is the bearing stiffness under static conditions.

3.2 Bearing Actuator

A circuit model of the bearing actuator coils and pole pieces is shown in Figure 3.

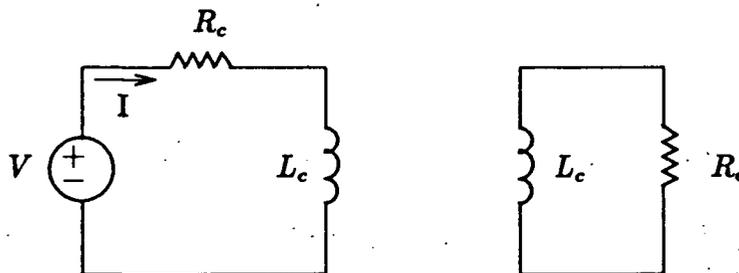


Figure 3: Circuit model of actuator coil

The circuit consists of a primary loop with a voltage source driving current through the equivalent resistance and inductance of the actuator coil. A secondary circuit consisting of a resistor and inductor in series is used to model the eddy currents set up in the actuator's pole piece.

The differential equation relating the actuator voltage and current is

$$\frac{d^2 i(t)}{dt^2} + K_4 \frac{di(t)}{dt} + K_5 i(t) = K_6 \frac{dv(t)}{dt} + K_7 v(t) \quad (4)$$

where the constants are defined as:

$$K_4 = \frac{L_c R_e + L_e R_c}{2L_c L_e}$$

$$K_5 = \frac{R_c R_e}{2L_c L_e}$$

$$K_6 = \frac{1}{2L_c}$$

$$K_7 = \frac{R_e}{2L_c L_e}$$

3.3 Optical Sensor

The dynamics of the optical sensor and its signal conditioning circuitry are very fast with respect to other components in the bearing system and for the purposes of control system design can be considered instantaneous. Noise and other sources of measurement error are small so that the position measurement is assumed to be perfect.

4 Control System Design

The design of a magnetic bearing control system is challenging for a number of reasons. The first, and most fundamental, is that the open loop system is unstable. This is obvious from the model equations and is due to the fact that the magnetic force of attraction on the shaft increases as the bearing gap decreases. Without active control, therefore, the shaft moves inexorably toward the bearing housing. Second, two stable but slow poles exacerbate the the instability by causing additional low frequency phase lag. The combination of these two factors causes traditional series compensators such as proportional-integral-derivative (PID), lead, lag, and lag-lead to fail as viable controllers. Additional control problems arise from the nonlinear system dynamics and the uncertain validity of the system model and its parameter values.

Nevertheless, there are a number of design tools that can be brought to bear on the problem. Even though the design objective is a digital controller, it is good practice to first design a corresponding analog controller. Since the sampling inherent with digital control inevitably degrades system performance the analog controller can serve as a useful benchmark. It is also possible, and this approach will be followed in this preliminary design study, to develop the digital controller from the analog controller by an appropriate mapping from continuous time to discrete time [2].

4.1 Design of Analog Controller

A particularly effective and widely used analog design method is the root locus technique of Evans. This method is especially appropriate in this problem because the open loop system is unstable and the sinusoidal steady state methods of Nyquist and Bode, while still applicable, require special care when used on non-minimum phase systems. So-called modern methods based on a state variable problem formulation and direct digital design techniques are also available and are being investigated. Regardless of what method is used, however, the resulting implementations are often surprisingly similar.

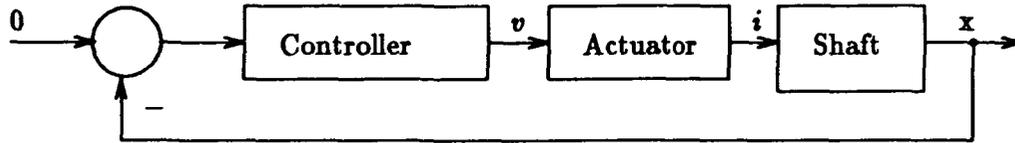


Figure 4: Block diagram of magnetic bearing system

Figure 4. is a block diagram of the closed loop magnetic bearing system and makes the choice of series compensation control explicit. The design procedure is best illustrated by example. Equations 3 and 4 are rewritten below using the numerical values obtained from Reference 1. The shaft equation is

$$\frac{d^2x(t)}{dt^2} + .0128 \frac{dx(t)}{dt} - 44x(t) = 11,940 i(t) \quad (5)$$

and the coil equation is

$$\frac{d^2i(t)}{dt^2} + 764.7 \frac{di(t)}{dt} + 11,460 i(t) = .7143 \frac{dv(t)}{dt} + 1082 v(t) \quad (6)$$

The range of coefficients in these equations suggests time scaling to produce a factor of 100 slowdown to facilitate design and analysis. Replacing t with $100/\tau$ in Equations 5 and 6 produces the time scaled equations for the shaft

$$\frac{d^2x(\tau)}{d\tau^2} + .00218 \frac{dx(\tau)}{d\tau} - .0044x(\tau) = 1.194i(\tau) \quad (7)$$

and the coil

$$\frac{di^2(\tau)}{d\tau^2} + 7.647 \frac{di(\tau)}{d\tau} + 1.146 i(\tau) = .007143 \frac{dv(\tau)}{d\tau} + .1082 v(\tau) \quad (8)$$

Taking Laplace transforms yields the shaft transfer function

$$\frac{X(S)}{I(S)} = \frac{1.194}{(S + .067)(S - .0657)} \quad (9)$$

and the coil transfer function

$$\frac{I(S)}{V(S)} = \frac{.007143(S + 15.15)}{(S - .06)(S + .07)(S + .15)(S + 7.5)} \quad (10)$$

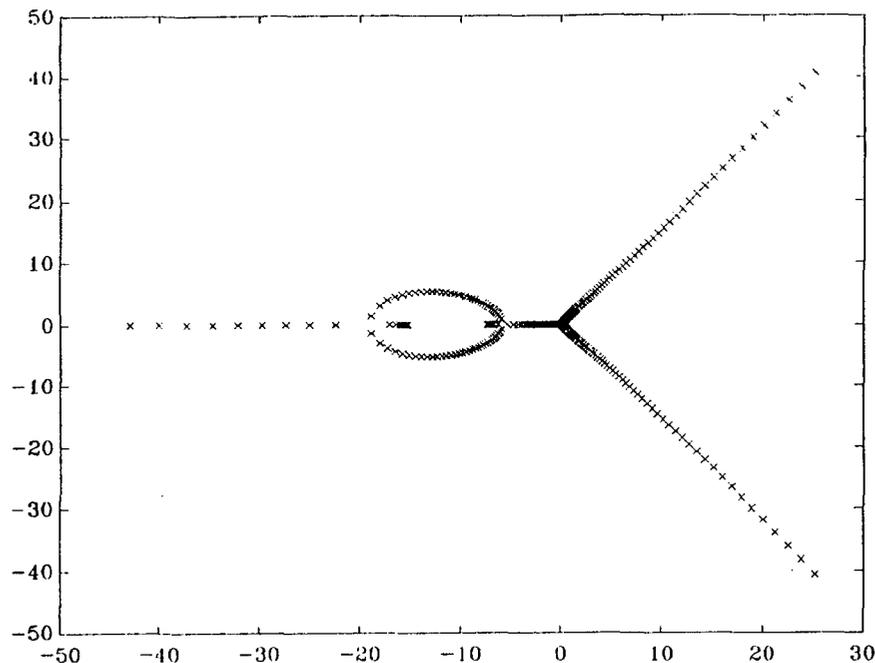


Figure 5: Root locus (scaled for 100:1 slowdown) of the uncompensated magnetic bearing

Cascading these two transfer functions produces the plant transfer function

$$G_p(S) \equiv \frac{X(S)}{V(S)} = \frac{.008(S + 15)}{(S - .06)(S + .07)(S + .15)(S + 7.5)} \quad (11)$$

where numerical values have been rounded for convenience.

The inherent instability of the plant is shown in the root locus of Figure 5 as a proportional gain factor varies from .001 to 10,000. The root locus clearly shows that the uncompensated plant is unstable for all values of gain. The pole-zero pattern and root locus plot reveal the essence of the control problem. The poles at -.07 and -.15 must be canceled, or nearly so, with compensating zeros to allow the unstable root to migrate into the left half plane. The compensator zeros may then be balanced with poles farther into the left half plane for noise suppression and to ensure a causal digital controller. Fast finite zeros may also be added further to the left of the new poles to aid in steady state error reduction.

A candidate controller with a double lead-lag characteristic meeting these requirements is given by

$$G_C(S) = K_C \left[\frac{(S + .1)(S + 100)}{(S + 10)^2} \right]^2 \quad (12)$$

The root locus of the compensated system is shown in Figure 6 as the controller gain varies from 1.25 to 1250.

The locus of the closed loop system roots shows a region of stable operation for controller gains between 6.25 and 250. Setting the controller gain to 62.5 yields a damping ratio from the dominant second order poles of about .78 which should produce fast step

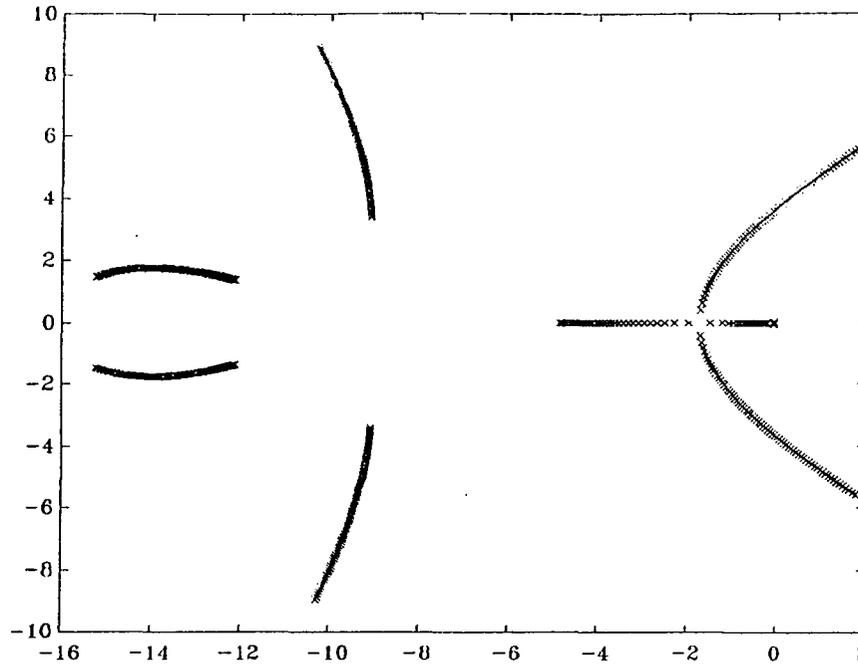


Figure 6: Root locus (scaled for 100:1 slowdown) of double lead-lag compensated system

response with less than 5% overshoot. This setting also produces reasonable robustness to gain variations allowing for a factor of 10 decrease and a factor of 4 increase before stability bounds are reached. The relative steady state error with this compensator and a gain of 62.5 is less than 7% or, about 1.3 microns for a nominal 19 micron gap. Steady state error could readily be reduced to less than 1 micron by increasing the gain to about 82.7 with little degradation of transient performance.

The response of the controlled system with a compensator gain of 62.5 to a unit step input is shown in Figure 7 and verifies transient performance and steady state error approximations. Bode plots of log magnitude and phase versus log frequency for the compensated open loop system with the same controller gain are shown in Figure 8.

They illustrate the care required in interpreting these plots for non-minimum phase systems. Note that while the plots indicate nearly 180 deg of phase shift at low frequencies where the open loop gain is greater than 1, this is due to the right half plane pole. Also note the time and frequency scale factors on the plots.

4.2 Design of Digital Controller

Given an analog controller design as embodied in the transfer function of Equation (12) there are a number of methods of developing an equivalent digital controller. The basic procedure in each approach is to select an appropriate mapping to convert the Laplace transform transfer function to an equivalent Z transform transfer function. Taking the inverse Z transform of this transfer function produces a difference equation that determines the digital control algorithm.

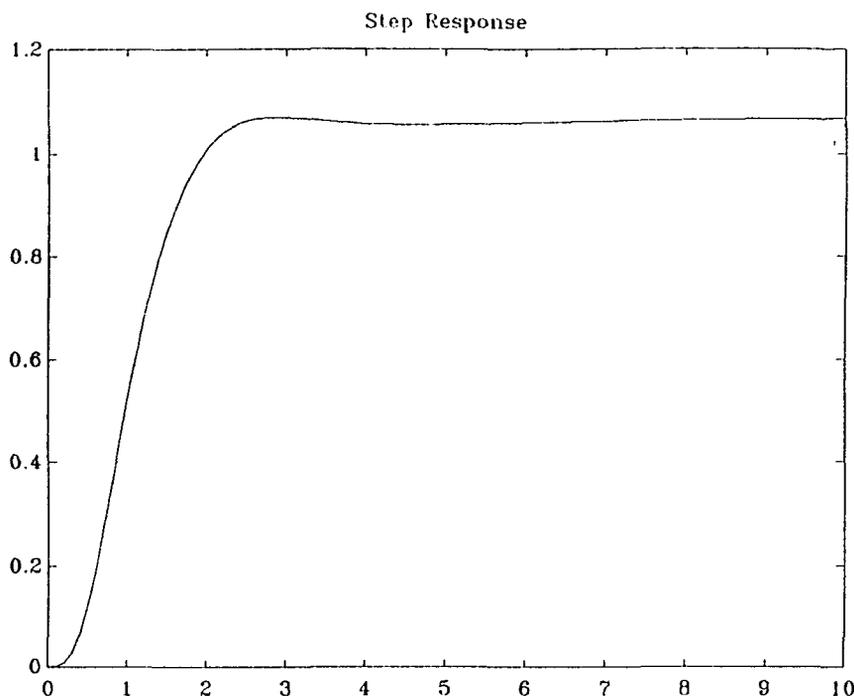


Figure 7: Step response (scaled for 100:1 slowdown) of compensated system

One commonly used method based on trapezoidal integration, and often referred to as the Tustin or bi-linear transformation, uses the substitution

$$S = \frac{2z - 1}{Tz + 1} \quad (13)$$

This mapping introduces the parameter T , the sampling interval. Experience has shown [2] that if the sampling frequency is 20 or more times higher than the highest frequency present in the system the digital controller behaves similarly to the equivalent analog controller and justifies the analog emulation design method. Figure 8 indicates that the bearing system bandwidth is approximately 100 rad/sec or about 16 Hertz. Choosing a sampling frequency of 500 Hertz, over 30 times the system bandwidth, should give good performance. This sampling rate corresponds to a sampling period of 2 msec and thus imposes an upper limit on the total time required to perform the analog to digital conversion, the digital control calculation, and digital to analog conversion in the digital controller.

Applying the Tustin transformation to the analog controller transfer function give the digital transfer function

$$G_c(z) = 62.5 \left[\frac{(1010z - 990)(11,000z + 9000)}{(2000z)^2} \right]^2 \quad (14)$$

Taking the inverse Z transform gives the controller difference equation relating the controller input $y(k)$ to the controller output $u(k)$

$$y(k) = 62.5 [7.714u(k) - 2.5u(k-1) - 12.17u(k-2) + 2u(k-3) + 4.962u(k-4)] \quad (15)$$

7.2.10

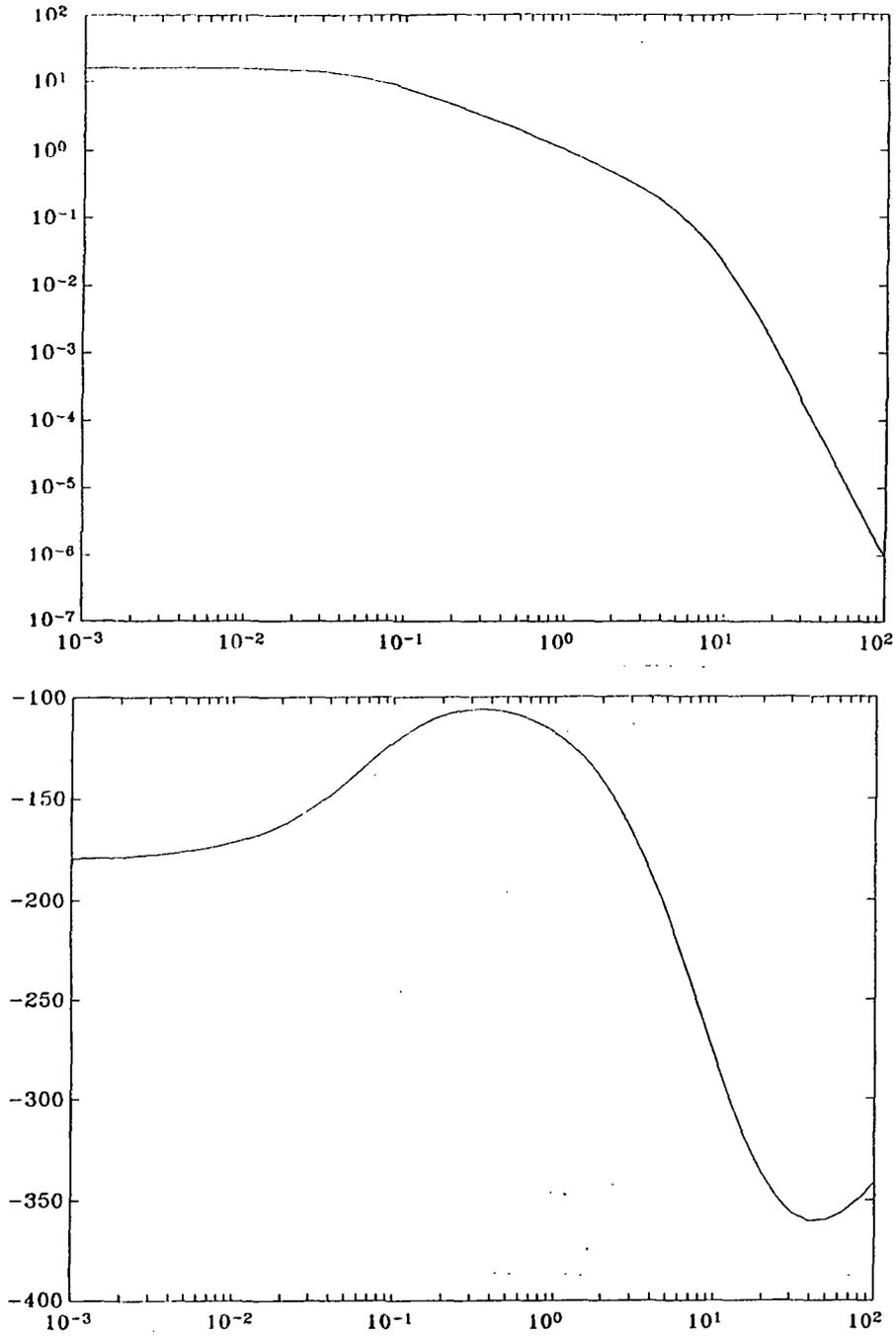


Figure 8: Bode magnitude and phase plots (scaled for 100:1 slowdown) of compensated open loop transfer function

C-5.

Different algebraic approaches will produce different forms for the controller difference equation that may have some timing or scaling advantages, but Equation (15) will serve as a suitable point of departure for further studies. It should also be noted that since the controller transfer function is merely a proper (as opposed to strictly proper) rational function, a controller output $y(k)$ is required at the same instant the latest measurement $u(k)$ is received. This non-causal characteristic can be ameliorated somewhat by appropriate sequencing of computations and data output in the control algorithm.

5 Conclusions and Future Work

A simple nonlinear magnetic bearing model was developed from first principles. Model parameters have been evaluated from readily available data and the model appears consistent with reported experimental data. A double lead-lag analog control system based on the linearized model equations was designed. The controller is effective in stabilizing the unstable plant, providing good transient response, and acceptable steady state error. An equivalent digital controller was developed from the analog controller using Tustin's transformation. The selected sampling rate, 500 Hertz, is over 30 times the highest system frequency and should cause the digital controller to perform nearly as well as the analog controller.

Work is currently proceeding on several fronts. A detailed nonlinear simulation of the magnetic bearing and its controller is being developed. This will be a useful tool for evaluating candidate controllers and sampling rates. A microcontroller based on the Intel 80196 family is being developed. It will be interfaced with an analog computer simulation of the magnetic bearing for real time testing. Candidate architectures for an ASIC real time controller chip are being evaluated. Advanced robust control design methods are being tested for use in direct design of a robust digital controller. Adaptive performance feedback and fuzzy control methods are also being investigated to form the foundation of an adaptive bearing controller.

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] G.F. Franklin, J.D. Powell, and M.L. Workman, "Digital control of dynamic systems," Addison-Wesley, 1990.

"Page missing from available version"

SESSION 7 → N94-71125

PARTS MISSING

Burst Length v	$P(\text{miscorrect} - v)$
17	$1.4(10)^{-34}$
18	$3.6(10)^{-32}$
19	$9.1(10)^{-30}$
20	$2.3(10)^{-27}$
21	$5.8(10)^{-25}$
22	$1.4(10)^{-22}$
23	$3.6(10)^{-20}$
24	$9.1(10)^{-18}$
25	$2.3(10)^{-15}$
26	$5.7(10)^{-13}$
27	$1.4(10)^{-10}$
28	$3.6(10)^{-8}$
29	$9.0(10)^{-6}$
30	$2.3(10)^{-3}$
31	1.0

Table 1: Probability of miscorrection for a (255,223) Reed Solomon code with given burst length

4 Example

The (255,223) Reed Solomon code over $GF(2^8)$ is used to protect satellite communication channels [5]. This code is concatenated with a convolutional inner code. The purpose of the convolutional code is to correct random errors. When the raw bit error rate is too high for the convolutional decoder, a burst of errors occurs in the output. This data is passed to a Reed Solomon decoder for burst correction. The Reed Solomon code has a designed correctable error of $t = 32$. The code corrects all bursts of length 16 or less that occur within a block. If a burst of length greater than 16 occurs, the conditional probabilities of miscorrection can be found through direct application of Equation 2.15 in reference [4]. These results are summarized in Table 1.

5 The Procedure for Finding Burst Errors

The assumption made for following decoding procedure is:

For all v_m and v_n , the probability of a burst of length v_n is less than the probability of a burst of length v_m , where $v_m < v_n$.

The decoding procedures are as follows:

STEP 1: Assume a value for v_{max} and t . Form the X_1 equations by applying the v_{max} and t values to (6).

If this is the end of the decoding, go to STEP 7. If this is not the

STEP 2: end of the decoding, computer the syndromes. Find all the common roots of the X_1 equations.

STEP 3: If only one X_1 from (6) is in common, this is the real X_1 and go to STEP 4. If more than one of the X_1 are common, go to STEP 5.

STEP 4: Obtain all Y's from equation (4). Go to STEP 6.:

Find all the burst-error lengths by solving for the Y's. Chose one

STEP 5: burst error which has the shortest length. If more than one error has the shortest length, this error is uncorrected.

STEP 6: Report the decoding result. GO to STEP 2.

STEP 7: Exit.

6 Decoding Examples

First form the X_1 equations by applying following values.

The length of the burst error: $v \leq 4$
 The designed correctable errors: $t = 3$
 The number of the syndromes: $2t = 6$

The field containing the code words: **GF(16)**

Assume $v = 4$, then equation 3 becomes:

$$\begin{bmatrix} \frac{S_1}{X_1} \\ \frac{S_2}{X_1^2} \\ \frac{S_3}{X_1^3} \\ \frac{S_4}{X_1^4} \\ \frac{S_5}{X_1^5} \\ \frac{S_6}{X_1^6} \end{bmatrix} = \begin{bmatrix} \alpha^0 \alpha^1 \alpha^2 \alpha^3 \\ \alpha^0 \alpha^2 \alpha^4 \alpha^6 \\ \alpha^0 \alpha^3 \alpha^6 \alpha^9 \\ \alpha^0 \alpha^4 \alpha^8 \alpha^{12} \\ \alpha^0 \alpha^5 \alpha^{10} \alpha^0 \\ \alpha^0 \alpha^6 \alpha^{12} \alpha^3 \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix}$$

and the inverse of $[\alpha_v]$ matrix is:

$$= \begin{bmatrix} \alpha^0 \alpha^1 \alpha^2 \alpha^3 \\ \alpha^0 \alpha^2 \alpha^4 \alpha^6 \\ \alpha^0 \alpha^3 \alpha^6 \alpha^9 \\ \alpha^0 \alpha^4 \alpha^8 \alpha^{12} \end{bmatrix}^{-1} = \begin{bmatrix} \alpha^{10} \alpha^2 \alpha^0 \alpha^4 \\ \alpha^1 \alpha^5 \alpha^9 \alpha^{11} \\ \alpha^{13} \alpha^8 \alpha^1 \alpha^9 \\ \alpha^1 \alpha^9 \alpha^8 \alpha^{13} \end{bmatrix}$$

The magnitude Y's can be expressed as:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} \alpha^{10} \alpha^2 \alpha^0 \alpha^4 \\ \alpha^1 \alpha^5 \alpha^9 \alpha^{11} \\ \alpha^{13} \alpha^8 \alpha^1 \alpha^9 \\ \alpha^1 \alpha^9 \alpha^8 \alpha^{13} \end{bmatrix} \begin{bmatrix} \frac{S_1}{X_1} \\ \frac{S_2}{X_1^2} \\ \frac{S_3}{X_1^3} \\ \frac{S_4}{X_1^4} \\ \frac{S_5}{X_1^5} \\ \frac{S_6}{X_1^6} \end{bmatrix}$$

Next multiply $[\alpha_v]^{-1}$ by the remaining equations in (3). i.e. evaluate (5). In this manner general X_1 equations are obtained:

$$\begin{bmatrix} \frac{S_1}{x_1^0} \\ \frac{S_2}{x_1^1} \\ \frac{S_3}{x_1^2} \\ \frac{S_4}{x_1^3} \end{bmatrix} = \begin{bmatrix} \alpha^0 \alpha^5 \alpha^{10} \alpha^0 \\ \alpha^0 \alpha^6 \alpha^{12} \alpha^{13} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha^6 \alpha^0 \alpha^4 \alpha^{12} \\ \alpha^3 \alpha^4 \alpha^4 \alpha^{14} \end{bmatrix} \begin{bmatrix} \frac{S_1}{X_1} \\ \frac{S_2}{X_1^2} \\ \frac{S_3}{X_1^3} \\ \frac{S_4}{X_1^4} \end{bmatrix}$$

If the syndromes are given, the X_1 equations for a particular error are formed. After obtaining the X_1 , the Y 's can be obtained by solving a set of linear equations.

Example 1

Assume that the syndromes are:

$$\begin{aligned} S_1 &= e(\alpha) = \alpha^{13} \\ S_2 &= e(\alpha^2) = \alpha^{11} \\ S_3 &= e(\alpha^3) = \alpha^0 \\ S_4 &= e(\alpha^4) = \alpha^7 \\ S_5 &= e(\alpha^5) = \alpha^5 \\ S_6 &= e(\alpha^6) = \alpha^0 \end{aligned}$$

With these syndromes, the X equations become:

$$a^4 x_1^4 + a^{11} x_1^3 + a^4 x_1^2 + a^4 x_1 + \alpha^5 = 0 \quad (7)$$

$$a^1 x_1^5 + a^0 x_1^4 + a^4 x_1^3 + a^6 x_1^2 + \alpha^0 = 0 \quad (8)$$

Because there are only a finite number of field elements to check, the simplest way to find the zeros of (7) and (8) is with an exhaustical search, a process known as a Chien search.

By a Chien search, the common root of (7) and (8) is α^1 the only root. Hence, the beginning location of the burst error is α^1 .

The magnitudes Y 's can now be solved for easily:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} \alpha^{10}\alpha^2\alpha^0\alpha^4 \\ \alpha^1\alpha^5\alpha^9\alpha^{11} \\ \alpha^{13}\alpha^8\alpha^1\alpha^9 \\ \alpha^1\alpha^9\alpha^8\alpha^{13} \end{bmatrix} \begin{bmatrix} \alpha^{12} \\ \alpha^9 \\ \alpha^{12} \\ \alpha^3 \end{bmatrix} = \begin{bmatrix} \alpha^0 \\ \alpha^0 \\ \alpha^0 \\ \alpha^0 \end{bmatrix}$$

Finally, the error equation is:

$$e(x) = x^1 + x^2 + x^3 + x^4$$

Example 2

Assume that the syndromes are:

$$\begin{aligned} S_1 &= e(\alpha) = \alpha^8 \\ S_2 &= e(\alpha^2) = \alpha^1 \\ S_3 &= e(\alpha^3) = \alpha^0 \\ S_4 &= e(\alpha^4) = \alpha^9 \\ S_5 &= e(\alpha^5) = \alpha^2 \\ S_6 &= e(\alpha^6) = \alpha^{11} \end{aligned}$$

With these syndromes, the X_1 equations become:

$$\alpha^{14}x_1^4 + \alpha^1x_1^3 + \alpha^4x_1^2 + \alpha^6x_1 + \alpha^2 = 0 \quad (9)$$

$$\alpha^{11}x_1^5 + \alpha^5x_1^4 + \alpha^4x_1^3 + \alpha^8x_1^2 + \alpha^{11} = 0 \quad (10)$$

Since only a finite number of field elements needed be checked, the simplest way to find the zeros of (9) and (10) is again a Chien search.

By a Chien search, the common roots of (9) and (10) are $\alpha^0, \alpha^3, \alpha^4$. There is more than one root. All three roots are the potential beginning location of the burst errors. They have to be considered one by one.

CASE 1: Assume that $x_1 = \alpha^0$. The magnitude Y 's can be solved for:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} \alpha^{10}\alpha^2\alpha^0\alpha^4 \\ \alpha^1\alpha^5\alpha^9\alpha^{11} \\ \alpha^{13}\alpha^8\alpha^1\alpha^9 \\ \alpha^1\alpha^9\alpha^8\alpha^{13} \end{bmatrix} \begin{bmatrix} \alpha^8 \\ \alpha^0 \\ \alpha^1 \\ \alpha^9 \end{bmatrix} = \begin{bmatrix} \alpha^6 \\ \alpha^9 \\ \alpha^6 \\ \alpha^4 \end{bmatrix}$$

The error equation is:

$$e_1(x) = \alpha^6 + \alpha^9 x^1 + \alpha^6 x^2 + \alpha^4 x^3$$

CASE 2: Assume that $x_1 = \alpha^3$. The magnitude Y 's can be solved for:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} \alpha^{10} \alpha^2 \alpha^0 \alpha^4 \\ \alpha^1 \alpha^5 \alpha^9 \alpha^{11} \\ \alpha^{13} \alpha^8 \alpha^1 \alpha^9 \\ \alpha^1 \alpha^9 \alpha^8 \alpha^{13} \end{bmatrix} \begin{bmatrix} \alpha^5 \\ \alpha^{10} \\ \alpha^6 \\ \alpha^{12} \end{bmatrix} = \begin{bmatrix} 0 \\ \alpha^{14} \\ \alpha^{10} \\ \alpha^0 \end{bmatrix}$$

The error equation is:

$$e_2(x) = \alpha^{14} x^4 + \alpha^{10} x^5 + \alpha^0 x^6$$

CASE 3 Assume that $x_1 = \alpha^4$. The magnitude Y 's can be solved for:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{bmatrix} \alpha^{10} \alpha^2 \alpha^0 \alpha^4 \\ \alpha^1 \alpha^5 \alpha^9 \alpha^{11} \\ \alpha^{13} \alpha^8 \alpha^1 \alpha^9 \\ \alpha^1 \alpha^9 \alpha^8 \alpha^{13} \end{bmatrix} \begin{bmatrix} \alpha^4 \\ \alpha^8 \\ \alpha^3 \\ \alpha^8 \end{bmatrix} = \begin{bmatrix} \alpha^{14} \\ \alpha^{10} \\ \alpha^0 \\ 0 \end{bmatrix}$$

The error equation is:

$$e_3(x) = \alpha^{14} x^4 + \alpha^{10} x^5 + \alpha^0 x^6$$

Through this process two distinct errors (note that $e_2(x) = e_3(x)$) have been obtained. This is the error CASE 3 in which the received code word is in two decoding spheres. As previously discussed, the error that has the higher probability should be chosen. The user has to make a decision. In this example, it is assumed that the probability of a burst error of length v_n is less than the probability of a burst error of length v_m , where $v_n > v_m$. Therefore the $e_2(x)$ should be chosen.

7 Summary

For a burst-error environment, the error correcting ability of RS codes can be extended beyond the Singleton and Rieger Bounds with a high degree of confidence. This is significant in that the code rate of a code can be reduced without any reduction in the burst-error correcting ability. A new decoding algorithm has been shown for correcting bursts greater than $(n - k)/2$ in this paper. Many uses can be found for burst error correcting decoders.

The method discussed in this paper will also help to understand a new composite-error correcting algorithm, which will be discussed in future papers.

References

- [1] W. W. Peterson, "Encoding and Error-Correction Procedures for the Bose-Chaudhuri Codes," *IEEE Trans. Inf. Theor.* IT-6 (1960), pp. 459-470.
- [2] D. C. Gorenstein, and N. Zierler, "A Class of Error-Correcting Codes in p Symbols," *J. Soc. Indust. Appl. Math.* 9 (1961), pp. 207-214.
- [3] R. E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Company, 1983.
- [4] P. Owsley, "Burst Error Correction Extensions for Reed Solomon Codes," PH.D Dissertation, E.E. Dept. University of Idaho, July 1988.
- [5] Consultive Committee for Space Data System, *Telemetry Channel Coding "Blue Book,"* 1984.

Minimal Test Set for Stuck-at Faults in VLSI

M. Shamanna and S. Whitaker
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - Minimal test sets have the property that each input vector simultaneously tests several faults in a network. Existing techniques to determine a minimal set of detection tests rely heavily on complicated algebraic techniques. In this paper, two new methods are presented which do not require Boolean algebra or Karnaugh maps. The first is a graphical approach using fault folding graphs. The second is a design by inspection technique. This work follows the unique approach of first finding all the faults that can be detected by a single test. This tremendously reduces the work required to determine a minimal test set. The design by inspection method could be automated for programmatic generation of minimal stuck-at fault tests.

1 Introduction

System and circuit design, over the years has advanced much and achieved a great deal of sophistication and reliability. Because of increased complexity and the lack of externally available test points, circuits implemented in VLSI are especially difficult to test. Furthermore, the shrinking distance between signal lines due to reduced geometries and complex processes has introduced new types of failures, such as data dependent or neighboring-interaction faults. When devices are tested only functionally their reliability can be relatively low. Specialized fields, viz. Space Engineering and Aero-Engineering often need *zero-defect* systems. To ensure zero-defect conditions all the system components must be tested comprehensively for defects and faults.

The problems of determining whether or not a digital circuit operates correctly and of ensuring its correct operation in spite of failure of certain parts are of both practical concern and theoretical interest. Present day digital systems may be disabled by almost any internal failure. Due to the testing constraints, the emphasis is shifting from efficient designs to testable designs [9]. Considerable work has been done in recent years in the field of Fault diagnosis [6,8,9,10,11]. But there is still a need for simpler, better and more efficient test techniques.

Often the information provided to test engineers concerning a digital circuit consists solely of its logic schematic diagram. Boolean equations, Karnaugh maps and State tables are often not available and are tedious to derive. Since the starting data for generation of the *Fault Diagnostic* tests is often the logic schematic diagram, methods developed to generate fault-tests should take this into account [3].

The main objective of this work is to develop techniques to arrive at a minimal set of test vectors for stuck-at faults. Although this problem has been studied by many researchers, a sufficiently simple solution is not available at present. The *simple* methods usually generate a single stuck-at fault test set which is *nearly minimal* but not *completely minimal*. The methods which do generate minimal stuck-at fault test sets, are quite complicated and can get out of hand for large circuits. This problem first gained the attention of researchers in the Sixties [1] and continued in the Seventies [2,4,5,17], but none of these complex methods became popular. The procedures proposed here are very simple and highly efficient. The only required data is the logic schematic diagram. There is no need to either derive Boolean equations or Fault tables, etc..

Section 2 reviews, the need for testing and various terms and terminologies associated with testing. Section 3 reviews the previous work in connection with minimal fault test vectors. In Section 4, the procedure to arrive at the minimal test set for stuck-at faults is derived and examples are given.

2 Testability in Combinational Logic Circuits

2.1 Testing Approaches

There are two basic approaches to testing digital systems: *Functional* and *Structural*[1]. In functional testing, the aim is to verify that the system under test is fault-free by determining that all its functions (tasks) are being carried out as specified (a counter increments or decrements correctly, Memory locations can be read from or written into, etc.). In structural testing, all the individual hardware constituents of the unit are ensured to be perfect. Thus a structural test assures every AND gate gives an output of 1 iff all its inputs have a logic value of 1: that every flipflop can be properly set, reset or toggled; and so forth.

Functional tests are based on heuristics and are derived manually on the basis of familiarity with the system without regard for fine structural details. It is not unusual for functional tests to be limited in fault detecting capabilities to only half of the possible faults in a digital unit [13]. Structural tests are more amenable to being derived algorithmically and hence the whole process could be automated. Structural testing can be made complete although it might result in excessively long test times for large systems.

2.2 Fault Detection in Combinational Circuits

Faults could be of two types: *temporary* and *permanent*. Temporary faults may occur due to noise and nonideal transient behavior of switching components while the permanent faults may result from component or physical failures. This work deals only with permanent faults. This work is also limited to the fault detection in combinational circuits, that is, feedback loops are not allowed in the circuits being tested.

One way of determining whether a combinational circuit is fault-free is to apply all possible input combinations and verify the veracity of the results by comparing them with the truth table or a faultless version of the same circuit. Any deviation indicates the

presence of a fault. If we know the mapping of all possible faults to the deviations observed, we could diagnose the fault (in the case of 1-1 mapping) or at least classify the faults within a subset of faults whose effects on the circuit outputs are the same (onto-mapping).

Such exhaustive tests are very lengthy and hence impractical. Moreover for most circuits, it is possible to diagnose the faults by a shorter test sequence. Such tests are referred to as either fault detection or fault location tests depending on whether they merely reveal the presence of a fault or in addition locate it and diagnose it. The precise identification of the fault that has occurred is very complicated and often impossible, since two different faults may give rise to identical incorrect responses. Therefore the main effort is directed to devising tests which locate the impaired sub-circuit.

In order to arrive at simple and practical fault detection procedures, it is necessary to make several simplifying assumptions. It is a normal practice in the field of fault diagnosis to assume that only single faults occur. Testing for single faults will simultaneously test certain multiple faults too but not all multiple faults are tested. If a network contains logical redundancy, then not all faults are detectable [5].

2.3 Fault Models

The primitive fault or "stuck-at" fault model is a widely accepted fault model. It supposes that the *failure mechanism* in a gate results in its inputs or outputs being either stuck-at 1 or stuck-at 0. The notation $s\text{-}a\text{-}1(0)$ and $s@1(0)$ are used interchangeably to represent stuck-at-1(0) faults.

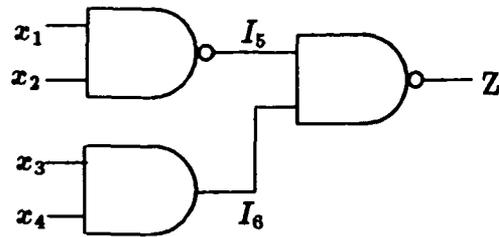
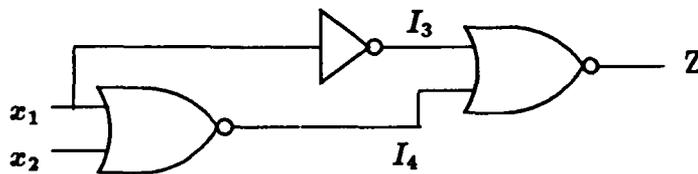
An AND gate with 1 or more inputs or the output $s\text{-}a\text{-}0$ supplies a logical signal of 0 to all of the gates to which it is connected. A j -input AND gate with k -inputs $s\text{-}a\text{-}1$ becomes a $j\text{-}k$ input AND gate, while the output $s\text{-}a\text{-}1$ causes the gate to supply a permanent 1 to all its loads. Similar statements can be made about OR gates: one or more inputs $s\text{-}a\text{-}1$ cause the OR gate to act as though its output were $s\text{-}a\text{-}1$, and OR gate inputs $s\text{-}a\text{-}0$ can no longer affect gate operation. An inverter input $s\text{-}a\text{-}0$ is equivalent to its output permanently at logical 1 and an inverter output $s\text{-}a\text{-}0$ is equivalent to its input permanently at 1.

It is to be noted that under the stuck-at fault model, failures cause fixed signals to appear at leads, ie. signals become *clamped*. Thus tests based on stuck-at fault models deal with *static faults*. Parameters that affect dynamic behavior such as switching speed are verified by other tests.

2.4 Classical and Testability Notions of Redundancy

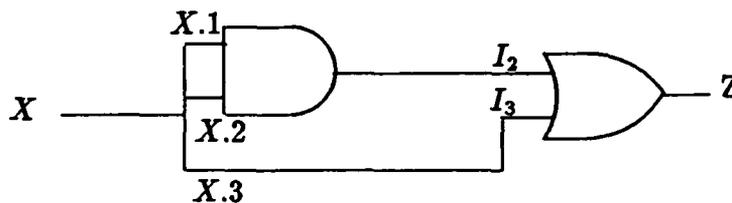
In switching theory terminology, a circuit is classified as either *Redundant* or *Irredundant*. The classical definition of redundancy is that a circuit is irredundant if a connection (path) or gate can be cut without altering the output functions. In terms of fault testability, Whitney[2] has classified the circuits as shown below:

1. A path or gate g is *irredundant* if there exists input vectors x_0 and x_1 such that x_0 is a test for $g\text{-at-}0$ and x_1 is a test for $g\text{-at-}1$.

Figure 1: Path I_5 is irredundantFigure 2: Path I_4 is partially redundant

2. A path or gate is partially redundant iff there exists a test for the fault g -at-0, but there exists no test for the fault g -at-1, or vice-versa.
3. A path or a gate g is fully redundant iff there exists tests for neither g -at-0 nor g -at-1.

Simple examples for the different classes of the circuits are shown in Figures 1, 2 and 3 respectively. In Figure 1, path I_5 is irredundant, since $\langle x_1, \dots, x_4 \rangle = \langle 0111 \rangle$ is a test for I_5 -at-0 and $\langle x_1, \dots, x_4 \rangle = \langle 1111 \rangle$ is a test for I_5 -at-1. In Figure 2, path I_4 is partially redundant since $\langle x_1, x_2 \rangle = \langle 1, 1 \rangle$ is a test for I_4 -at-1; but there does not exist a test for I_4 -at-0. In Figure 3, path $x.2$ is fully redundant since there does not exist a test for either $x.2$ -at-0 or $x.2$ -at-1. Henceforth x will be called as *fanout origin* and $x.1, x.2, x.3$ as *fanout branches*.

Figure 3: Path $x.2$ is fully redundant

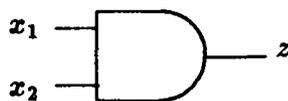


Figure 4: AND Gate

3 Review of Existing Methods

One of the popular methods being used to find a minimal test vector set is the fault table method and the subsequent application of Petrick's method [20]. The fault table could be filled using the path sensitization technique [1] where a given fault is provoked by the test vector and a path is sensitized through the logic to propagate the fault to a primary output. The fault table could also be filled using the Boolean Differences method [12]. This method is an algebraic-type procedure for determining the complete set of tests that detect a given fault. Kohavi [15] developed simple procedure called *a and b tests* but this was restricted to two level AND-OR logic circuits and requires the use of K-maps. Another version of the same technique was given by Bearnson and Carroll [17]. The *enf* method was first demonstrated by Armstrong [1] but this was quite complicated. An improved version of the same was outlined by Fridrich and Davis [5] and S.M.Reddy [4]. However all these methods involve considerable computations and, in addition are time consuming too.

4 Minimal Test set For Irredundant Circuits

In this section, the concept of fault folding is introduced first and then the procedures to derive a set of minimal test vectors is demonstrated. Only irredundant, combinational logic circuits are considered, with the single stuck-at fault assumption.

Fault-Folding is a very powerful technique and was developed by Kilin To [3]. Any two faults $x_1\text{-a-}\alpha$ and $x_2\text{-a-}\beta$ where $\alpha, \beta \in \{0,1\}$ are said to be *test equivalent*, (written as $x_1\text{-a-}\alpha \leftrightarrow x_2\text{-a-}\beta$) iff a test for one fault is also a test for the other that is, $t(x_1\text{-a-}\alpha) \in T(x_1\text{-a-}\alpha)$ implies $t(x_1\text{-a-}\alpha) \in T(x_2\text{-a-}\beta)$ and vice-versa, where $T(x_i\text{-a-}\gamma)$ is the set of input vectors that can serve as a test for fault $x_i\text{-a-}\gamma$ and $t(x_1\text{-a-}\alpha)$ is one of the input vectors. Two faults are said to be *test implied*, (written as $x_1\text{-a-}\alpha \rightarrow x_2\text{-a-}\beta$) if the implication is only one way i.e. $t(x_1, \alpha) \in T(x_1, \alpha)$ implies $t(x_1, \alpha) \in T(x_2, \beta)$ but there is a $t'(x_2, \beta) \notin T(x_1, \alpha)$.

Consider the AND gate shown in Figure 4. The faults $x_1\text{-a-}0$, $x_2\text{-a-}0$, and $z\text{-a-}0$ are said to be test equivalent where as $z\text{-a-}0$ is test implied by the faults $x_1\text{-a-}0$ and $x_2\text{-a-}0$.

The fault $x_1\text{-a-}\alpha$ is said to test cover $x_2\text{-a-}\beta$ (written as $x_1\text{-a-}\alpha \leq_t x_2\text{-a-}\beta$) iff $x_1\text{-a-}\alpha$ is test equivalent to $x_2\text{-a-}\beta$ ($x_1\text{-a-}\alpha \leftrightarrow x_2\text{-a-}\beta$) or $x_2\text{-a-}\beta$ is test implied by $x_1\text{-a-}\alpha$ ($x_1\text{-a-}\alpha \rightarrow x_2\text{-a-}\beta$). The fault folding operation $<_f$ is defined as

$$F_i <_f F_j = \begin{cases} F_i & \text{if } F_i \leq_t F_j \\ \text{undefined} & \text{otherwise} \end{cases}$$

7.4.6

The motivation for folding is that a test for fault F_i also tests the fault F_j ; hence it is only necessary to find tests for F_i .

The set of tests T is said to be a complete set of tests iff the presence of all stuck-at faults in the circuit could be detected by exercising the test set T . At this point some of the *classic* Theorems of testability are restated.

Theorem 1 *A set of tests that can detect all single faults at primary inputs of a Non-Reconvergent Fanout (NRFO) combinational circuit is a complete set of detection tests.*

Theorem 2 *A set of tests that can detect all single faults at all primary inputs and fanout branches of a Reconvergent Fanout (RFO) combinational circuit is a complete set of detection tests.*

Lemma 1 *If several faults are equivalent given by $S = \{x_{1-a-i_1}, x_{2-a-i_2}, \dots, x_{n-a-i_n}\}$ where $i_1, i_2, \dots, i_n \in \{0,1\}$ then the set of primary input faults F formed by fault folding of an NRFO circuit should contain only one representative (any) from the fault set S .*

Example 1 *For the circuit of Figure 5, find the minimal set of faults by fault-folding.*

First, assume the output f to be s-a-0. A s-a-0 fault on a NAND gate output is test implied by a s-a-1 fault on any of its inputs. Therefore a test for either x_9 -a-1 or x_{10} -a-1 is also a test for f -a-0. This is indicated in Figure 6. Using the transitive properties of test cover relations x_9 -a-1 is test covered by x_6 -a-0 or x_5 -a-0. This process is continued till the primary inputs are reached.

The same procedure is also followed for f -a-1 (Figure 7). The graphs formed by the fault-folding are called *fault-folding* graphs. Fault folded classes are formed by tracing different paths of the graphs and applying fault folding. In our example, the fault-folded classes are:

$$\begin{aligned} \{x_5\text{-a-0}, x_6\text{-a-0}\} &<_t x_9\text{-a-1} <_t f\text{-a-0} = \{x_5\text{-a-0}, x_6\text{-a-0}\} \\ \{x_1\text{-a-0}, x_2\text{-a-0}\} &<_t x_3\text{-a-1} <_t \{x_7\text{-a-0}, x_8\text{-a-0}\} <_t x_{10}\text{-a-1} <_t f\text{-a-0} = \{x_1\text{-a-0}, x_2\text{-a-0}\} \\ x_4\text{-a-1} &<_t \{x_7\text{-a-0}, x_8\text{-a-0}\} <_t x_{10}\text{-a-1} <_t f\text{-a-0} = x_4\text{-a-1} \\ x_5\text{-a-1} &<_t \{x_9\text{-a-0}, x_{10}\text{-a-0}\} <_t f\text{-a-1} = x_5\text{-a-1} \\ x_6\text{-a-1} &<_t \{x_9\text{-a-0}, x_{10}\text{-a-0}\} <_t f\text{-a-1} = x_6\text{-a-1} \\ x_8\text{-a-1} &<_t \{x_9\text{-a-0}, x_{10}\text{-a-0}\} <_t f\text{-a-1} = x_8\text{-a-1} \\ x_1\text{-a-1} &<_t \{x_3\text{-a-0}, x_4\text{-a-0}\} <_t x_7\text{-a-1} <_t \{x_9\text{-a-0}, x_{10}\text{-a-0}\} <_t f\text{-a-1} = x_1\text{-a-1} \\ x_2\text{-a-1} &<_t \{x_3\text{-a-0}, x_4\text{-a-0}\} <_t x_7\text{-a-1} <_t \{x_9\text{-a-0}, x_{10}\text{-a-0}\} <_t f\text{-a-1} = x_2\text{-a-1} \end{aligned}$$

The minimum set of faults sufficient to derive a set of tests that completely detect all possible faults in the circuits is $F = \{x_5\text{-a-0}, x_1\text{-a-0}, x_4\text{-a-1}, x_5\text{-a-1}, x_6\text{-a-1}, x_8\text{-a-1}, x_1\text{-a-1}, x_2\text{-a-1}\}$. It is to be noted that only one fault has been chosen from the each of the equivalence classes $\{x_1\text{-a-0}, x_2\text{-a-0}\}$ and $\{x_5\text{-a-0}, x_6\text{-a-0}\}$ as per Lemma 1. The faults $x_2\text{-a-0}$ and $x_6\text{-a-0}$ can also be chosen instead of $x_1\text{-a-0}$ and $x_5\text{-a-0}$ respectively.

It is to be observed that the fault-folding graphs for f -a-0 and f -a-1 are always duals (except for buffers and inverters). In other words, a test implication between two nodes

in one fault folding graph becomes test equivalence in the other and vice-versa. Hence for information representation only one graph is necessary, the other graph can be drawn without any further calculation.

Fault Folding results in a minimal set of faults but it does not yield a minimal test set. In fact To[3] himself states "Note that although the minimal set of faults (F) is sufficient to derive the complete set of detection tests (CSDT's); the number of tests in the CSDT is equal to or smaller than the number of faults in the minimal set F".

4.1 Minimal Test Set for Non-Reconvergent Fanout Circuits

4.1.1 Fault Folding Method

Consider an equivalence class $\{x_{1-a-i_1}, x_{2-a-i_2}, \dots, x_{n-a-i_n}\}$ where x_1, x_2, \dots, x_n are the circuit nodes (input or internal) and $\{i_1, i_2, i_3, \dots, i_n\} \in \{0,1\}$. Then three different cases could exist:-

1. All the nodes $\{x_1, \dots, x_n\}$ are primary input nodes.
2. Some of the nodes are internal nodes and the rest are primary input nodes.
3. All the nodes $\{x_1, \dots, x_n\}$ are internal nodes.

Case-1:-

If all the nodes of the Equivalence class $\{x_1, \dots, x_n\}$ are primary input nodes.

To [3] has proven that it is enough to consider one member from the equivalence class to determine the minimal set of faults and hence the minimal test set.

Case-2:-

Some of the nodes of the equivalence class $\{x_1, x_2, x_3, \dots, x_n\}$ are primary input nodes.

Theorem 3 *The primary input faults could be neglected (for the purposes of finding minimal test set) in an equivalence class containing primary inputs and internal nodes.*

Proof:-

This theorem follows from the definition of the equivalent fault class. By definition, it is enough to test one member of the equivalent fault class to test the whole fault class. By the Theorem's initial assumption, there exists at least one non-primary node in the equivalence class. Since all non-primary nodes of the equivalent fault class are being tested, it follows that the entire fault class is tested. Hence it is enough to consider only the internal nodes of the equivalence class.

Case-3:- *All the nodes of the equivalent fault class $\{x_{1-a-i_1}, x_{2-a-i_2}, \dots, x_{n-a-i_n}\}$ are internal nodes.*

Theorem 4 *If m_1 primary input node faults test cover an internal node fault x_{1-a-i_1} , m_2 primary input node faults test cover an internal node fault x_{2-a-i_2} ,*

⋮

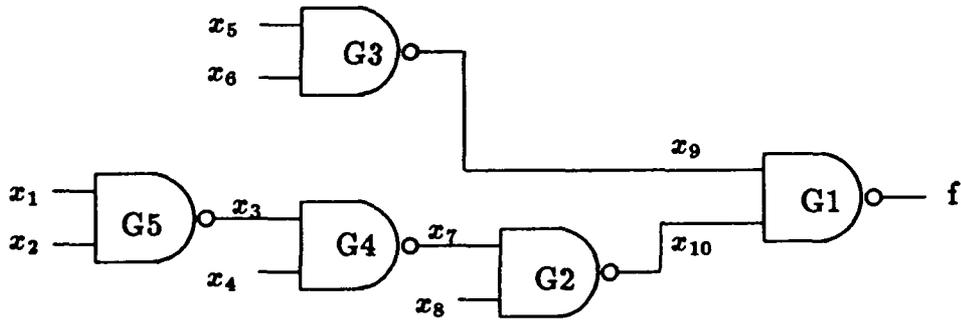
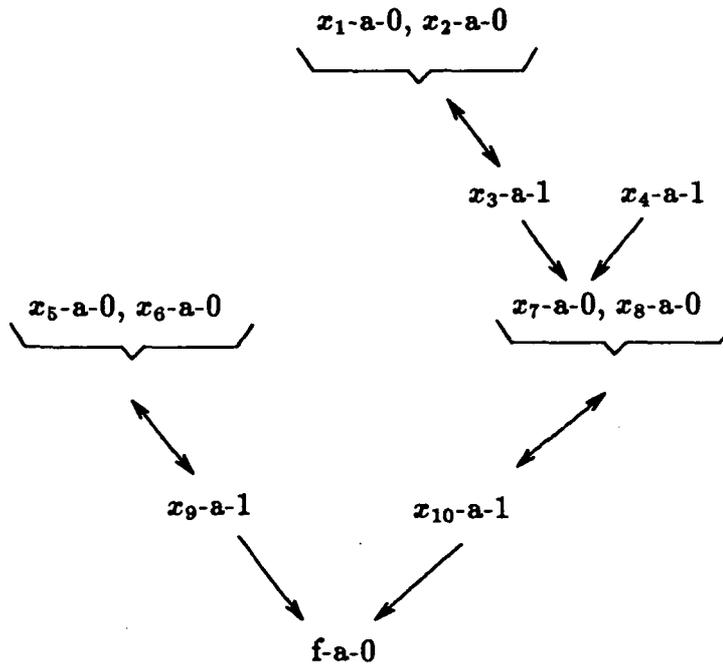


Figure 5: A fanout free circuit

Figure 6: Fault folding graph for $f-a-0$

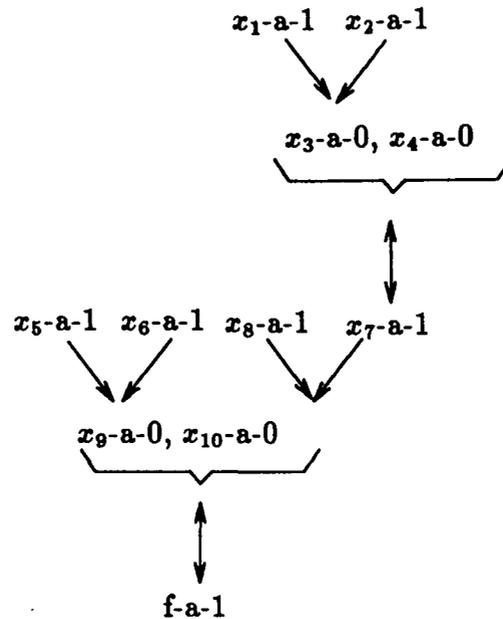


Figure 7: Fault folding graph for f-a-1

and m_n primary input node faults test cover an internal node fault x_n-a-i_n , and if $\{x_1-a-i_1, x_2-a-i_2, \dots, x_n-a-i_n\}$ form an equivalent fault class, then a test vector can be found which would test, any one of the m_1 , any one of the m_2, \dots , any one of the m_n faults, simultaneously.

Proof:-

Consider the equivalence fault class $\{x_1-a-i_1, x_2-a-i_2, \dots, x_n-a-i_n\}$. Let x_1-a-i_1 be test covered by the primary input faults $\{y_{11-a-p_{11}}, y_{12-a-p_{12}}, \dots, y_{1m-a-p_{1m}}\}$ and x_2-a-i_2 be test covered by the primary input faults $\{y_{21-a-p_{21}}, y_{22-a-p_{22}}, \dots, y_{2m-a-p_{2m}}\}$.

Let T_1 be the set of test vectors which tests the fault set $\{y_{11-a-p_{11}}, y_{12-a-p_{12}}, \dots, y_{1m-a-p_{1m}}\}$ and let T_2 test the fault set $\{y_{21-a-p_{21}}, y_{22-a-p_{22}}, \dots, y_{2m-a-p_{2m}}\}$. By the definition of test cover, T_1 and T_2 also test the faults x_1-a-i_1 and x_2-a-i_2 . Since x_1-a-i_1 and x_2-a-i_2 are in the same equivalent fault class, $T_1 = T_2$. This means that any test vector $t_1 \in T_1$ will test one member of the fault class $F_1 = \{y_{11-a-p_{11}}, y_{12-a-p_{12}}, \dots, y_{1m-a-p_{1m}}\}$, and one member of $F_2 = \{y_{21-a-p_{21}}, y_{22-a-p_{22}}, \dots, y_{2m-a-p_{2m}}\}$ simultaneously.

For NRFO circuits, it so happens that a member of the fault class F_1 could be tested with any one member of F_2 simultaneously.

Corollary 1 For the case described in Theorem 4, only m_p faults need to be tested where $m_p \in \{m_1, m_2, \dots, m_n\}$ such that $m_p > m_j |_{j \neq p}$ and $m_j \in \{m_1, m_2, \dots, m_n\}$

Example 2 Find the minimal Test set for the circuit shown in Figure 5.

The fault folding graphs of the Figures 6, 7 are first constructed. By To's method[3] the minimal fault set is found to be,

$$\left\{ \left(\begin{array}{c} x_5-a-0 \\ x_6-a-0 \end{array} \right), \left(\begin{array}{c} x_1-a-0 \\ x_2-a-0 \end{array} \right), x_4-a-1, x_8-a-0, x_5-a-1, x_6-a-1, x_8-a-1, x_1-a-1, x_2-a-1, x_4-a-0 \right\}$$

By Theorem 3, the faults x_4-a-0 and x_8-a-0 can be neglected, since these faults will be tested by any test for the faults x_3-a-0 and x_7-a-0 respectively. Applying Theorem 4, one finds that the fault x_5-a-1 can be *conjoined* with either x_8-a-1 or x_1-a-1 or x_2-a-1 ; and x_6-a-1 can be *conjoined* with x_8-a-1 or x_1-a-1 or x_2-a-1 . If x_5-a-1 and x_8-a-1 are conjoined as are x_6-a-1 and x_1-a-1 , the following multiple fault set is obtained;

$$F_{m1} = \left\{ \left(\begin{array}{c} x_5-a-0 \\ x_6-a-0 \end{array} \right) \left(\begin{array}{c} x_1-a-0 \\ x_2-a-0 \\ x_8-a-0 \end{array} \right) \left(\begin{array}{c} x_4-a-1 \\ x_8-a-0 \end{array} \right) \left(\begin{array}{c} x_5-a-1 \\ x_8-a-1 \end{array} \right) \left(\begin{array}{c} x_6-a-1 \\ x_1-a-1 \\ x_4-a-0 \end{array} \right) \right. \\ \left. \left(\begin{array}{cc} x_2-a-1 & x_2-a-1 \\ x_4-a-0 & \text{or } x_4-a-0 \\ x_5-a-1 & x_6-a-1 \end{array} \right) \right\}.$$

Definition 1 Multiple fault set is a set whose members include multiple faults.

As an example, the fault set F_{m1} obtained in Example 2 is a multiple fault set whose members are all multiple faults.

Since there are three choices each for conjoining or overlapping the faults x_5-a-1 and x_6-a-1 with other faults, we could obtain different multiple fault sets of the same length which test cover all the single faults. The fault set enclosed by the parenthesis {...} represents the multiple fault set, whose members are multiple faults. Each multiple fault is enclosed by the parenthesis (...). The next step is to find the tests for each of the multiple faults. Each multiple fault might have more than one test vector. In such cases, all the possible test vectors testing a single multiple fault are enclosed within the parenthesis (...) with the agreement that only one test vector is to be chosen among them. The minimal test set for F_{m1} is given by:-

$$T_{m1} = \langle x_1 x_2 x_4 x_5 x_6 x_8 \rangle =$$

$$\left\{ \left(\begin{array}{ccc} - & - & -110 \\ -01111 \\ 0 & - & 1111 \end{array} \right) \left(\begin{array}{ccc} 1110 & - & 1 \\ 111 & - & 01 \end{array} \right) \left(\begin{array}{ccc} 0 & - & 00 & - & 1 \\ - & 00 & - & 01 \end{array} \right) \left(\begin{array}{ccc} - & - & 0010 \\ 111010 \end{array} \right) \left(\begin{array}{ccc} 011101 \end{array} \right) \left(\begin{array}{ccc} 101011 \\ 101101 \end{array} \right) \right\}$$

It is again reiterated that only one test from each of the test vector sets enclosed within the parenthesis (...) is to be chosen to form a minimal test set. In the above example, the length of the minimal test set is found to be six.

4.1.2 Method by Inspection

It is *ultimately* desirable that the minimal length multiple fault set covering all single faults be found, by direct inspection of the network, without even drawing the fault folding

graphs. From our observations, by working out a plethora of examples certain conjectures are made (to follow) which eases one's task.

Consider a NRFO circuit with m different paths. This implies that there are m primary gates. A primary gate is a gate whose inputs are all primary inputs. Each gate is assigned a level (in an obvious manner) starting with with level 2 at the output and working backwards. Then,

Theorem 5 *It is only necessary to find s-a-1(0) tests on all primary inputs of the non-primary gates and both s-a-0 and s-a-1 tests on all the input leads of the primary gates in an NRFO NAND-NAND(NOR-NOR) circuit.*

Proof:-

The primary gates may be at odd or even levels. It has been proven [12] for a NAND-NAND circuit that a s-a-0 on an input of an odd(even) primary gate will also be a s-a-0 test for the leads of all the odd(even) gates in that path to the output. Also, a s-a-1 test on an input of an odd(even) primary gate will also be a s-a-0 test for the leads of all the even(odd) gates in that path to the output. Hence if the primary gates(odd or even) are tested for both s-a-0 and s-a-1 faults, all the gates in the path from that primary gate to the output will be tested for s-a-0 faults.

So if all the primary gates are tested for both s-a-0 and s-a-1 faults, all possible s-a-0 faults in the circuit will be checked. It is only needed to test the s-a-1 faults on the remaining primary inputs to ensure that all the primary inputs are checked for both s-a-0 and s-a-1 faults. To[3] has stated that only the primary inputs in an NRFO circuit has to be tested for s-a-0 and s-a-1 faults to completely test the circuit. Since this objective has been accomplished, the entire circuit will have been tested completely. Hence the theorem is verified.

At this point, a conjecture is made to help find directly the multiple fault set which yields the minimal test vectors.

Conjecture 1 *In an NRFO NAND-NAND circuit, the s-a-1(0) fault on a primary input of an even(odd) level gate in a path P_i could be conjoined with one of the following faults in each path $P_j |_{j \neq i}$.*

1. s-a-1 fault on a primary input of an even level gate.
2. s-a-0 fault on a primary input of an odd level gate.

Example 3 .

Consider the circuit in Figure 5. According to Theorem 5, one needs to consider only the primary gate faults $\{x_1-a-0, x_1-a-1, x_2-a-1, x_5-a-0, x_6-a-1, x_5-a-1\}$ and the primary input faults $\{x_4-a-1, x_8-a-1\}$. Gates G_3, G_2, G_5 are even level gates where as G_4 and G_1 are odd level gates. So by Conjecture 1, each member of the fault set $\{x_5-a-1, x_6-a-1\}$ (even level gate) can be conjoined with either one of the following faults:-

1. x_1-a-1 or x_2-a-1 (even gate)

2. $x_4\text{-a-0}$ (odd gate)

3. $x_8\text{-a-1}$ (even gate)

The test for $x_4\text{-a-0}$ is not needed as per Theorem 5. If we conjoin $x_5\text{-a-1}$ with $x_1\text{-a-1}$ and $x_6\text{-a-1}$ with $x_8\text{-a-1}$, the following reduced fault set is obtained

$$F = \left\{ x_1\text{-a-0}, x_2\text{-a-1}, x_4\text{-a-1}, x_5\text{-a-0}, \begin{bmatrix} x_1\text{-a-1} \\ x_5\text{-a-1} \end{bmatrix}, \begin{bmatrix} x_6\text{-a-1} \\ x_8\text{-a-1} \end{bmatrix} \right\}$$

where $\{x_1\text{-a-1}, x_5\text{-a-1}\}$ and $\{x_6\text{-a-1}, x_8\text{-a-1}\}$ are multiple faults. This fault set could have been obtained by a different choice of conjoining test vectors in Example 2. Obtaining test vectors for this fault set yields a minimal length test vector set. Similar Theorems and Conjectures can be derived for NOR-NOR circuits.

4.2 Minimal Test Set for Internal Reconvergent Fanout Circuits

4.2.1 Fault Folding Method

To[3] has stated and proven the following theorem.

Theorem 6 *Any set of tests which checks all the faults at the primary inputs and fanout branches of a RFO irredundant circuit is sufficient to check the entire circuit.*

To [3] implies that one has to test all the fanout branches in addition to primary inputs. But the faults in the fanout branches can always be folded back to faults on the primary inputs. So for the purpose of finding a minimal set of test vectors, the primary input fault and the fault on the fanout branch which it test covers, are conjoined to form a multiple fault, so that a single test will test both these essential faults. The primary input fault, covering the fault on the fanout branch, is found using the fault folding graphs.

Consider the circuit shown in Figure 8. There are two fanout branches: $I_{5.1}$ and $I_{5.2}$. According to the Theorem 6, the s-a-0 and s-a-1 faults on $I_{5.1}$ and $I_{5.2}$ are essential faults. The tests for s-a-1 faults on x_2 and x_3 propagated through the lines $I_{5.1}$ and $I_{5.2}$ respectively will test cover the s-a-0 faults on the lines $I_{5.1}$ and $I_{5.2}$.

Theorems 3, 4, 5 developed for NRFO circuits can also be used here. But one has to be more careful when dealing with reconvergent fanout circuits. There are three *golden rules* which if followed, might make one's task less arduous.

Rule-1:- *Be sure that the multiple fault set obtained using Theorems 3 and 4 contain the s-a-0 and s-a-1 faults on all the fanout branches.*

Illustration:- Consider the fault folding graphs in the Figures 9 and 10 for the circuit in the Figure 8. If the faults on the fanout branches are not treated as essential faults, then using Theorems 3 and 4 one might get a multiple fault set

$$F_m = \left\{ \begin{bmatrix} x_2\text{-a-1} \\ x_4\text{-a-1} \end{bmatrix}, \begin{bmatrix} x_3\text{-a-1} \\ x_4\text{-a-1} \end{bmatrix}, \begin{bmatrix} x_2\text{-a-1} \\ x_1\text{-a-0} \end{bmatrix}, \begin{bmatrix} x_3\text{-a-1} \\ x_1\text{-a-0} \end{bmatrix}, \begin{bmatrix} x_1\text{-a-1} \\ x_4\text{-a-0} \end{bmatrix}, \begin{bmatrix} x_2\text{-a-0} \\ x_3\text{-a-0} \end{bmatrix} \right\}$$

From Figure 9, it is found that the equivalence fault class $\{x_2\text{-a-0}, x_3\text{-a-0}\}$ test covers

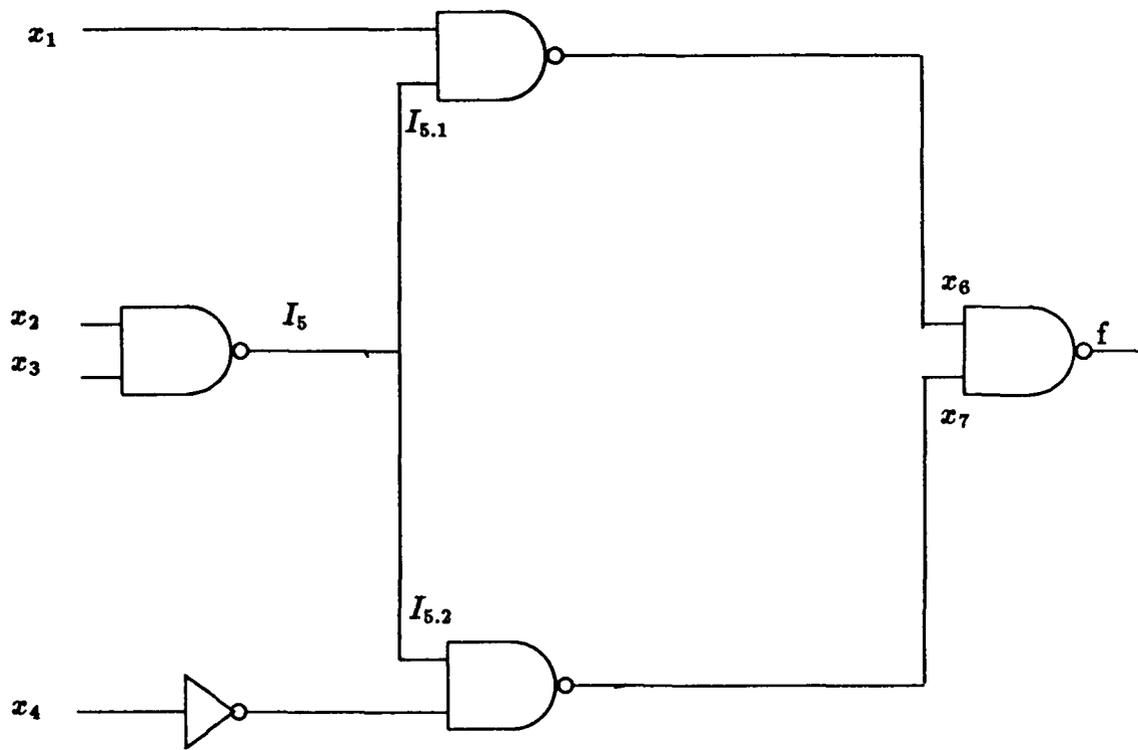


Figure 8: An Internal Reconvergent fanout circuit .

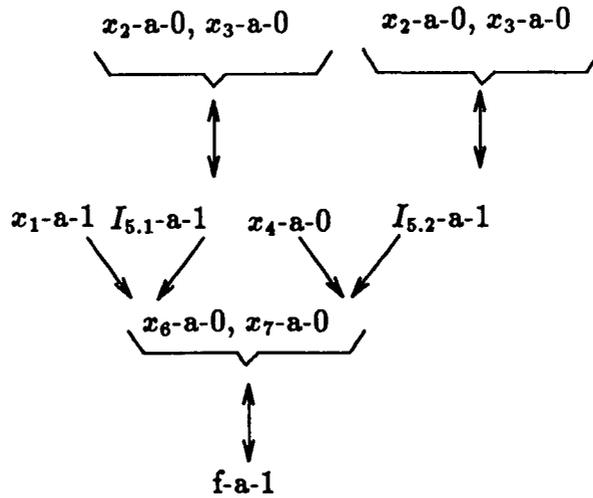


Figure 9: Fault folding graph for f-a-1

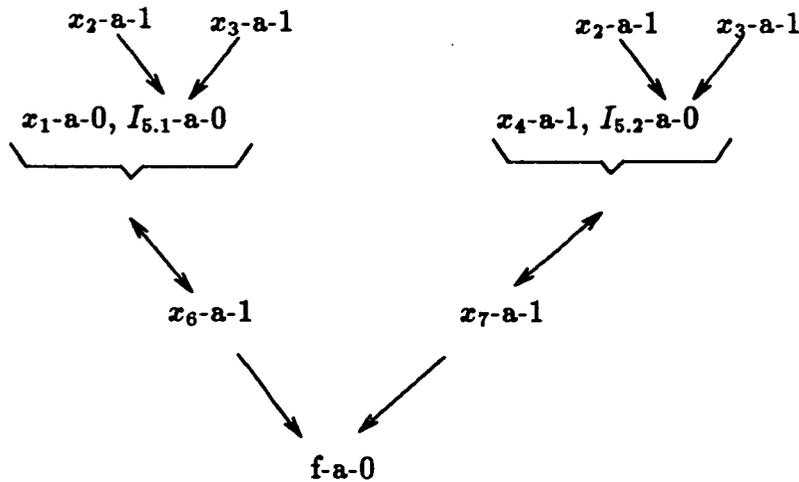


Figure 10: Fault folding graph for f-a-0

both $I_{5.1-a-1}$ and $I_{5.2-a-1}$. Using Theorems 3 and 4, which say only primary input faults can be conjoined and hence tested, the faults x_1-a-1 and x_4-a-0 have been conjoined as are $\{x_2-a-0, x_3-a-0\}$ and $\{x_2-a-0, x_3-a-0\}$ to get the multiple faults $\{x_1-a-1, x_4-a-0\}$ and $\{x_2-a-0, x_3-a-0\}$ respectively.

The first and second, third and fourth multiple faults in set F_m will test cover the faults $I_{5.1}$ and $I_{5.2}$ stuck-at-0 respectively. There are three choices of test vectors to test the last multiple fault in F_m which is $\{x_2-a-0, x_3-a-0\}$. The test vectors are $\langle x_1x_2x_3x_4 \rangle = \{0110, 1111, 1110\}$. The test vector $\langle x_1x_2x_3x_4 \rangle = 0110$ propagates the multiple fault $\{x_2-a-0, x_3-a-0\}$ through the path $I_{5.2}$ and in the process also tests the fault $I_{5.2-a-1}$. Similarly the test vector $\langle x_1\dots x_4 \rangle = 1111$ propagates the multiple fault $\{x_2-a-0, x_3-a-0\}$ through $I_{5.1}$ and also happens to test $I_{5.1-a-1}$. But the test vector $\langle x_1x_2x_3x_4 \rangle = 1110$, not only test the multiple fault $\{x_2-a-0, x_3-a-0\}$ but also the faults $I_{5.1-a-1}$ and $I_{5.2-a-1}$. According to Theorem 6, the s-a-0 and s-a-1 faults on the fanout branches have to be tested as they are essential faults. So if one chooses $\langle x_1x_2x_3x_4 \rangle = \{0110$ or $1111\}$ as the test vector to test the multiple fault $\{x_2-a-0, x_3-a-0\}$, the s-a-1 faults on only one of the fanout branches instead of both the fanout branches will be tested. Since all the essential faults according the Theorem 6 are not covered by the test set, this test set is incomplete. Hence the test vector $\langle x_1x_2x_3x_4 \rangle = 1110$ is to be chosen to test the multiple fault $\{x_2-a-0, x_3-a-0\}$ in order to get a valid minimal complete test set.

In order to avoid the problem of a potential incomplete test set, the faults on the fanout branches must be included in addition to the primary input faults which test cover them (faults on fanout branches), while specifying the multiple fault in the minimal multiple fault set.

Therefore in the current example, applying Theorems 3 and 4, the faults on the fanout branches are also specified in the multiple fault. From Figure 9, the multiple fault $\{x_2-a-0, x_3-a-0, I_{5.1-a-1}\}$ is conjoined with the multiple fault $\{x_2-a-0, x_3-a-0, I_{5.2-a-1}\}$ to form the

multiple fault $\{x_{2-a-0}, x_{3-a-0}, I_{5.1-a-1}, I_{5.2-a-1}\}$. So the minimal multiple fault set F_m is now modified as

$$F_m = \left\{ \left[\begin{array}{c} x_{2-a-1} \\ x_{4-a-1} \end{array} \right], \left[\begin{array}{c} x_{3-a-1} \\ x_{4-a-1} \end{array} \right], \left[\begin{array}{c} x_{2-a-1} \\ x_{1-a-0} \end{array} \right], \left[\begin{array}{c} x_{3-a-1} \\ x_{1-a-0} \end{array} \right], \left[\begin{array}{c} x_{1-a-1} \\ x_{4-a-0} \end{array} \right], \left[\begin{array}{c} x_{2-a-0} \\ x_{3-a-0} \\ I_{5.1-a-1} \\ I_{5.2-a-1} \end{array} \right] \right\}$$

The test vector $\langle x_1 x_2 x_3 x_4 \rangle = 1110$ is the only test for the multiple fault $\{x_{2-a-0}, x_{3-a-0}, I_{5.1-a-1}, I_{5.2-a-1}\}$ which agrees with the earlier finding that only the vector 1110 is to be chosen to form a complete test set.

There is still some redundancy in F_m , that is F_m is still not a truly minimal multiple fault set. The method of removing this redundancy will be explained further on.

Definition 2 *Faults are said to be contradictory if they cannot be tested simultaneously.*

The faults x_{j-a-i} and $x_{j-a-\bar{i}}$ are complementary faults as they can never be tested simultaneously. Also for any gate (NAND or NOR except XOR) checking one of the inputs for a s-a-0 fault and the other(s) input(s) for s-a-1 fault is impossible. Additionally testing one input for s-a-1(0) fault and the other input(s) of the same gate for s-a-1(0) fault in the case of NAND (NOR) gate is impossible. Hence such faults are said to be contradictory faults.

Rule-2:- *Do not conjoin contradictory faults while using Theorem 4.*

Theorem 4 says that two or more primary input faults can be conjoined if they test cover faults which are in the same equivalence class. This might not always be true in the case of RFO circuits. Hence the theorem is modified as follows:-

Theorem 7 *If m_1 primary input node faults test cover an internal node fault x_{1-a-i_1} , m_2 primary input node faults test cover an internal node fault x_{2-a-i_2} ,*

⋮

and m_n primary input node faults test cover an internal node fault x_{n-a-i_n} ,

and if $\{x_{1-a-i_1}, x_{2-a-i_2}, \dots, x_{n-a-i_n}\}$ form an equivalent fault class;

then a test vector can be found which would test any one of the m_1 , any one of the m_2 , ..., any one of the m_n faults simultaneously, "provided that these faults are not contradictory".

One comes across such instances of contradictory faults when dealing with RFO circuits, especially where the inversion parity along the reconverging paths is not the same. The inversion parity of a reconverging path is defined to be the number of inversions, modulo 2, along the path between the specified fanout node and the node of reconvergence.

Illustration 2:- Consider the circuit in Figure 8. There is no difference in the Inversion parities along the reconverging paths $I_{5.1}$ and $I_{5.2}$. In the fault folding graph of Figure 9, the multiple fault $\{x_{2-a-0}, x_{3-a-0}, I_{5.1-a-1}\}$ and the fault x_{4-a-0} cannot be conjoined together though they imply the faults x_{6-a-0}, x_{7-a-0} which are in the same equivalence class. This is because the provoking conditions for the faults $I_{5.1-a-1}$ and $I_{5.2-a-1}$ are the same. It has already been noted before, that for a NAND gate testing both inputs for s-a-1

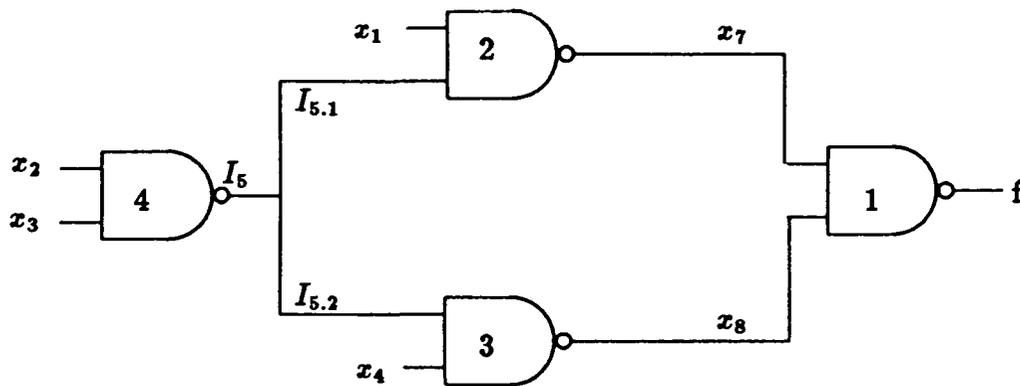


Figure 11: An internal RFO circuit

faults simultaneously is impossible. In the present case, the fault x_4 -a-0 if combined with $\{x_2$ -a-0, x_3 -a-0, $I_{5.1}$ -a-1 $\}$ will imply that both the faults x_4 -a-0 and $I_{5.2}$ -a-1 will be tested (provoking conditions of $I_{5.1}$, $I_{5.2}$ are the same) which is not possible as both $I_{5.2}$ and x_4 are inputs of the same NAND gate and hence are contradictory faults. So the only way to conjoin faults from the fault folding graph for f-a-1 is,

$$\left\{ \left[\begin{array}{c} x_2\text{-a-0} \\ x_3\text{-a-0} \\ I_{5.1}\text{-a-1} \\ I_{5.2}\text{-a-1} \end{array} \right] \left[\begin{array}{c} x_1\text{-a-1} \\ x_4\text{-a-0} \end{array} \right] \right\}.$$

Rule-3:- Try not to cover the same fault twice.

If one is not careful, redundant tests will be incorporated in the minimal test vector set. This usually happens in RFO circuits where the same fault on the primary input might occur more than once in the fault folding graphs.

Conjecture 2 If two primary input faults x_a -a- i_1 and x_b -a- i_2 , implying a fault on the same internal node, appears more than once in the fault folding graphs; then it is enough to consider the fault x_a -a- i_1 in one of the appearances and x_b -a- i_2 in the other appearance.

Using this conjecture for the circuit in the Figure 8, we find that the implying primary input fault class x_2 -a-0 and x_3 -a-0 appear twice in the fault folding graph f-a-0. So in the first appearance (implying $I_{5.1}$ -a-0) consider x_2 -a-0 and neglect x_3 -a-0. In the second appearance (implying $I_{5.2}$ -a-0) consider x_3 -a-0 and neglect x_2 -a-0, while forming the multiple fault sets.

The faults x_3 -a-0 and x_2 -a-0 could also have been chosen (neglecting x_2 -a-0 and x_3 -a-0) in the first and second appearances of the implying class respectively. This would also have resulted in a different minimal multiple fault set albeit of the same length.

Example 4 For the circuit in Figure 11, find the minimal test vector set.

Step 1:- The fault folding graph f-a-0 or f-a-1 is first drawn and then its dual is constructed.

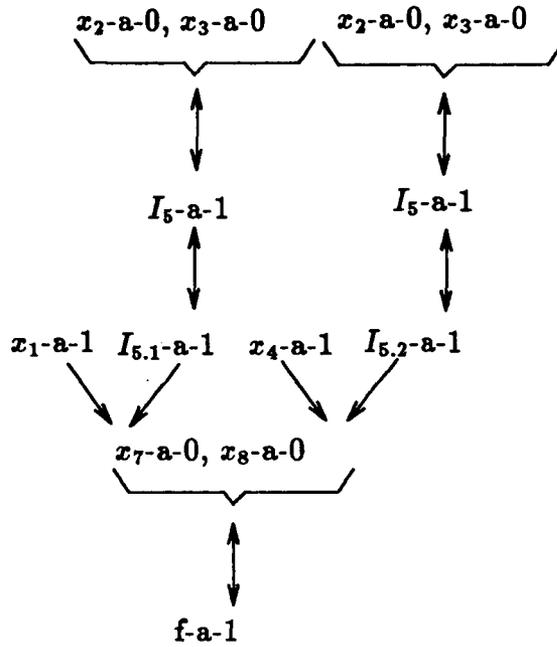


Figure 12: Fault folding graph for f-a-1

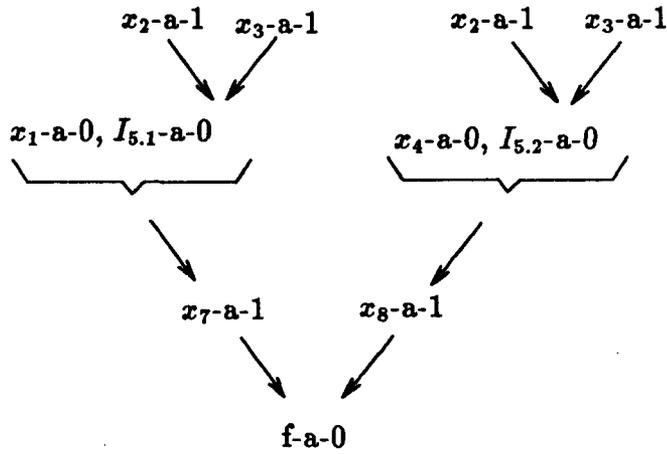


Figure 13: Fault folding graph for f-a-0

Step 2:- In the fault folding graph of f -a-1 (Figure 12), the stuck-at fault x_1 -a-1 is conjoined with x_4 -a-1 using Theorem 7. If x_1 -a-1 and $\{x_2$ -a-0, x_3 -a-0, $I_{5.2}$ -a-1 $\}$ are conjoined it effectively means that the faults x_1 -a-1 and $I_{5.1}$ -a-1 are being tested simultaneously because the provoking conditions of the faults $I_{5.1}$ -a-1 and $I_{5.2}$ -a-1 are the same. Since, both x_1 and $I_{5.1}$ are the inputs of the same NAND gate (gate 2) the faults x_1 -a-1 and $\{x_2$ -a-0, x_3 -a-0, $I_{5.2}$ -a-1 $\}$ are contradictory faults.

Applying Theorem 7 to the Figure 12, the faults $\{x_2$ -a-0, x_3 -a-0, $I_{5.1}$ -a-1 $\}$ and $\{x_2$ -a-0, x_3 -a-0, $I_{5.2}$ -a-1 $\}$ are conjoined to form the multiple fault $\{x_2$ -a-0, x_3 -a-0, $I_{5.1}$ -a-1, $I_{5.2}$ -a-1 $\}$. Note that though $\{x_2$ -a-0, x_3 -a-0 $\}$ occurs twice, they test cover two different faults namely $I_{5.1}$ -a-1 and $I_{5.2}$ -a-1. So the test vectors are obviously different. The test vector for the multiple fault $\{x_2$ -a-0, x_3 -a-0, $I_{5.1}$ -a-1 $\}$ is given by the $\langle x_1, x_2, x_3, x_4 \rangle = \langle 1111 \rangle$, where as the test vector for the multiple fault $\{x_2$ -a-0, x_3 -a-0, $I_{5.2}$ -a-1 $\}$ is given by the $\langle x_1, x_2, x_3, x_4 \rangle = \langle 0110 \rangle$. The test vector for the multiple faults $\{x_2$ -a-0, x_3 -a-0, $I_{5.1}$ -a-1, $I_{5.2}$ -a-1 $\}$ is 1110.

From the fault folding graph of f -a-0 (Figure 13), it is observed that two faults x_2 -a-1 and x_3 -a-1 implying the same fault I_5 -a-0 appear twice. Using Conjecture 2, one has the choice of multiple faults as

$$\left\{ \begin{bmatrix} x_2\text{-a-1} \\ x_1\text{-a-0} \\ I_{5.1}\text{-a-0} \end{bmatrix} \begin{bmatrix} x_3\text{-a-1} \\ x_4\text{-a-0} \\ I_{5.2}\text{-a-0} \end{bmatrix} \right\}$$

or

$$\left\{ \begin{bmatrix} x_3\text{-a-1} \\ x_1\text{-a-0} \\ I_{5.1}\text{-a-0} \end{bmatrix} \begin{bmatrix} x_2\text{-a-1} \\ x_4\text{-a-0} \\ I_{5.2}\text{-a-0} \end{bmatrix} \right\}$$

So concatenating the multiple faults from both the fault folding graphs, the minimal length multiple fault set is obtained as

$$F_{m1} = \left\{ \begin{bmatrix} x_4\text{-a-1} \\ x_1\text{-a-1} \end{bmatrix} \begin{bmatrix} x_2\text{-a-0} \\ x_3\text{-a-0} \\ I_{5.1}\text{-a-1} \\ I_{5.2}\text{-a-1} \end{bmatrix} \begin{bmatrix} x_2\text{-a-1} \\ x_1\text{-a-0} \\ I_{5.1}\text{-a-0} \end{bmatrix} \begin{bmatrix} x_4\text{-a-0} \\ x_3\text{-a-1} \\ I_{5.2}\text{-a-0} \end{bmatrix} \right\}$$

The other minimal length multiple fault set is

$$F_{m2} = \left\{ \begin{bmatrix} x_4\text{-a-1} \\ x_1\text{-a-1} \end{bmatrix} \begin{bmatrix} x_2\text{-a-0} \\ x_3\text{-a-0} \\ I_{5.1}\text{-a-1} \\ I_{5.2}\text{-a-1} \end{bmatrix} \begin{bmatrix} x_3\text{-a-1} \\ x_1\text{-a-0} \\ I_{5.1}\text{-a-0} \end{bmatrix} \begin{bmatrix} x_4\text{-a-0} \\ x_2\text{-a-1} \\ I_{5.2}\text{-a-0} \end{bmatrix} \right\}$$

It is to be noted that the fault sets F_{m1} and F_{m2} obey Rule 1, as both s-a-0 and s-a-1 faults on all primary inputs and fanout branches are covered by the multiple faults. Rule 2 is also obeyed by ensuring that contradictory faults are not conjoined together. Rule 3 is also obeyed since we have used Conjecture 2 to form the multiple faults.

Step 3:- The set of test vectors T_{m1} and T_{m2} are found for the multiple fault sets F_{m1} and F_{m2} respectively by path sensitization.

$$T_{m1} = \langle x_1, x_2, x_3, x_4 \rangle = \{ [00-0 \text{ or } 0-00], [1111], [1010], [0101] \}$$

$$T_{m2} = \langle x_1, x_2, x_3, x_4 \rangle = \{ [00-0 \text{ or } 0-00], [1111], [1100], [0011] \}$$

Either of the test vector sets T_{m1} and T_{m2} could be used to test the circuit in a truly minimal fashion. It is to be borne in mind that only one test vector is to be selected from the vectors enclosed by the square parenthesis—[], since they represent the different choices of test vectors for the same multiple fault.

4.2.2 Minimal Test vector set for IRFO circuits by Inspection

The Conjecture 1 developed for NRFO circuits could be used here with a small modification. The earlier conjecture was applicable only for primary inputs. In IRFO circuits, the faults on the fanout branches are also essential faults. So the Conjecture 1 is modified to include both primary inputs and fanout branches in the case of IRFO circuits. In addition, the following conjectures are to be used when dealing with IRFO circuits.

Conjecture 3 *For internal RFO paths in a NAND-NAND circuits with the same parities of inversion,*

1. *The s-a-1(0) faults on the fanout branches must be conjoined together if the gates, to which the fanout branches feed, are even(odd).*
2. *The s-a-i faults on the fanout branches can be conjoined with s-a- \bar{i} faults on the inputs of the gate whose output fans out.*

Conjecture 4 *For internal RFO paths in a NAND-NAND circuits with different parities of inversion,*

1. *The stuck-at faults on the fanout branches must not be conjoined together.*
2. *The s-a-i faults on the fanout branches can be conjoined with s-a- \bar{i} faults on the inputs of the gate whose output fans out.*

Conjecture 5 *The Clause 1 of Conjectures 3 and 4 could also be restated as follows:-*

For NAND-NAND IRFO circuits, the s-a-1(0) faults on the fanout origin is propagated simultaneously through all the fanout paths with even(odd) parities of inversion, if the reconvergent paths have the same parities of inversion.

For NAND-NAND IRFO circuits, the s-a-i (i=1 or 0) faults on the fanout origin should not be propagated simultaneously through the fanout paths which have the different parities of inversion.

Using the above conjectures, the Example 4 will be reworked by inspection.

Example 5 *Find a minimal set of test vectors for the circuit in Figure 11 by Inspection.*

Gates 1 and 4 are odd level gates and gates 2 and 3 are even level gates. It is known that the s-a-0 and s-a-1 faults on all the inputs of the primary gates and s-a-1 faults on the primary inputs of other gates only need to be checked. Since the circuit in question is a RFO circuit, the faults on the fanout branches are also to be checked. From the

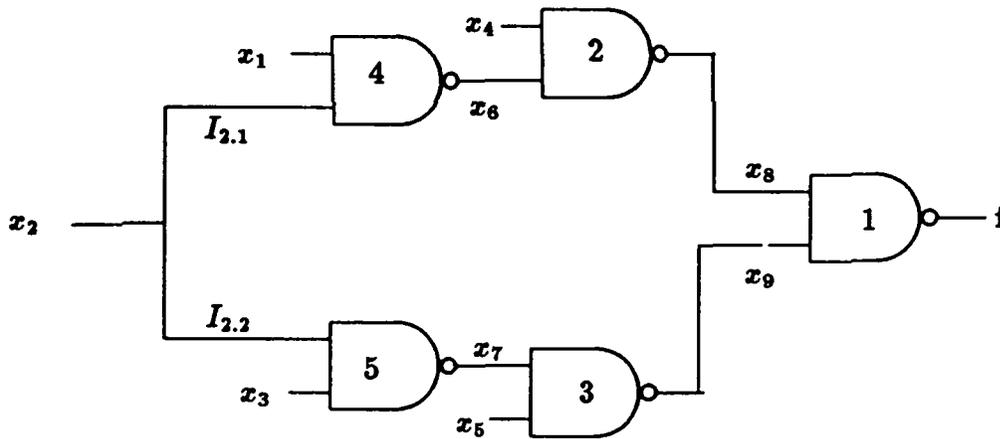


Figure 14: A Primary input fanout Circuit

circuit in the Figure 11, it is observed that the inversion parities along the reconvergent fanout paths are the same. So using Conjecture 3, the s-a-1 fault on line $I_{5.1}$ (Gate 2 is even) is conjoined with s-a-1 fault on line $I_{5.2}$ (Gate 3 is even). By the second clause of Conjecture 3, the s-a-1 faults on $I_{5.1}$ and $I_{5.2}$ are conjoined with s-a-0 faults on x_2 and x_3 . It is to be noted that x_2 -a-0 and x_3 -a-0 are equivalent faults. At this point, the s-a-0 faults on the fanout paths remain to be tested. As the gates 2 and 3 are at even levels, they cannot be conjoined together according to Conjecture 1. Therefore using Clause 2 of the Conjecture 3, the s-a-0 fault on $I_{5.1}$ is conjoined with the s-a-1 fault on either x_2 or x_3 . Similarly the s-a-0 fault on $I_{5.2}$ is conjoined with the s-a-1 faults on either x_2 or x_3 . For minimality purposes, if $I_{5.1}$ -a-0 is conjoined with x_2 -a-1, then $I_{5.2}$ -a-0 is conjoined with x_3 -a-1 or vice-versa. Lastly, the s-a-1 faults on line x_1 is conjoined with s-a-1 fault on line x_4 , using Conjecture 1. So the minimal multiple fault set is given by

$$F_m = \left\{ \left[\begin{array}{c} x_1\text{-a-1} \\ x_4\text{-a-1} \end{array} \right] \left[\begin{array}{c} x_3\text{-a-0} \\ x_2\text{-a-0} \\ I_{5.1}\text{-a-1} \\ I_{5.2}\text{-a-1} \end{array} \right] \left[\begin{array}{c} x_2\text{-a-1} \\ I_{5.1}\text{-a-0} \end{array} \right] \left[\begin{array}{c} x_3\text{-a-1} \\ I_{5.2}\text{-a-0} \end{array} \right] \right\}$$

This multiple fault set F_m is the same as F_{m1} obtained in the Example 4. Using similar arguments it is possible to derive F_{m2} .

4.3 Minimal Test set for Primary Input Fanout Circuits

4.3.1 Fault Folding

The primary input fanout circuits are also treated like the internal RFO circuits. Here the stuck-at faults on the primary input fanout branches in addition to stuck-at faults on the primary inputs are essential faults. The theorems and conjectures developed for RFO

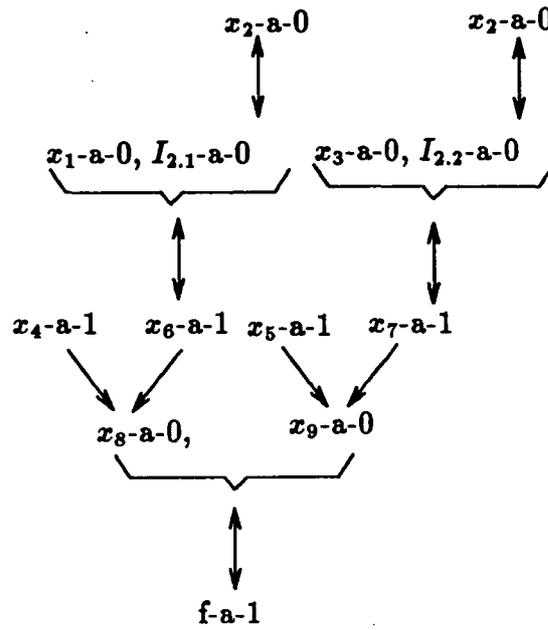


Figure 15: Fault folding graph for f-a-1

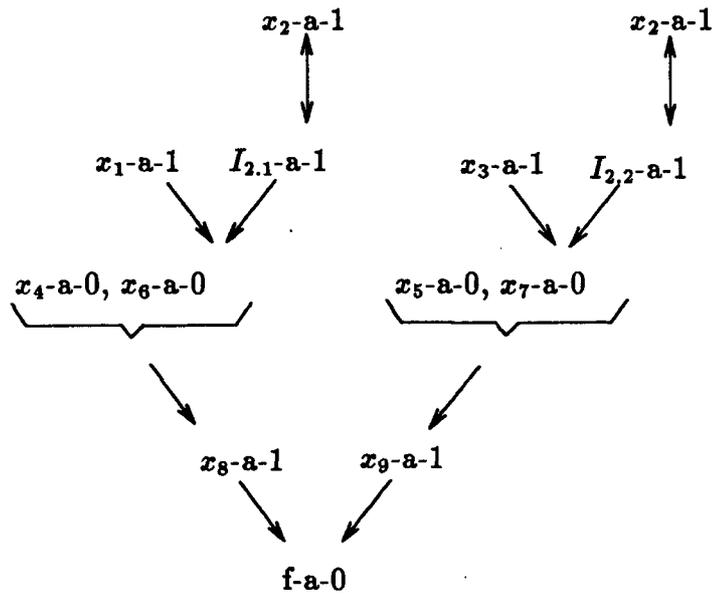


Figure 16: Fault folding graph for f-a-0

are also valid here. An example will be considered to describe the process of deriving the minimal test set.

Example 6 Find the minimal Test set for the circuit shown in Figure 14 .

Step 1:- The fault folding graphs $f-a-1$ and $f-a-0$ are drawn (See figures 15 and 16).

Step 2:- From the fault folding graph for $f-a-0$ (Figure 16) the multiple fault set is formed

$$f_m = \left\{ \left[\begin{array}{c} x_{1-a-1} \\ x_{4-a-0} \end{array} \right] \left[\begin{array}{c} x_{4-a-0} \\ x_{2-a-1} \\ I_{2.1-a-1} \end{array} \right] \left[\begin{array}{c} x_{2-a-1} \\ I_{2.2-a-0} \\ x_{5-a-0} \end{array} \right] \left[\begin{array}{c} x_{3-a-1} \\ x_{5-a-0} \end{array} \right] \right\}$$

From the fault folding graph for $f-a-1$ (Figure 15), the multiple fault set is formed as

$$f_m = \left\{ \left[\begin{array}{c} x_{1-a-0} \\ x_{3-a-0} \\ x_{2-a-0} \\ I_{2.1-a-0} \\ I_{2.2-a-0} \end{array} \right] \left[\begin{array}{c} x_{4-a-1} \\ x_{5-a-1} \end{array} \right] \right\}$$

One of the minimal multiple fault sets covering all the essential single stuck-at faults is

$$F_{m1} = \left\{ \left[\begin{array}{c} x_{1-a-1} \\ x_{4-a-0} \end{array} \right] \left[\begin{array}{c} x_{4-a-0} \\ x_{2-a-1} \\ I_{2.1-a-1} \end{array} \right] \left[\begin{array}{c} x_{2-a-1} \\ I_{2.2-a-1} \\ x_{5-a-0} \end{array} \right] \left[\begin{array}{c} x_{3-a-1} \\ x_{5-a-0} \end{array} \right] \left[\begin{array}{c} x_{1-a-0} \\ x_{3-a-0} \\ x_{2-a-0} \\ I_{2.1-a-0} \\ I_{2.2-a-0} \end{array} \right] \left[\begin{array}{c} x_{4-a-1} \\ x_{5-a-1} \end{array} \right] \right\}$$

Step 3:- The test vector sets T_{m1} for the fault sets F_{m1} is found:-

$$T_{m1} = \langle x_1 x_2 x_3 x_4 x_5 \rangle =$$

$$\left\{ \left(\begin{array}{c} 01010 \\ 01110 \\ 01111 \end{array} \right) \left(\begin{array}{c} 10110 \\ 10010 \end{array} \right) \left(\begin{array}{c} 00101 \\ 10101 \end{array} \right) \left(\begin{array}{c} 01001 \\ 11001 \\ 11011 \end{array} \right) (11111) \left(\begin{array}{c} 00000 \\ 01000 \\ 10100 \end{array} \right) \right\}$$

4.3.2 Method by Inspection

The Example 6 will be reworked using method by Inspection. The same rules used for RFO circuits will be followed.

Gates 1, 4 and 5 are odd level gates and gates 2 and 3 are even level gates. Using the Conjectures 1 through 6 the following results have been arrived at:-

1. x_{4-a-1} (even level gate 2) can be conjoined with x_{5-a-1} (even level gate 3).

2. Gates 4 and 5 are to be treated as primary gates since there is no other gate between the primary inputs and inputs of the gates 4 and 5. The parity of inversion through the two primary input fanout branches is the same(odd). Hence, the fault $I_{2.1-a-0}$ (even gate 4) can be conjoined with $I_{2.2-a-0}$ (even gate), using Clause 2 of Conjecture 3.

Note that for a primary gate, it is enough if one of its inputs is tested for stuck-at-0 fault. So when we test multiple fault $\{I_{2.1-a-0}, I_{2.2-a-0}\}$ we will have tested gates 4 and 5 for stuck-at-0 faults leading to multiple fault set F_m

$$F_m = \left\{ \left[x_{1-a-1} \right] \left[x_{3-a-1} \right] \left[I_{2.2-a-1} \right] \left[I_{2.1-a-1} \right] \left[\begin{array}{c} x_{4-a-1} \\ x_{5-a-1} \end{array} \right] \left[\begin{array}{c} I_{2.1-a-0} \\ I_{2.2-a-0} \end{array} \right] \right\}$$

This is the same as F_{m1} obtained previously in Example 6. The test vectors are found as usual either by path sensitization or Boolean differences.

5 Conclusions

Techniques for obtaining a minimal test vector set for stuck-at faults have been developed. Two new procedures have been described, one of which allows the test vector set to be derived by inspection. The other procedure is an algorithmic procedure and uses the fault folding technique. Both techniques are very simple and highly ergonomically efficient as compared to existing methods. Certain well known theorems have been modified to remove their latent redundancies.

The fault folding methods technique is a graphical method utilizing fault folding graphs. This technique has been proven to work for all kinds of irredundant combinational circuits - NRFO, RFO and primary input fanout circuits. This method is simple and straight forward to use when compared with the existing methods.

The second method - Method by Inspection was developed so that the fault folding graphs needed in the first method need not be drawn. All that is needed is the circuit diagram. The conjectures forming this method were developed by translating the graphical language of the fault folding graphs to a layman's language. The only thing one needs to do is to keep track of the gate number. This method, also, works very well with all irredundant combinational circuits.

The whole procedure for stuck-at faults could be automated, such that a computer program could look at a network file and develop a minimal test vector set or minimal fault set. Such software would be of immense help to the industry and test engineering community as a whole.

References

- [1] D. B. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets", *IEEE Transactions on Electronic Computers*, Vol EC-15, Feb. 1966, pp. 66-73.

- [2] G. E. Whitney, "Algebraic Fault Analysis for Constrained Combinational Networks", *IEEE Transactions on Computers*, Vol. C-20, Feb. 1971, pp. 148-153.
- [3] Kilin To, "Fault Folding for Irredundant and Redundant Combinational circuits", *IEEE Transactions on Computers*, Vol. C-22, Nov. 1973, pp. 1008-1015.
- [4] S. M. Reddy, "Comments on Minimal Fault Tests for Combinational Networks", *IEEE Transactions on Computers*, Vol. C-26, Mar. 77, pp. 318-319.
- [5] M. Fridrich and W. A. Davis, "Minimal Fault Tests for Combinational Networks", *IEEE Transactions on Computers*, Vol. C-23, Aug. 74, pp. 850-859.
- [6] S. M. Reddy, V. D. Agarwal and S. K. Jain, "A Gate Level Model for CMOS Combinational Logic Circuits", *Proceedings of the 14th International Fault-Tolerant Computing Symposium*, Jun. 84.
- [7] D. Baschiera and B. Courtois, "Testing CMOS: A Challenge", *VLSI DESIGN*, Oct. 1984, pp. 58-62.
- [8] S. M. Reddy, M. K. Reddy and J. G. Kuhl, "On Testable Design for CMOS Logic Circuits", *1989 International Test Conference*, paper 15.2, pp. 435-445.
- [9] S. M. Reddy, M. K. Reddy and V. D. Agrawal, "Robust Tests for Stuck-open Faults in CMOS Combinational Logic Circuits", *Proceedings of 14th Annual Symposium on Fault-Tolerant Computing*, Jun. 1984, pp. 44-49.
- [10] S. M. Reddy and M. K. Reddy, "Testable Realization for FET Stuck-open Faults in CMOS Combinational Logic Circuits", *IEEE Transactions on Computers*, Aug. 1986, pp. 742-754.
- [11] S. S. Yau and Y. S. Tang, "An Efficient Algorithm for Generating Complete Test Sets for Combinational Logic Circuits", *IEEE Transactions on Computers*, Vol. C-10, Nov. 1971, pp. 1245-1251.
- [12] A. K. Susskind, "Diagnostics for Logic Networks", *IEEE Spectrum*, Oct. 1973, pp. 40-47.
- [13] I. Kohavi, "Fault Diagnosis of Logic Circuits", *IEEE Conference Records of 10th Annual Symposium On Switching and Automata Theory*, Oct. 1969, pp. 166-173.
- [14] L. W. Bearnson and C. C. Carroll, "On the design of minimum length fault tests for Combinational Logic Circuits", *IEEE Transactions on Computers*, Vol. C-20, Nov. 1971, pp. 1353-1356.
- [15] Givone, "Introduction to Switching Theory", *Tata-McGraw Hill, India*, 1970.

Session 8
VLSI Processors

Chairman: Roy Foote

Application Specific Serial Arithmetic Arrays

K. Winters, D. Mathews and T. Thompson
Department of Electrical Engineering
Montana State University
Bozeman, Montana

Abstract— High performance systolic arrays of serial-parallel multiplier elements may be rapidly constructed for specific applications by applying hardware description language techniques to a library of full-custom CMOS building blocks. Single clock precharged circuits have been implemented for these arrays at clock rates in excess of 100Mhz using economical 2-micron (minimum feature size) CMOS processes, which may be quickly configured for a variety of applications. A number of application-specific arrays are presented, including a 2-D convolver for image processing, an integer polynomial solver, and a finite-field polynomial solver.

1 Introduction

Parallel arrays of serial arithmetic processor modules have been proposed since the late 1950s [13] for high performance signal and information processing. Only since the development of very large scale integrated (VLSI) circuit technologies nearly 20 years later, has it been possible to fully exploit the speed and modularity of this architecture type. This paper presents an approach to the design of special-purpose systolic arrays that attempts to maximize performance and cost effectiveness and minimize design time. A family of high-speed CMOS circuits has been tailored to take advantage of the inherent locality of connection of serial processor arrays, while modern hardware description language (HDL) and logic synthesis techniques are employed to enable very rapid development of simulation models, standard-cell prototype components, and high-performance full custom CMOS implementations.

A design methodology has been adopted for the development of these arrays that combines bottom-up and top-down approaches. First, a hardware description language (HDL) model is written that reflects the structure of the array to the cell level and functional behavior at the cell level. This model is simulated and debugged, becoming the defined specification of the array. At this point, a standard-cell semi-custom implementation may be quickly generated using logic synthesis tools. Standard-cell realizations may be sufficient in performance for many applications, or optionally may serve as prototypes for full-custom components developed later.

Finally, full-custom CMOS versions of the datapath cells are designed to the HPL specification, verified with SPICE, and laid out with a physical artwork editor. The artwork is checked for design rule violations, and extracted for simulation using the same test

vectors used to verify the HDL models. Parasitic capacitances are also extracted, and SPICE timing models annotated. Once the datapath cell dimensions are known, the HDL models used to define the topology of the array can, with very little modification, be used to construct the tiled datapath artwork. Composite artwork layout, padding generation, and module place-and-route are performed automatically, using the same tools as for standard-cell prototypes.

It is crucial that there be a close coupling between the array architecture and the CMOS circuit design if clock speed is to be maximized. A family of dynamic logic circuits was developed that avoids precharge race conditions, clock skew, and hold time considerations, and minimizes series fet and static inverter stage delays. The resulting differential synchronous logic elements are clocked by a single uncomplemented signal. Conversely, array architectures are constrained to highly systolic configurations with very local connectivity between synchronous elements.

The OCT toolset from the University of California, Berkeley, is used for top-down behavioral and structural HDL modeling, logic and switch simulation, CMOS standard-cell prototype synthesis, and custom layout. The toolset is an integrated system for VLSI design, including tools and libraries for multi-level logic synthesis, standard cell placement and routing, programmable logic array and gate matrix module generation, custom cell design, and utility programs for managing design data. Most tools are integrated with the OCT data-base manager and the X-based VEM graphical user interface.

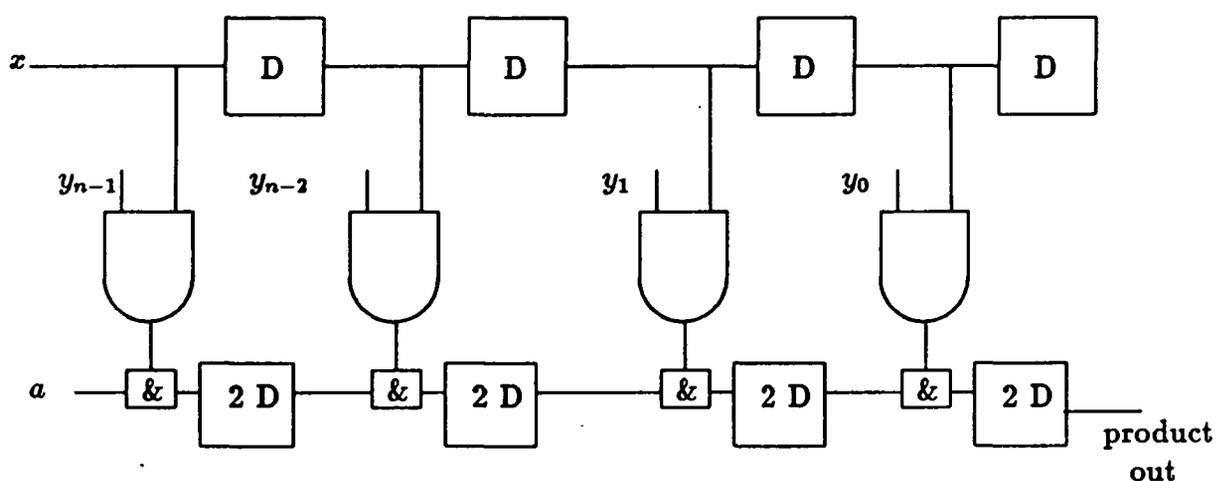


Figure 1: Fully Systolic Serial-Parallel Multiplier

2 Single-Clock Serial-Parallel Multipliers

A single-clock dynamic multiplier circuit has been developed in 2 micron CMOS technology using differential single-clock NORA techniques. This multiplier, shown in Figure 1, is a

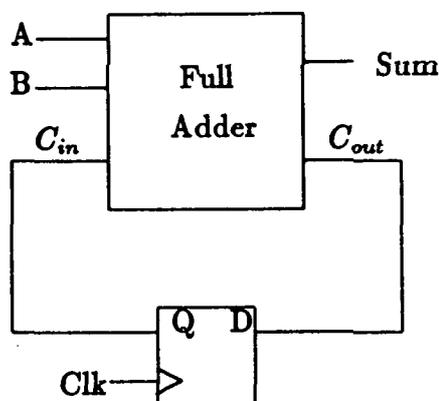


Figure 2: Bit Serial Adder

systolic adaptation of the serial-parallel multiplier introduced by Yaohan Chu in 1962 [3] and later by Daniel Hampel [7]. The boxes labeled with ampersands represent bit-serial adders, shown in Figure 2. "D"s represent delay flip-flops.

This multiplier requires $2n$ clock cycles to multiply two n -bit operands. Essentially, the parallel multiplicand, y , is multiplied by the serial multiplier, x , on each clock cycle and accumulated in the product pipeline. The multiplier input is pipelined through n stages, which reduces the maximum fanout of the multiplier input to 2 gate inputs. Commensurate delay elements are added to the product pipeline, totaling two delay elements per bit-slice. A primary advantage of this systolic multiplier circuit is that it has the proper ratio of operand to product storage. That is, it contains n bits of storage in the multiplier pipeline, and $2n$ bits of storage in the product pipeline [15].

In operation, the multiplier is shifted into the module in n cycles, followed by n cycles of zeroes. In some applications, it is desirable to utilize the full bandwidth of the multiplier pipeline during these "dead" cycles. This is accomplished by ANDing the multiplicand, y , with the output of an n -bit Johnson-Ring or Mobius [12] counter, as illustrated later.

The product pipeline can accumulate external addends with its partial products without additional adder logic. The external addend is shifted serially in the a input and must be pre-shifted n bits into the product pipeline before the LSB of the multiplier is entered. Thus, the lower n bits of the addend occupy the high n bits of the product pipeline at the beginning of a multiply sequence. Then, n multiplier bits are shifted into the multiplier pipeline, leaving the LSB of the accumulated product in the LSB of the product pipeline. During the next n clock cycles, the multiplier is shifted out (replaced by zeroes), the low n bits of the product are shifted out of the product pipeline, the high n bits of the product are left in the low half of the product pipeline, while the low n bits of the next addend are pre-shifted into the high product pipeline half.

This multiplier configuration operates on positive integers. For signed two's complement operands, the sign bit of the multiplier operand, MSB, must be repeated, or sign extended, and shifted into the multiplier pipeline during the second n cycles of the mul-

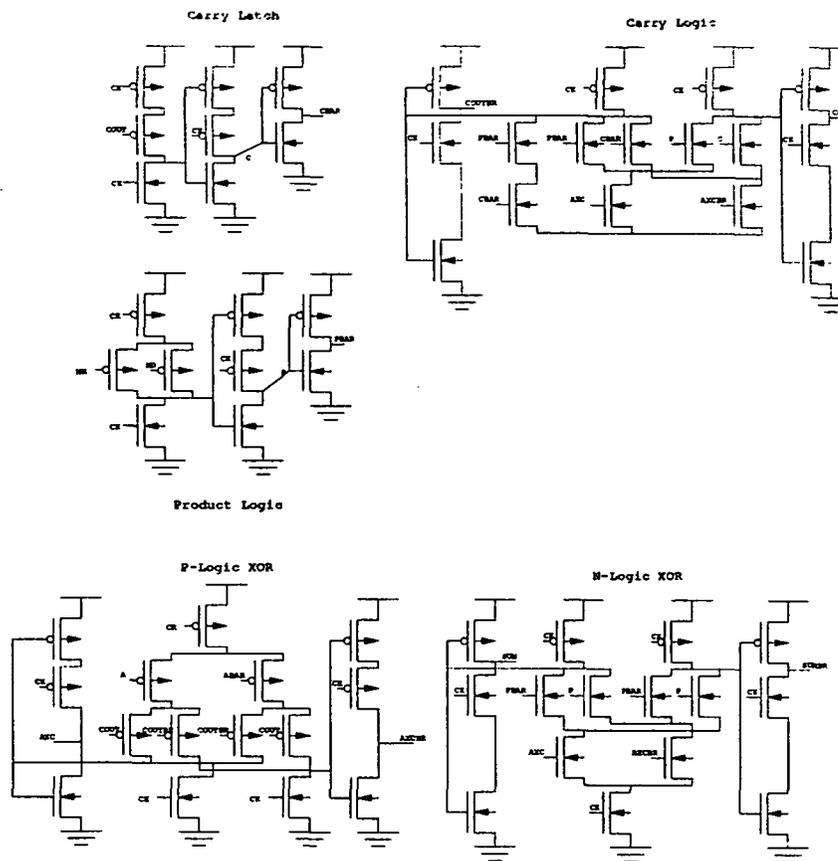


Figure 3: Single Clock Multiplier Cell Schematic

multiply. This can be implemented simply with a sign hold flip-flop preceding the multiplier pipeline that is load-disabled during the second n cycles of the multiply [7,11].

A circuit diagram for the systolic multiplier bit-slice is shown in Figure 3. This circuit was designed using a single-phase, noncomplemented clocking scheme, capable of higher clock rates than conventional dual-clock or complemented-clock techniques, since clock skew is less of a problem [8], [9], [1]. Here, synchronous systems may be constructed from alternating precharged P-fet and N-fet logic stages separated by clocked inverters. This scheme, like its predecessors NORA [4] and Domino [10] logic, is free of precharge race failures, yet eliminates the need for a complemented clock and associated skewing problems at very high clock rates.

The P-logic stages of the circuit at left precharge low during clock high and evaluate during clock low, while the N-logic stages in the center precharge high during clock low and evaluate during clock high. These two stages provide the product of the operands, and also one of the bits of storage in the product pipeline. The product of the operands is generated in the following manner. The product logic is a P-logic AND gate, which multiplies mr and md to produce p . The addend a is an input from the previous stage of the multiplier, and is XORed with c in the P-logic XOR. The Carry logic generates the carry which will be used in the succeeding clock cycle's multiplication. The accumulated product is generated in the N Logic by XORing a -XOR- c and p . This product is used as the addend for the next cell.

The two flip-flop circuits at the far right complete the pipelines. The mr flip-flop

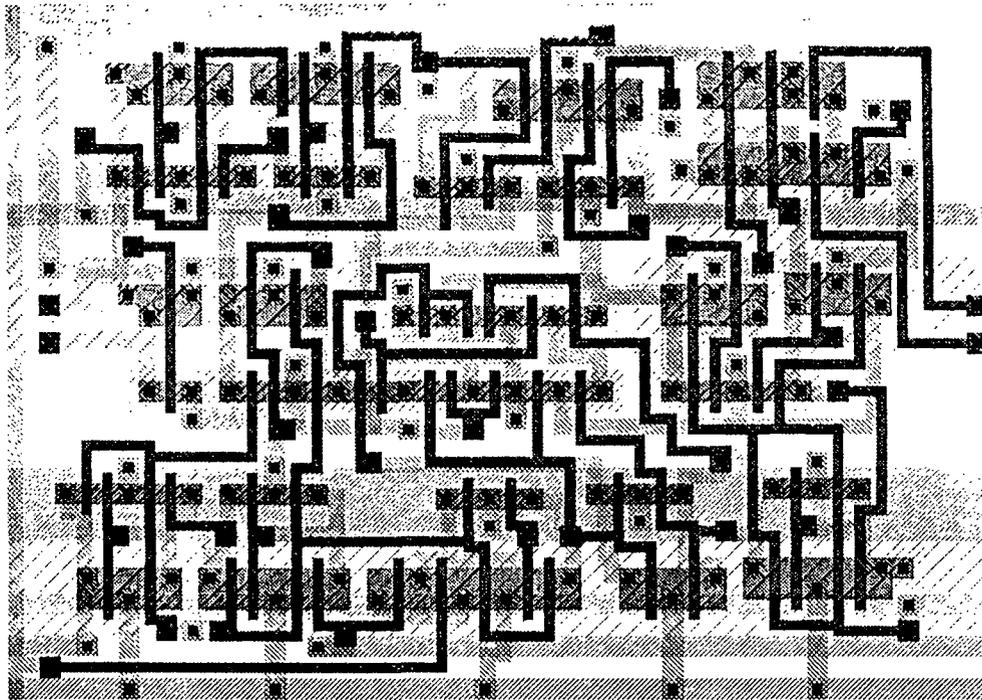


Figure 4: Single clock multiplier cell artwork

provides the storage in the multiplier pipeline, and the sumbar flipflop provides the second bit of storage in the product pipeline.

The following constraints are imposed to maximize clock rate in array architectures with very high locality of connection. A maximum of two series logic fets are used in any dynamic stage. Differential logic configurations are used wherever there are two logic fets in series and complemented outputs are required, such as the P-logic and N-logic differential XOR stages. Discrete output inverters are allowed only where one series logic fet exists in a dynamic stage, such as the *md-AND-mr* P-logic stage. N-logic stages are used to drive the cell outputs to take advantage of higher N mobility.

In state machines built from differential or dual-rail logic, there exist possible lockup states where differential outputs are stuck both ones or both zeroes. To address lockup states in the multiplier cell, the additional term *pbar-AND-cbar* is added to the logic equation for *coutbar*. This ensures that after a clock cycle the circuit will break out of the case where *cout* and *coutbar* both are stuck at zero. The case where *cout* and *coutbar* are stuck at one corrects itself when *p* becomes a one. This suggests that a power-up procedure should be run to ensure that the circuit has broken out of the stuck cases and also to initialize the pipelines.

The layout of the systolic multiplier cell is shown in Figure 4. This cell is 130 by 183 microns in size. The multiplier was simulated¹ at clock rates greater than 100 MHz using MOSIS 2-micron design rules, assuming that the output load of the cell was an identical cell.

¹Using Tektronix, Inc. Tekspice, Tektronix MFET Level 2 device models, and slow speed device parameters for a MOSIS 2-micron (drawn gate length) SCMOS process, Revision 6 design rules, at T=100 degrees C and Vdd = 4.5V.

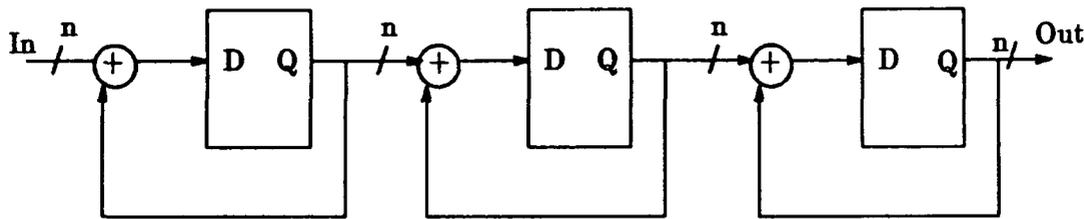


Figure 5: Babbage polynomial solver

3 Babbage Polynomial Solver

The Difference Engine, proposed by Charles Babbage in 1822, can be easily adapted to serial arithmetic, providing a simple example of the proposed design methodology and circuit implementation. Once initialized for a known value of polynomial $p(X)$, the Babbage polynomial solver evaluates $p(X+1)$, $p(X+2)$, and so on, requiring only one clock cycle per evaluation. This method of finite differences is based on the binomial theorem, which guarantees that a polynomial $p(X)$, when evaluated at $p(X+1)$, will yield a new polynomial of X : the sum of $p(X)$ and a remainder polynomial, $r(X)$. Thus, polynomials evaluated at successive values may be decomposed recursively.

The general form of the Babbage Difference Engine is shown in Figure 5. It consists of only adders and pipeline registers.

Since the Babbage machine is systolic by column, the performance bottleneck is in the adder carry propagation logic. This can be reduced by pipelining each adder bit by row, replacing each parallel adder stage with a bit-serial adder. The resulting datapath architecture, shown in Figure 6, consists of a single cell.

The first step is to define the behavior and structure of the Babbage array cell, called *babbit*. This is done in the Berkeley OCT environment using an HDL called BDS to define behavior, Table 1, and BDNET to define structure as in Table 2.

Next, a BDNET description of the datapath topology is constructed,

The general form of the Babbage Difference Engine is shown in Figure 5. It consists of only adders and pipeline registers.

Since the Babbage machine is systolic by column, the performance bottleneck is in the adder carry propagation logic. This can be reduced by pipelining each adder bit by row, replacing each parallel adder stage with a bit-serial adder. The resulting datapath architecture, shown in Figure 6, consists of a single cell.

The first step is to define the behavior and structure of the Babbage array cell, called *babbit*. This is done in the Berkeley OCT environment using an HDL called BDS to define behavior, Table 1, and BDNET to define structure as in Table 2.

Next, a BDNET description of the datapath topology is constructed, configured here for a degree-3 by 4-bit engine in Table 3. This process required less than one afternoon, excluding simulation test vector generation.

After simulation and optional mask release of the standard-cell prototype IC, custom versions of the bit cell are designed employing the single-clock dynamic circuits described

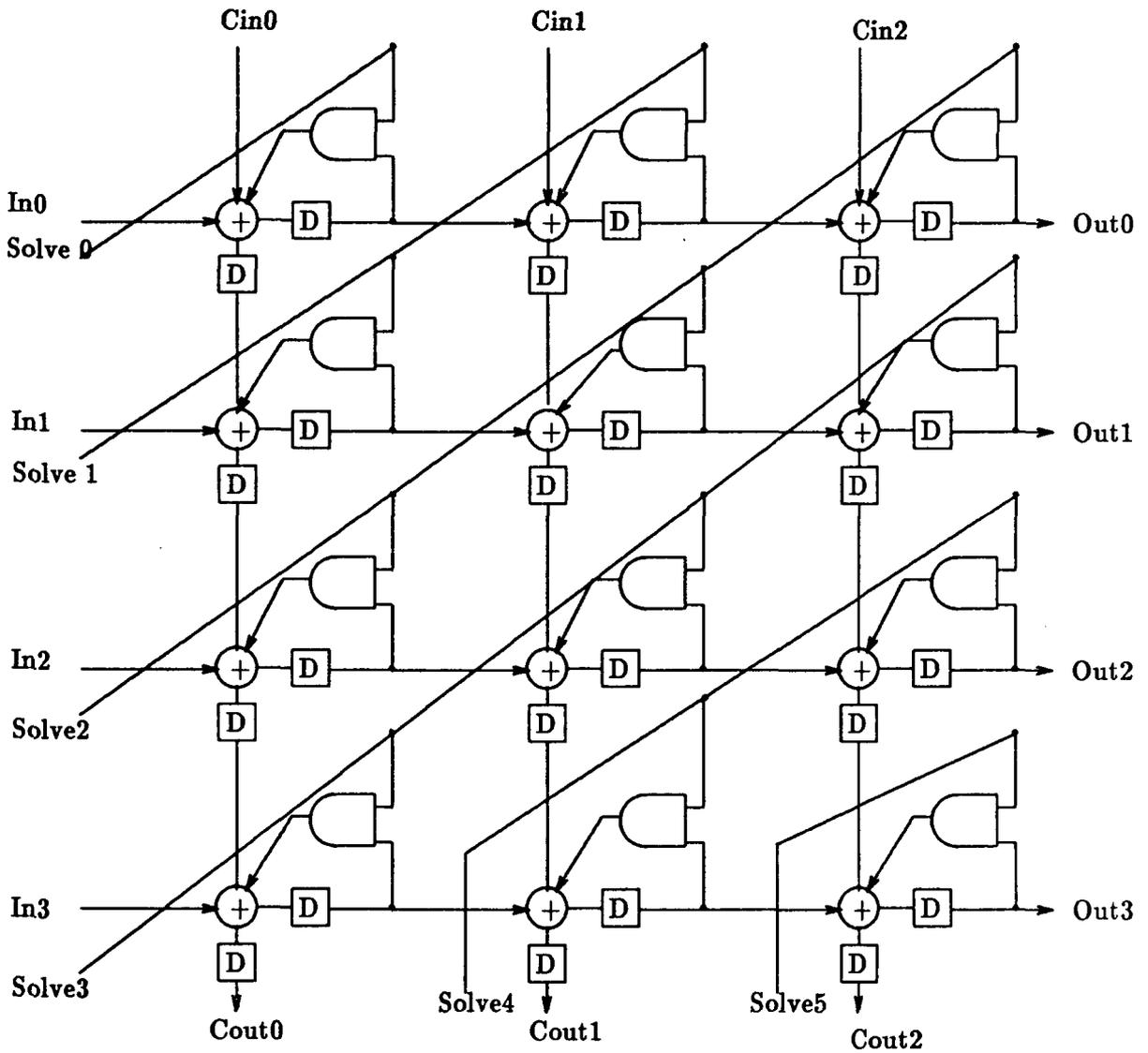


Figure 6: Systolic Babbage polynomial solver

8.1.8

```
! BABIT difference engine bit module BDS description
! FILE: babit.bds
! Kel Winters
! Rev: 5-22-90
```

```
!Vars:          in - addend input, solve - add enable
!              cin - carry in, f - feedback input, p - solve AND f
!              sum - sum output, c - carry out
```

```
model babit sum < 0 >, c < 0 > = in < 0 >, solve < 0 >, f < 0 >, cin < 0 >;
routine cycle;
state p < 0 >;
p = solve AND f;
sum = in XOR p XOR cin;
c = (in AND p) OR (in AND cin) OR (p AND cin);
ENDROUTINE;
ENDMODEL;
```

Table 1: Babit Behavioral Description

earlier. Starting with a basic set of CMOS “building blocks” for constructing serial adder cells with delay elements, full custom babit realization required about two engineer-weeks. The same BDNET description used to construct the standard cell datapath is then configured to build the tiled custom version, Figure 7.

4 Two-Dimensional Integer Convolver

The two-dimensional convolver discussed in this section is essentially a digital filter for two-dimensional integer-valued image data. The purpose of the convolver is to filter out specific information, or data patterns, which are defined by the convolution mask at high speeds. The use of the convolver assumes that the processed data will be subtracted from the image so that the desired data is the only information which is retained. This module is intended for configuration for applications such as LaPlacian pyramid image compaction [2], [6], feature extraction, and enhancement/restoration.

The organization of the convolver is now row parallel, so that a column of image data enters the convolver at the same time. The structure is basically an array of serial-parallel multiplier/accumulators with a few modifications to the circuit discussed previously.

A block diagram of the convolver module is shown in Figure 8. The convolver moves the raw data through its registers, accumulating the convolved result as it travels. The input is multiplied by the first values in the positive-valued convolution mask, and is piped to the next multiplier/accumulator register (column) as the addend for its operation. The partially accumulated result, for a given location in the data, is piped through the

```

! BABIT difference engine bit cell BDNET description
! File: babit.bdnet
! Kel Winters
! Rev: 5-23-90
!! Variables:    in - addend input, solve - add enable
!               cin - carry in, cout - carry out delayed
!               out - sum output delayed
!
MODEL           babit:unplaced;
TECHNOLOGY      scmos;
VIEWTYPE        SYMBOLIC;
EDITSTYLE       SYMBOLIC;

INPUT           clk, in < 0 >, solve < 0 >, cin < 0 >;
OUTPUT          cout < 0 >, out < 0 >;
SUPPLY          Vdd;
GROUND          GND;

INSTANCE        babit:logic PROMOTE;
                f < 0 >: out < 0 >;
INSTANCE        "~/octtools/lib/technology/scmos/msu/stdcell2.0/dfnf311":physical
                DATA1:sum < 0 >;
                CLK3:clk;
                Q:out < 0 >;
                Qb:UNCONNECTED;
                "Vdd!":Vdd;
                "GND!":GND;

INSTANCE        "~/octtools/lib/technology/scmos/msu/stdcell2.0/dfnf311":physical
                DATA1:c < 0 >;
                CLK3:clk;
                Q:cout < 0 >;
                Qb:UNCONNECTED;
                "Vdd!":Vdd;
                "GND!":GND;

ENDMODEL;

```

Table 2: Babit Structural Description

8.1.10

! BABAR3x4, difference engine array BDNET description

! File: babar.bdnet

! Kel Winters

! Rev: 5-24-90

!

! Variables: same as Babit

!
! NOTE: this file is written to be extensible in array height,
! 4, and width, 3. Copy this file into the actual bdnet
! bdnet file and substitute the desired values for 4 and
! 3 before running bdnet

MODEL babar3x4:unplaced;

TECHNOLOGY scmos;

VIEWTYPE SYMBOLIC;

EDITSTYLE SYMBOLIC;

INPUT $clk, in < (4 - 1) : 0 >, cin < (3 - 1) : 0 >, solve < (4 + 3 - 2) : 0 >$;

OUTPUT $out < (4 - 1) : 0 >: in < ((3 + 1) * 4 - 1) : (4 * 3) >$;

OUTPUT $cout < (3 - 1) : 0 >: cin < (3 * (4 + 1) - 1) : (3 * 4) >$;

SUPPLY Vdd ;

GROUND GND ;

ARRAY $\%x$ FROM 0 to (3-1) OF

ARRAY $\%y$ FROM 0 to (4-1) OF

INSTANCE babit:unplaced

$clk : clk$;

$in < 0 >: in < \%x * 4 + \%y >$;

$out < 0 >: in < (\%x + 1) * 4 + \%y >$;

$cin < 0 >: cin < \%y * 3 + \%x >$;

$cout < 0 >: cin < (\%y + 1) * 3 + \%x >$;

$solve < 0 >: solve < \%x + \%y >$;

$Vdd : Vdd$;

$GND : GND$;

ENDMODEL;

Table 3: BABAR 3 × 4 Structural Description

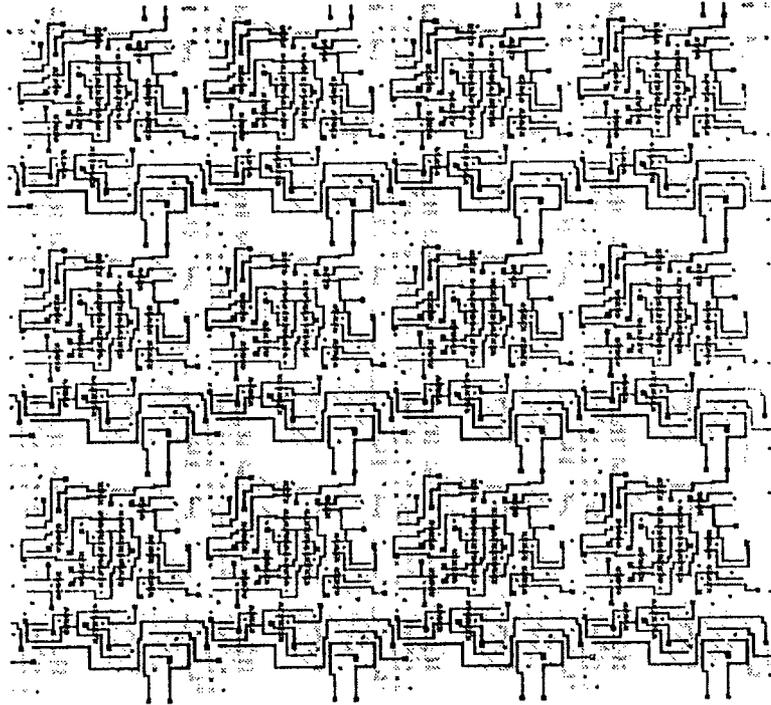


Figure 7: Custom single clock polynomial solver module

array in parallel with data at vertical neighbor locations. It is piped to the multiplication operation with which it must be added next (this includes diagonal/vertical routing.) The last register operation produces a convolved value for each point in one column of the two-dimensional input data.

Three modifications have been made on the systolic multiplier circuit to build the convolver circuit. A second multiplier (*mr*) pipeline has been added, along with a Johnson ring counter and a control cell which routes data through the multiplier/accumulator registers.

The multiplier storage line, or pipeline, consists of two flip-flops per bit slice (multiplier/accumulator cell.) This serves to provide a pipeline path for the multiplier (image) data that is matched in length to the product accumulator pipe so that the product sums from neighboring rows can be synchronized then combined.

This array configuration calculates a convolution matrix for every odd column of the input matrix. Therefore, each systolic multiplier stage receives an *n*-bit serial operand, then ignores the next as it completes the second *n* clock cycles of the multiply sequence. This second operand must not be lost, however, as it will be used by the next multiplier stage downstream in the succeeding *n* cycles. Therefore, an *n*-bit Johnson Ring counter, described previously, shifted through the multiplier pipeline.

The Johnson ring counter is effectively an *n*-bit shift register with its output fed back, inverted to its input. A serial reset is provided to initialize the ring. Again, the ring counter serves to allow a second set of *n*-bit words to share the *mr* pipeline, disabling the

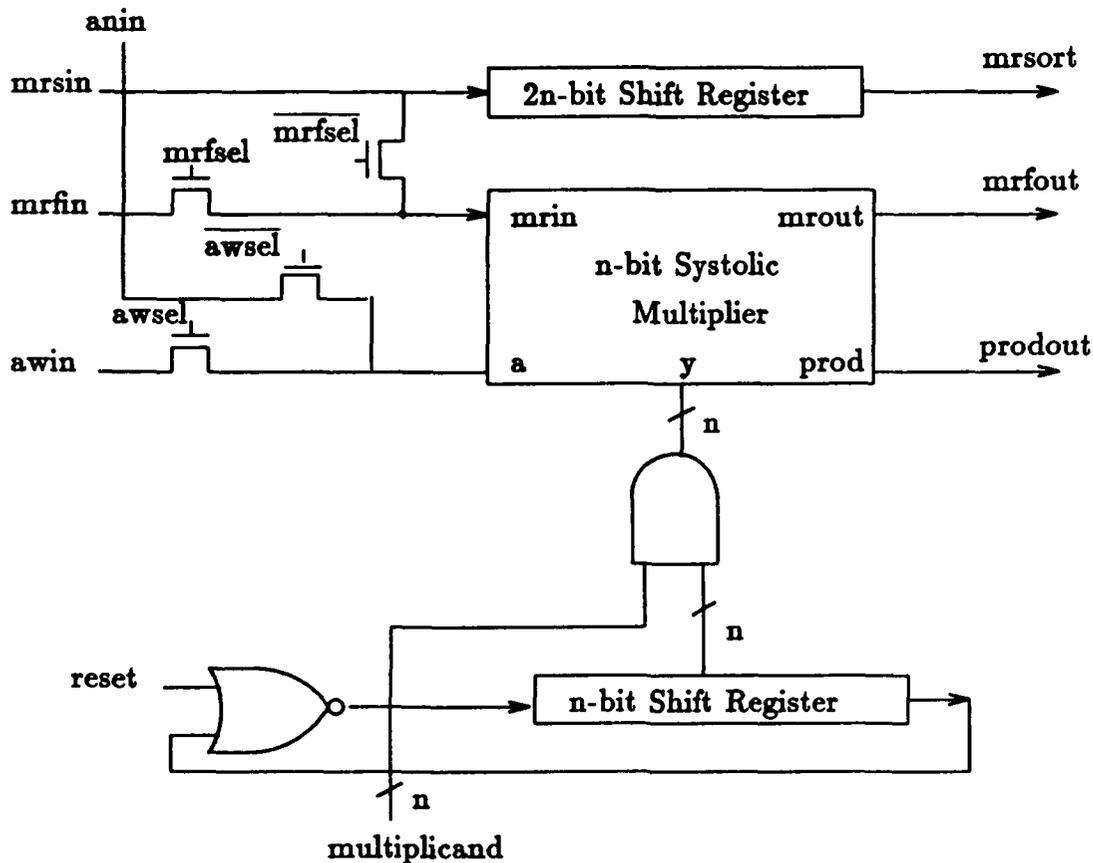


Figure 8: Two dimensional systolic convolver register

unused odd words from affecting the multiplier logic. For the two's-complement capability, the odd words in the mr pipeline should be filled with sign-extension bits. The Johnson ring counter would then be replaced with a second mr pipeline. Naturally, overflows from the sign-extension field must be cleared between multiply-accumulate operations.

The complete convolver bit cell contains 119 fets and is 150 by 160 microns in size, using 2-micron Mosis CMOS design rules. Artwork is shown in Figure 9.

The control cell includes four pass gates, the reset for the Johnson ring counter, and the power, ground and clock routing. Two of the pass gates switch the addend input from the west or the northwest neighbors upstream. The other two pass gates determine whether the incoming multiplier is processed through the multiplier pipeline or detoured through the slow (twice delayed) multiplier pipeline. This allows adjacent row products to be summed, with the multiplier pipelines matched to the product pipelines. Power, ground and control signals are routed vertically, by column.

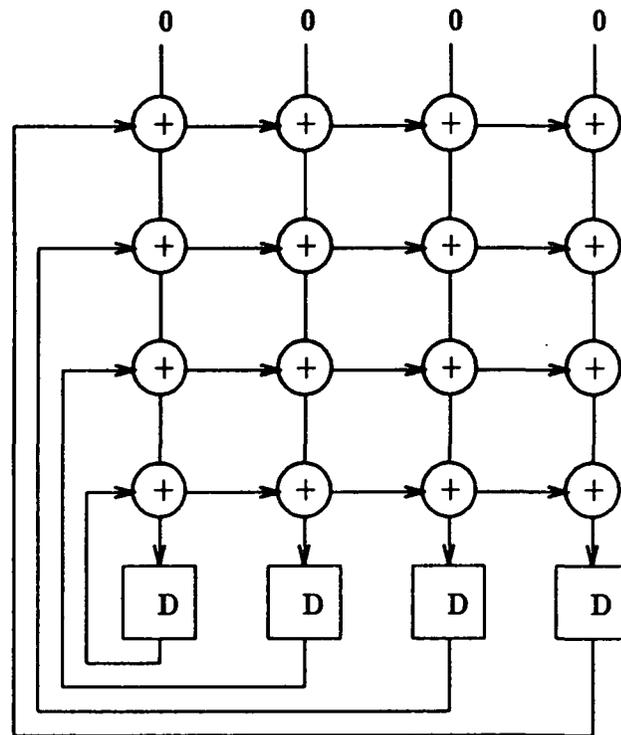


Figure 10: Finite field constant multiplier

5 Two-Dimensional Systolic State Machines

Two-dimensional systolic state machines are defined here as automata where the present-state feedback is organized as a set of n row pipelines and next-state equations organized as n column pipelines, where n is the width of the state variable vector. Rows and columns must be ordered so that all loops from a state-variable to itself contain n delays. The intent here is to maximize clock speed independently of the state vector width or functional complexity.

For example, a finite-field constant multiplier-accumulator consists of a synchronous register to store the present-state and an array of exclusive-or (XOR) gates, representing modulo-2 adders. Next-state equations are sum-of-product functions of the present-state vector. In its simplest organization [14], the present-state register outputs are fed back as row lines, and the next-state equations are organized as columns of XOR gates or crossovers as needed, Figure 10. Here, XOR gate sites are represented by "+," and flip-flops by "D." Note that the XOR gate outputs to the bottom; the row input is passed through the row. The critical timing path in this organization is the fanout of the state register, up to n XOR gates across n columns, and up to n XOR gate delays routed through n rows.

A 2-D systolic finite-field multiplier is shown in Figure 11. Here, the present-state feedback is through n n -stage shift registers, one per row. Column XOR gates are also pipelined, one stage per row. Again, rows and columns are organized so that all loops from feedback (present-state) signals to themselves contain n delays. This configuration

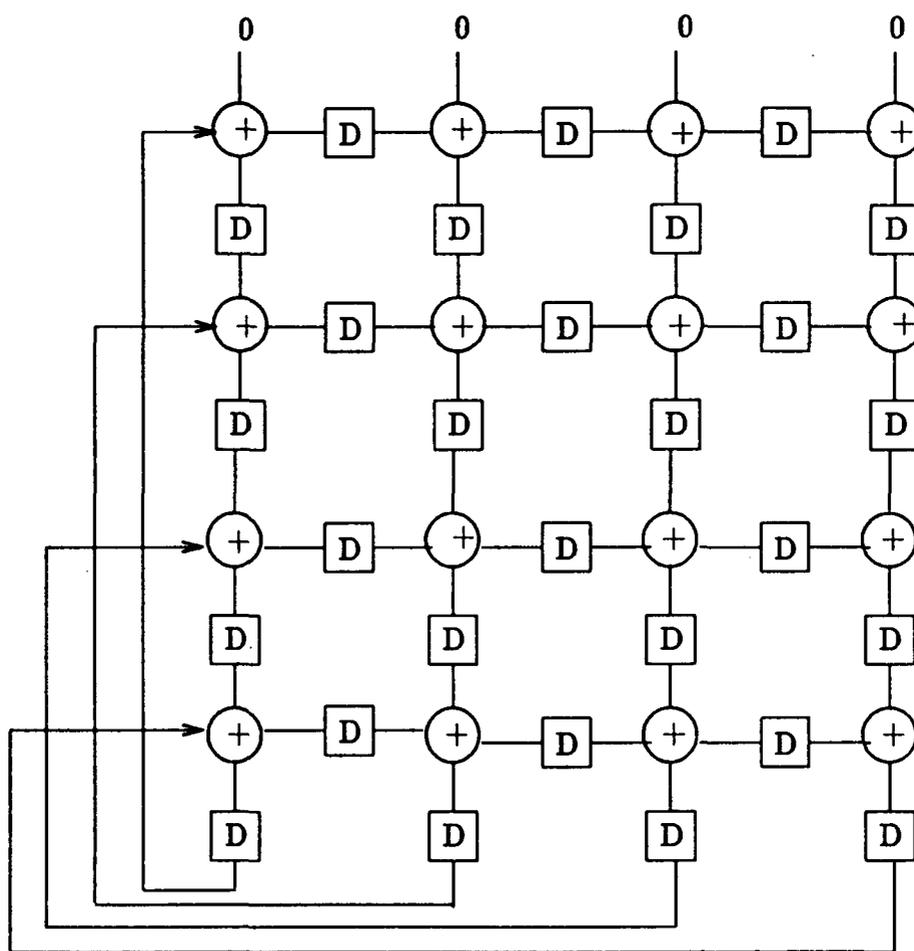


Figure 11: Systolic finite field multiplier

Configuration	Flip-Flops	XOR Sites	n-mult cycles	Fanout Max.	Series Delays
n Conventional	n^2	n^3	1	n	n
Systolic	$2n^2$	n^2	n	1	1

Table 4: Finite-Field Constant Multiplier Time-Area Considerations

performs the function of n conventional multipliers operating in parallel. The time-area considerations for n conventional multipliers and the proposed systolic array are compared in Table 4. The fourth column gives the number of clock cycles required to perform n multiplications of n bits.

If the ratio of flip-flop area to XOR gate area is assumed to be 1:3, an area figure of merit for the n conventional constant multipliers is given by $n + n^2/3$ compared to $7n/3$ for the systolic case. This implies that the area is comparable at $n = 4$. At $n = 8$, common to Reed-Solomon error correction codes, the n conventional constant multipliers would be about 50% larger than the systolic version. Thus, the systolic version would have to operate at about five times the clock rate of the n conventional multipliers to reach time-area parity (not difficult given the difference in fanout and series delays).

This thumbnail sketch is an oversimplification, given that each technique tends to favor different circuit realizations, but serves to illustrate the potential of 2-D systolic state machines. Practical applications in finite field constant multipliers would likely involve an optimal compromise: a systolic array of non-systolic sub-arrays.

The advantages of this technique are more apparent in applications where the value of n grows large. One such application under investigation is the simulation of Hopfield-type neural networks [5], where the next-state of n neurons is a function of the present-state of n neurons described by an n -by- n matrix. Here, neurons would be realized by systolic columns in the state machine array, dendrites by row pipelines, and synaptic connections by systolic serial-parallel multipliers (described previously) at the intersection of each row and column. This array would emulate n neural networks operating in parallel.

6 Conclusions and Future Directions

A technique for designing high-performance systolic serial VLSI arithmetic arrays has been presented, combining a new variation of differential single-clock dynamic CMOS circuits, a systolic adaptation of Chu's serial-parallel multiplier, and a structured design methodology that combines recent advances in hardware description language synthesis and modeling with full-custom IC design.

For the class of applications that can be decomposed into a regular array of arithmetic modules, it is proposed that HDL models may be developed in hours, standard-cell prototype ICs using logic synthesis techniques implemented to mask release in days, and high-speed full-custom implementations to mask release in weeks.

A standard-cell implementation of the Babbage style polynomial solver has been re-

leased for fabrication in a 2 micron (drawn gate length) CMOS process through the National Science Foundation MOSIS silicon brokerage service. VLSI Technology, Inc. will provide the fabrication. A full-custom prototype of the same polynomial solver has been designed and will be released shortly in the same process.

Standard-cell and full-custom implementations of the two-dimensional convolver are in development for release in a 2 micron process from the Massachusetts Micro-electronics Center and a 1.2 micron process from the Hewlett-Packard Co. through the MOSIS service. These convolver arrays are configured for hierarchical or Laplacian pyramid image compression, a technique proposed for the Mars Rover Sample Return mission [6]. Other configurations for space telescope correction/enhancement and digital signal processing are also under investigation.

Preliminary work on a finite-field polynomial solver based on 2-D systolic constant multipliers has recently begun. It is anticipated that rapid development of custom high-speed arithmetic arrays will provide practical solutions for a wide array of image, signal, and data processing applications.

Acknowledgement

This work was supported by grants from the NASA Space Engineering Research Center at the University of Idaho, Moscow, and the Montana State University Engineering Experiment Station as well as equipment donations from the Helwett-Packard Co. and Tektronix, Inc. The authors would especially like to thank the following people for their valuable assistance to this work: Gary Maki and Jon Gibson of the NASA SERC at the University of Idaho, Jaye Mathisen and Sanjay Mitra of Montana State University, Rick Spickelmier of UC Berkeley, Paul Cohen of the Massachusetts Microelectronics Center, and Fred Huck of the NASA Langley Research Center.

References

- [1] M. Afghahi and C. Svensson, "A Unified Single-Phase Clocking Scheme for VLSI Systems," *IEEE J. Solid-State Circuits*, Vol. 25, no. 1, Feb. 1990, pp. 225-233.
- [2] P. Burt and E. Adelson, "The Laplacian Pyramid as a Compact Image Code," *IEEE Trans. on Communications*, Vol. COM-31, No. 4, April 1983.
- [3] Y. Chu, *Digital Computer Design Fundamentals*, McGraw-Hill, 1962.
- [4] N. Goncalves and H. De Man, "NORA: A Racefree Dynamic CMOS Technique for

Pipelined Logic Structures," *IEEE J. Solid-State Circuits*, Vol. SC-18, no. 3, June 1983, pp. 261-266.

- [5] J. Hopfield and D. Tand, "Computing with Neural Circuits: A Model," *Science*, Vol. 233, pp. 625-633, August 1986.
- [6] F. Huck, "Rover Imaging Systems for the Mars Rover/Sample Return Mission," Proposal to the NASA Planetary Instrument Definition and Development Program, NASA Langley Research Center, Hampton, VA, March, 1989.
- [7] D. Hampel, K McGuire, and K. Prost, "CMOS/SOS Serial-Parallel Multiplier," *IEEE J. on Solid-State Circuits*, Vol. SC-10, No. 5, October 1975.
- [8] Y. Ji-Ren, I. Kaarlson, and C. Svensson, "A True Single- Phase-Clock Dynamic CMOS Circuit Technique," *IEEE J. Solid-State Circuits*, Vol. SC-22, no. 5, October 1987, pp. 899-901.
- [9] Y. Ji-Ren and C. Svensson, "High-Speed CMOS Circuit Technique," *IEEE J. Solid-State Circuits*, Vol. 24, no. 1, Feb. 1989, pp. 62-70.
- [10] R. Krambeck, C. Lee, and H. Law, "High-Speed Compact Circuits with CMOS," *IEEE J. Solid-State Circuits*, Vol. SC-17, June 1982, pp. 614-619.
- [11] R. Lyon, "Two's Complement Pipeline Multipliers," *IEEE Trans. on Communications*, COM-24, April, 1976, pp. 418-425. 97-107.
- [12] H. Taub and D. Schilling, *Digital Integrated Electronics*, McGraw-Hill Inc., New York, 1977, pp. 349-355.
- [13] S. Unger, "A Computer Oriented Toward Spatial Problems," *Proceedings of the IRE*, Vol. 46, no. 10, pp. 1744-1750, October, 1958.
- [14] K. Winters, "A VLSI Error Location Processor for a High Performance Reed-Solomon Decoder," Master's Thesis, University of Idaho, Moscow, ID, June, 1984.
- [15] K. Winters, "Serial Multiplier Arrays for Parallel Computation," *NASA SERC Symposium on VLSI Design*, Moscow, ID, Jan. 1990.

124
191660^{8.31}
p-8

ACE: Automatic Centroid Extractor for Real Time Target Tracking

K. Cameron, S. Whitaker and J. Canaris
NASA Space Engineering Research Center for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - A high performance video image processor has been implemented which is capable of grouping contiguous pixels from a raster scan image into groups and then calculating centroid information for each object in a frame. The algorithm employed to group pixels is very efficient and is guaranteed to work properly for all convex shapes as well as most concave shapes. Processing speeds are adequate for real time processing of video images having a pixel rate of up to 20 million pixels per second. Pixels may be up to 8 bits wide. The processor is designed to interface directly to a transputer serial link communications channel with no additional hardware. The full custom VLSI processor was implemented in a 1.6 μ m CMOS process and measures 7200 μ m on a side.

1 Introduction

ACE (Auto Centroid Extractor) groups contiguous pixels whose intensities are equal to or exceed a given threshold into separate objects and calculates their centroids. Proper assignment of contiguous pixels to their respective objects is guaranteed for all convex shapes and most concave shapes. Data for a $N \times M$ pixel image is processed in raster scan format with a maximum value of 1024 for N and M . ACE can process in excess of 500 objects per frame. It serially outputs the following information for each object:

$$\sum X_i I_i \quad \sum Y_i I_i \quad \sum I_i \quad \# \text{ of Pixels} \quad (1)$$

where X_i and Y_i are the x - and y -coordinates of the i_{th} pixel in an object respectively, and I_i is the intensity of the i_{th} pixel.

Threshold intensity levels are set as instructed by a Transputer via the serial link interface. Pixels whose intensities fall below the specified level are not recognized as object pixels. Masking of pixels is also performed according to an optional external RAM, which is controlled by the ACE chip. The ACE chip updates the masking pattern as instructed by a Transputer.

ACE also provides status information for each frame processed. This information includes a frame identification count, internal memory overflow flags, and an invalid shape detection flag.

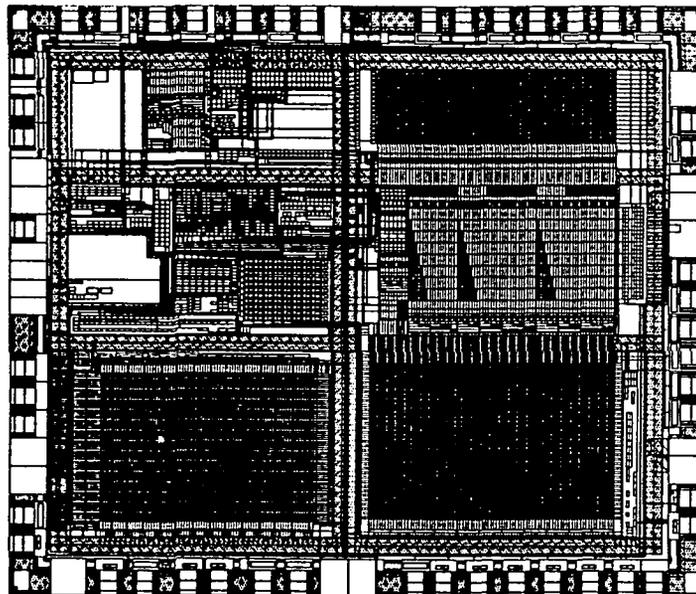


Figure 1: Auto Centroid Extractor

2 Architecture

2.1 Overview

The ACE processor may be divided into a number of subsections, each of which performs one or more of the basic operations necessary for the operation of the circuit. Figure 1 shows a plot of the ACE processor. Each of the major functional blocks in the processor are identified according to their function in Figure 2. Starting at the upper right hand corner, partial results of ongoing centroid calculations are stored in the *scratch ram*. The *calculator* section performs all of the actual arithmetic operations required. Upon the completion of an object, the results of the relevant centroid calculations are queued in the *output fifo*. These data sent out as requested by the host transputer via the *serial link*, after begin formatted by the *output processor*. The group numbers associated with the individual pixels of the previous scan line are supplied to the *image process controller* by the *group fifo*. The *image process controller* uses these group numbers in conjunction with the stream of incoming pixels from the current scan line to group the pixels into contiguous groups, and controls the *calculator* and the *scratch ram* accordingly. The *circle generator/ram interface* performs a number of related support functions. It writes circular masking regions to the optional external mask RAM in response to commands issued by the host transputer via the *serial link* interface. It also provides the masking and pixel level thresholding functions which are required during actual data processing.

Some of the major functional blocks comprising the processor are described below:

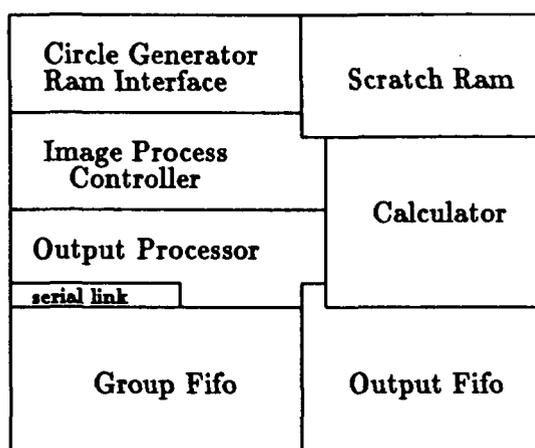


Figure 2: Image Processor Organization

2.2 Image Process Controller

The *image process controller* accepts commands from the *pixel pattern matcher* (see below) and pixel group data from the *group fifo*. Using this information, it controls the *calculator*, regulates the flow of data to and from the *scratch ram*, sends results to the *output fifo*, assigns group numbers, manages the free group storage list, detects and reports invalid objects, and arbitrates conflicting *scratch ram* read/write requests.

Scratch ram access conflicts are very simply handled. Due to the nature of the *trigger patterns* which initiate each data transfer, read requests are always separated by at least one unused access cycle. The same is true of write requests. All contentions can, therefore, be avoided by delaying the write by one cycle each time such a conflict occurs. Simultaneous read/write accesses to the same memory location are permitted, however, since very fast turn around of data through the *scratch ram* is occasionally required.

In addition to about 20 small (2-8 states), tightly interacting state machines, the controller also contains a small register stack, which is used primarily to track group numbers, and manage the *scratch ram* memory. Free (unused) group numbers are supplied by a free storage list that is maintained in the *scratch ram* itself. Group numbers are returned to the free list upon the termination of a pixel group. The controller was designed as a group of small state machines, as opposed to a monolithic controller, because of the quasi-independent, yet overlapping nature of the individual commands issued by the pixel pattern matcher (see below). The formulation of a single controller would have been very difficult and would probably have resulted in a combinatoric explosion of states.

2.3 Calculator

The *calculator* section consists of four computational units which transfer data to and from the *scratch ram* and send results to the *output fifo*. The physical organization of this section is shown in Figure 3. The leaf cells from which the calculator is constructed contain a bit slice of all necessary bussing. All signal paths are formed by abutting these

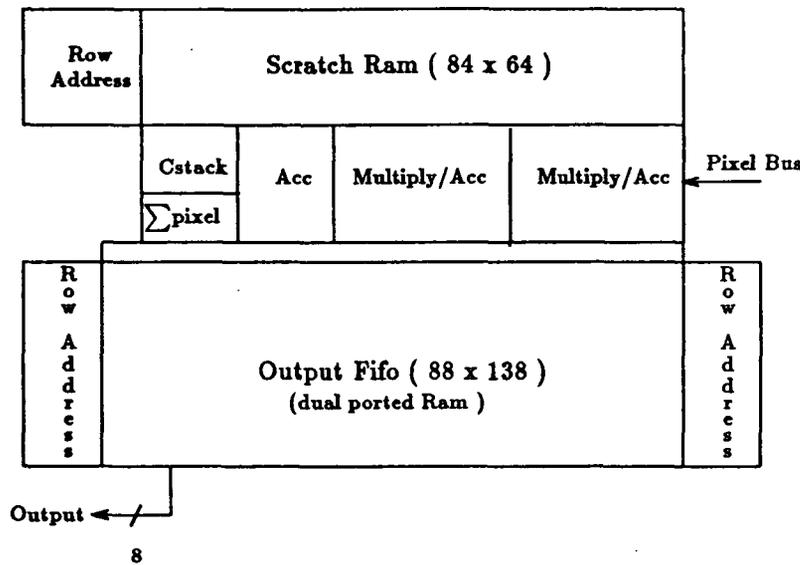


Figure 3: Computation Unit Architecture

cells. In addition, all leaf cells used to construct the calculator are pitch matched to a pair of the ram cells used to construct the *scratch ram* and the *output fifo*. This permitted the calculator section to be implemented as an extremely compact and regular structure.

The *calculator* consists two multiply/accumulators (mac), and two accumulators (acc). Each mac can performs an 8 by 10 bit multiply and two 28 bit accumulations every clock cycle. The pixel intensity acc can accumulate an 8 bit intensity and an 18 bit partial result from the *scratch ram* every clock cycle, and the pixel count acc accumulates a 1 bit pixel flag and 10 bit partial result each clock cycle. Each unit has a latency of two clock cycles.

The structure of the mac units is shown in Figure 4. The sum in and carry in inputs of the full adders at the top of the multiplier array are utilized to perform the required additions. The multiplier array is arranged in a standard configuration [5], with the sum and carry out signals feeding into separate sum and carry registers. The final addition of the carry and sum registers is performed by a Transmission-Gate Conditional-Sum Adder [4], because these adders are very fast and compact [3] when compared to other carry lookahead configurations [2]. The accumulators were created by re-arranging the cells constructed for the mac sections.

2.4 Circle Generator

The *circle generator* draws circular masking regions in the external mask RAM under the direction of the transputer. Each circle is specified in terms of its center point and radius squared. The *circle generator* uses finite difference equations (2) to calculate which pixels fall within the circles so specified, and either clears or sets the pixels accordingly.

$$R_{i+1}^2 = R_i^2 + \overline{2x} + 1 \quad \text{or,} \quad R_{i+1}^2 = R_i^2 + \overline{2y} + 1 \quad (2)$$

The hardware used to solve these equations is shown in Figure 5. It consists of a single

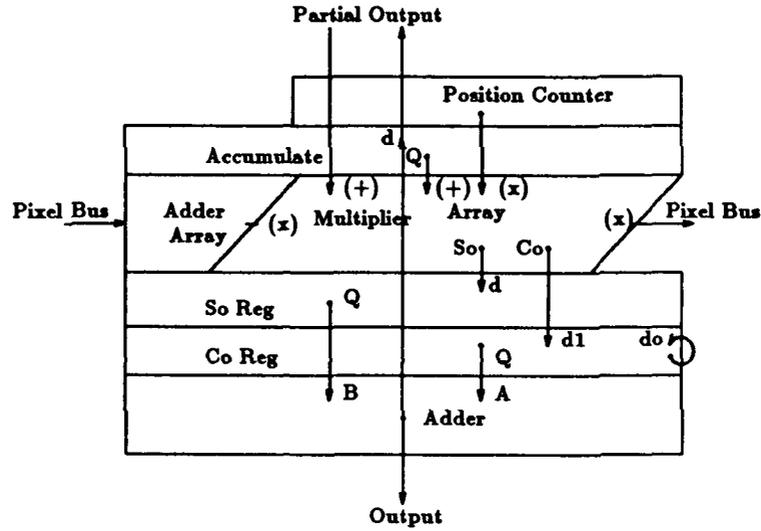


Figure 4: Multiply/Accumulator Architecture

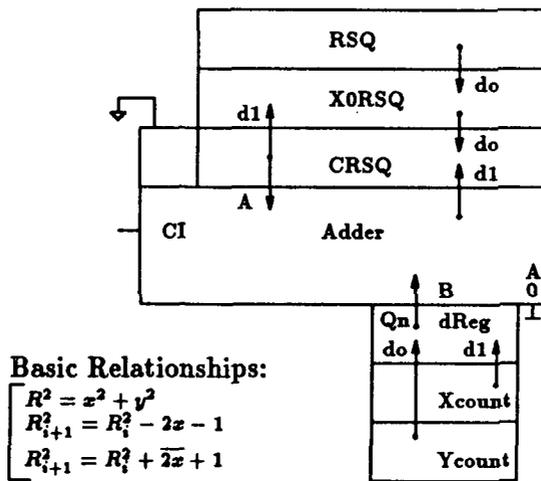


Figure 5: Circle Generator

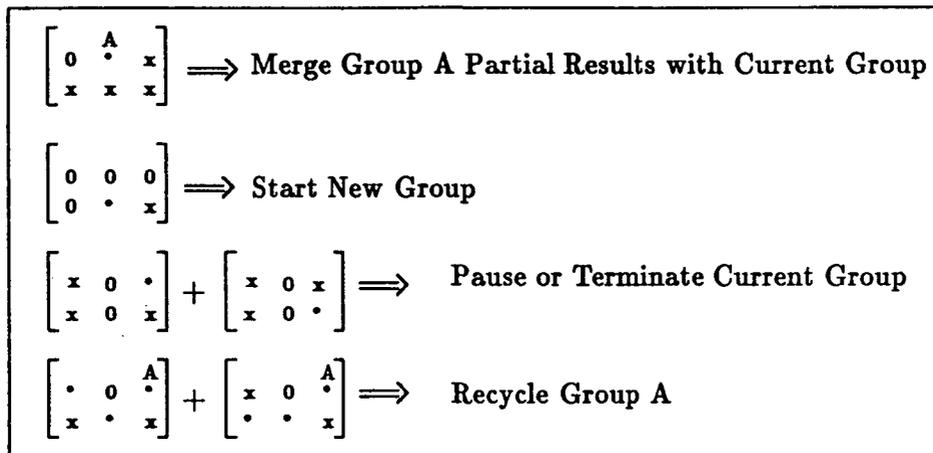


Figure 6: Primary Trigger Patterns

20 bit adder, two 20 bit registers and two 10 bit position counters. Cells designed for the *calculator* were used to implement this machine.

3 Pixel Grouper

3.1 Overview

Pixels entering ACE are first masked according to the information contained in the external mask RAM and then compared to a threshold value programmed from the host transputer. Pixels falling below the threshold are set to zero, the rest are processed as constituents of some group of pixels. Critical in this process is an efficient, readily implementable algorithm for assigning pixels to separate groups. Such an algorithm was developed for this processor. Though the algorithm used for ACE does not properly group pixels into arbitrarily complicated concave shapes, it does function correctly for all convex shapes, as well as most concave shapes. Complete generality was sacrificed in favor of an algorithm which can be easily implemented at high data rates. Subsequent implementations of ACE may utilize more general algorithms. The algorithm actually used is outlined in the below.

3.2 Algorithm

The algorithm used to group pixels utilizes information about the previous scan line, as well as the pattern of incoming pixels on the current scan line to form pixel groups. In particular, the controller scans the incoming pixels, together with the pixel patterns of the previous scan line for one of six distinct *trigger patterns*. Each *trigger pattern* contains six pixels, three from the previous scan line and three from the current scan line. Each time the incoming pixel pattern matches one of these *trigger patterns*, the controller initiates one of four basic operations. Figure 6 summarizes each of these *trigger patterns* and describes the operation initiated by each pattern.

The first pattern essentially indicates that, at every leading (leftmost) edge of a group in the previous scan line, the partial results calculated for that group are to be merged into the calculator as part of the current group (if one exists). Some preprocessing of the previous scan line information is performed to ensure that shapes consisting of multiple downward "fingers" will be merged into the current group only once. The second pattern causes a new group to be started if a pixel on the current line is not adjacent to any pixel on the previous scan line. It should be noted that it is entirely possible for subsequent occurrences of the first pattern to cause groups existing in the previous scan line to be merged into this new group.

Patterns three and four indicate that the current group should be either terminated (sent to the *output fifo*) or paused (sent to the *scratch ram*). These patterns are the ones that imply another group is about to be (re)started. If no pixels have been added to the current group on this scan line (or if the last scan line in the frame is being processed), it is assumed that the current shape does not extend into subsequent scan lines and it is terminated; otherwise, it is sent back to the *scratch ram* for subsequent processing.

The last set of patterns indicate that a group which existed in the previous scan line has been merged into another group. It presents signals the controller to recycle the merged group into the free group list, so that it may be re-used.

This set of *trigger patterns* or rules is highly successful in grouping pixels into contiguous groups. Failures occur in two situations, however. First, it is possible to fool the machine into merging a group from the previous scan line into the current group more than once. This occurs with shapes having multiple downward "fingers" which are separated by different, non-contiguous groups. The second failure mode occurs when the machine is fooled into prematurely terminating a group. This also occurs only with shapes having multiple downward "fingers" separated by different, non-contiguous groups. If the first downward finger has no new pixels added to it in the present scan line, *trigger patterns* three and four will cause the group to be terminated even though pixels may be added from subsequent fingers.

Both defects in the basic grouping algorithm may be remedied. Zeroing out the contents of the *scratch ram* associated with a group each time it is read prevents the first failure mode. Waiting until the end of a scan line before terminating groups by sending them to the *output fifo* remedies the second failure mode. In the present implementation, however, this would also require the addition of a machine which keeps track of whether or not a given group had pixels added to it in the present scan line. It also complicates the *scratch ram* timing considerably.

4 Summary

Highly efficient pixel grouping algorithms were developed for ACE. The required calculations are performed in a specially designed architecture which minimizes signal line interconnect, and maximizes data throughput and computational efficiency. All control, pattern matching, memory management, data flow, input/output, and computational operations

are performed in parallel. These factors, combined with a custom designed layout, result in a very high performance centroid processor.

5 Future Development

A number of extensions are under consideration. Among these are a version with more bits per pixel, and an enhanced pixel grouper. A version which calculates higher moments, and allows the threshold level to be a function of position within the frame is also being considered.

6 Acknowledgments

The work reported herein was performed under the auspices of the U.S. Department of Energy for the Lawrence Livermore National Laboratory by the NASA VLSI Hardware Acceleration Center for Space Research.

References

- [1] T. Axelrod, T. Tassinari, G. Barnes, K. Cameron, "A VLSI Centroid Extractor For Real-Time Target Tracking Applications," SPIE Conference, August 1989.
- [2] R. Brent, H. Kung, "A Regular Layout for Parallel Adders," IEEE Transactions on Computers, Vol. C-31, March 1982, pp. 260-264.
- [3] J. Canaris, K. Cameron, "A Comparison of Two Fast Binary Adder Configurations," NASA SERC 1990 Symposium on VLSI Design, University of Idaho, Moscow, Id, Jan. 1990, pp. 78-86.
- [4] A. Rothermel, et al., "Realization of Transmission-Gate Conditional-Sum (TGCS) Adders with Low Latency Time," IEEE JSSC, Vol. 24, June 1989, pp. 558-561.
- [5] N. Weste, K. Eshraghian, *Principles of CMOS VLSI Design*, Reading, Mass., Addison-Wesley, 1985, pp. 344-348.

Vertical Bloch Line Memory

R. Katti, J. Wu and H. Stadler
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Extended Abstract-

Vertical Bloch Line (VBL) Memory is a recently conceived, integrated, solid-state, block-access, VLSI memory which offers the potential of $1\text{Gbit}/\text{cm}^2$ a real storage density, gigabit per second data rates, and submillisecond average access times simultaneously at relatively low mass, volume, and power values when compared to alternative technologies. VBLs are micromagnetic structures within magnetic domain walls which can be manipulated using magnetic fields from integrated conductors. The presence or absence of VBL pairs are used to store binary information. At present, efforts are being directed at developing a single-chip memory using $25\text{Mbit}/\text{cm}^2$ technology in magnetic garnet material which integrates, at a single operating point, the writing, storage, reading, and amplification functions needed in a memory. This paper describes the current design architecture, functional elements, and supercomputer simulation results which are used to assist the design process.

The current design architecture uses three metal layers, two ion implantation steps for modulating the thickness of the magnetic layer, one ion implantation step for assisting propagation in the major line track, one NiFe soft magnetic layer, one CoPt hard magnetic layer, and one reflective Cr layer for facilitating magneto-optic observation of magnetic structure. Data are stored in a series of elongated magnetic domains, called stripes, which serve as storage sites for arrays of VBL pairs. The ends of these stripes are placed near conductors which serve as VBL read/write gates. A major line track is present to provide a source and propagation path for magnetic bubbles. Writing and reading, respectively, are achieved by converting magnetic bubbles to VBLs and vice versa. The output function is effected by stretching a magnetic bubble and detecting it magnetoresistively.

Experimental results from the past design cycle created four design goals for the current design cycle. First, the bias field ranges for the stripes and the major line needed to be matched. Second, the magnetic field barrier between the stripe and the read/write gates needed to be reduced. Third, current conductor routing needed to be improved to reduce occurrences of open-circuiting, short-circuiting, and eddy-current shielding. Fourth, a modified Co-alloy was needed with an increased coercivity and controlled magnetization to allow VBL stabilization to occur without affecting stripe stability.

1 Introduction

Vertical Bloch line (VBL) memory [1],[2],[3] is a solid-state, radiation-hard, nonvolatile, block access, magnetic VLSI memory. Research and development efforts for this novel memory are being pursued in the United States, Europe, and Japan. Table 1 shows the potential storage density that is achievable with VBL memory. The densities are a function of stripe width and line feature width, which are defined respectively by the magnetic garnet material and the lithographic process.

	$L_f = 1\mu m$	$L_f = 0.5\mu m$	$L_f = 0.1\mu m$
$s_w = 5\mu m$	10Mbits/cm ²	20Mbits/cm ²	100Mbits/cm ²
$s_w = 2\mu m$	25Mbits/cm ²	50Mbits/cm ²	250Mbits/cm ²
$s_w = 1\mu m$	50Mbits/cm ²	100Mbits/cm ²	500Mbits/cm ²
$s_w = 0.5\mu m$	100Mbits/cm ²	200Mbits/cm ²	1000Mbits/cm ²
$s_w = 0.25\mu m$	200Mbits/cm ²	400Mbits/cm ²	2000Mbits/cm ²

In a VBL memory, information is stored using VBL pairs in magnetic stripe domains in magnetic garnets. The presence or absence of a vertical Bloch line pair in a bit-cell location defines a binary "1" and "0," respectively. Input to the chip is performed by converting currents into magnetic bubbles and then into VBL pairs. Output sensing is performed by converting VBL pairs into magnetic bubbles and sensing magnetic bubbles magnetoresistively.

2 Present Device Design

2.1 Fabrication

The present design uses the magnet garnet, $(BiYGdHoCa)_3(FeGeSi)_5O_{12}$, as the storage medium. The thickness, stripe width, collapse field, saturation magnetization, and anisotropy field of the film is approximately $2.4\mu m$, 230 Oe, 450 Oe, and 1800 Oe, respectively. The film is grown epitaxially on a non-magnetic gadolinium-gallium-garnet (GGG) substrate. These films are transparent but also have a large Faraday rotation, so that magnetic stripes, magnetic bubbles, and, under certain conditions, VBLs can be observed magneto-optically with polarized light using the Faraday effect in a polarized light microscope.

The magnetic garnet has perpendicular magnetic anisotropy so that magnetization lies perpendicular to the film plane, with the bulk of the film magnetized in one direction, and the stripes in the opposite direction. A magnetic domain wall is the boundary between the stripe magnetization and the magnetization of the rest of the film. A twist of magnetization in the domain wall in the plane of the film is a VBL, and two such twists form a VBL pair. If the chirality, or sense of rotation, of the VBLs in the wall is the same, the VBL pair is stable, with a size calculated to be much less than $1\mu m$. The VBL pair is bound together energetically by VBL demagnetizing field energy and magnetic exchange energy.

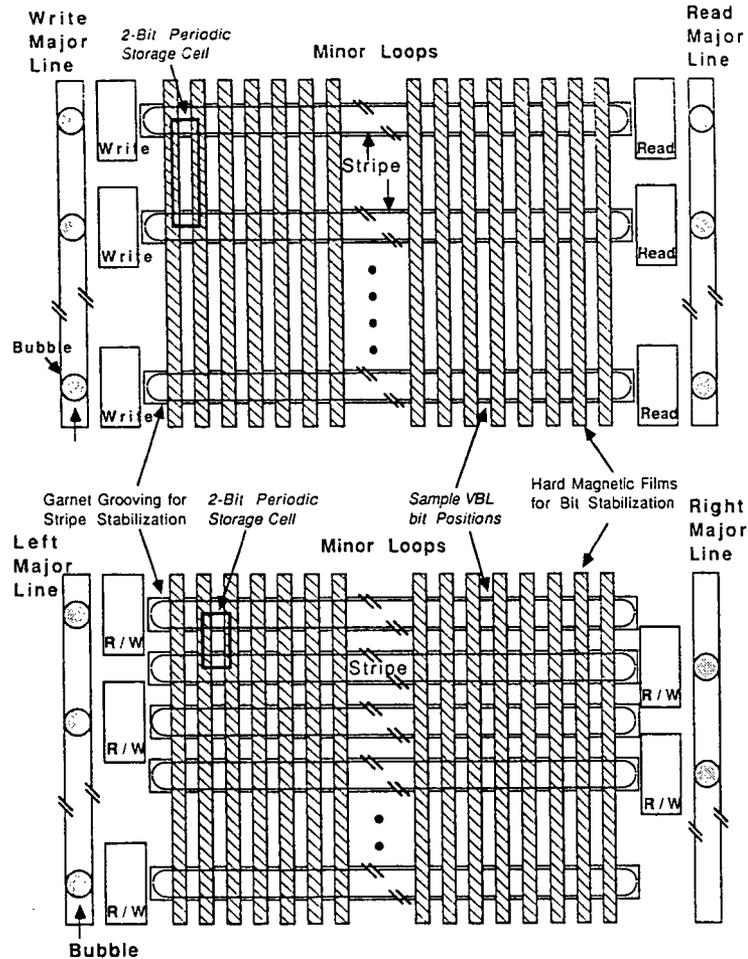
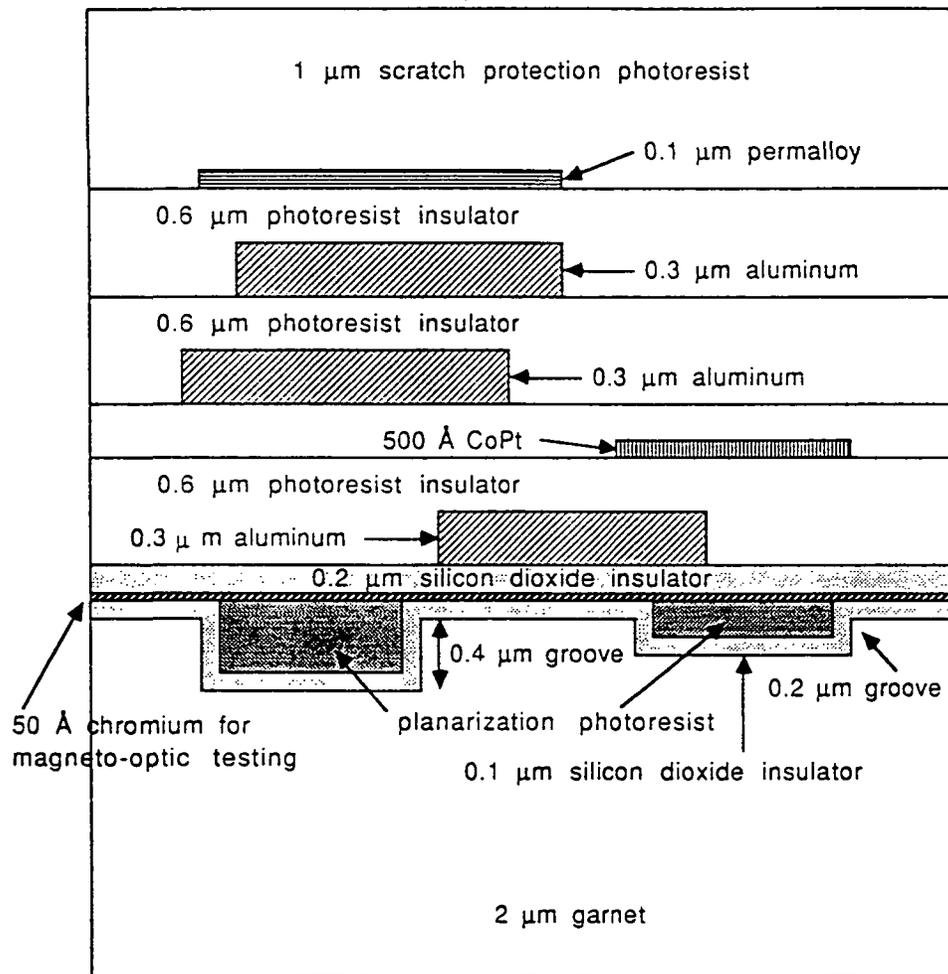


Figure 1: Schematic VBL architecture

The present device is divided into four main functional areas. First, the VBL storage area is designed to confine and stabilize stripe domains. Second, read/write gates are needed to convert VBLs to bubbles and vice versa. Third, a major line is needed for propagating bubbles which are used for input and output. Fourth, the output detector is needed for generating the output signal voltage. Typical architectures for a VBL chip are shown schematically in Figure 1

The present device is built with ten mask layers as shown in Figure 2. Three metal mask layers are used for providing the contact pads and conductors which control the stripe generator, read/write gates, bubble generator, and major line. SiO_2 and photoresist are used for insulating metal layers, and a window mask is used to open vias and contacts when needed between metal layers. Two ion implantation mask steps, using 150keV Ne^+ ions at a dosage of $4 \times 10^{15}\text{ ions/cm}^2$, are used so that, after etching, grooves which are $0.2\mu\text{m}$ and $0.4\mu\text{m}$ thick are created in the garnet film which create stable locations for VBL stripe domains in the presence of the applied bias field. An additional ion implantation mask layer is used in the major line to inhibit VBL formation in the input/output bubbles. A permalloy ($\text{Ni}_{0.8}\text{Fe}_{0.2}$) mask deposition is used to develop a magnetoresistive sensor which

Process Option

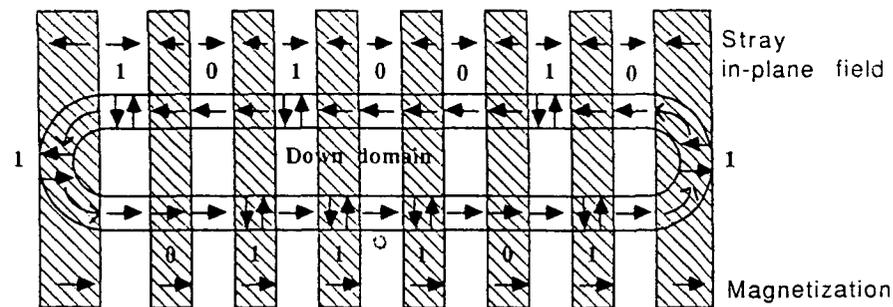


Note:

1. Total of 10 masks are required.
2. Planarization is done by coating 0.6 μm photoresist, thermal flow the photoresist, then etch back.
3. Window mask is used five times to pattern all the photoresist and the scratch protection insulation layers.

Figure 2: Current VBL fabrication process cross-section

Counter-clockwise stripe without static in-plane field



In-plane field stabilized stripe

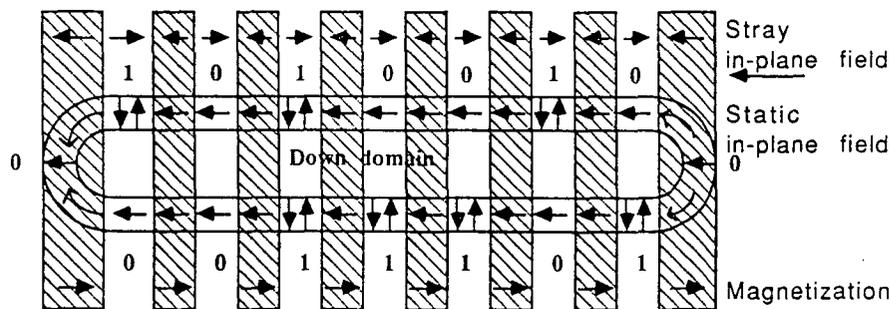


Figure 3: Schematic of VBL stripe storage structure

senses magnetic bubbles at the output and produces output voltages. A cobalt-alloy (i.e. CoPt) mask deposition is used for creating bit cells for VBL pairs along the walls of stripe domains. In test chips, a 5 nm thick Cr mirror layer is used to assist in stripe, bubble, and VBL observation using the magneto-optic Faraday effect during testing.

Supercomputer simulations [4],[5] which compute the effect of magnetic fields on magnetic domains, such as with stripe grooving, the major line, and the major line expander, are used heavily to assist in device design. VBL chip layouts are performed on an HP workstation and IBM PC/AT computers. Layouts are converted into fabrication masks via CIF and GDS II formats.

2.2 Stripe and Bit Stabilization

Data in the form of VBL pairs are stored in the domain walls of arrays of stripe domains as depicted in Figure 3. The stripes are physically located in grooved regions in the garnet, as shown in the partial design layout in Figure 4, which allows selecting the bias field so that stripes are stable when the other chip functions, including the major line, are operating. The demagnetizing field produced by the bias field at the edge of grooved regions serves to attract the stripe end and produce a stabilizing, effective edge-affinity magnetic field. The formation of stable stripes from nucleated isolated bubbles in grooved garnet is shown in Fig. 5. Computations were performed at 1 nsec time steps, and the computed stripe domain shape is shown at 40 nsec intervals.

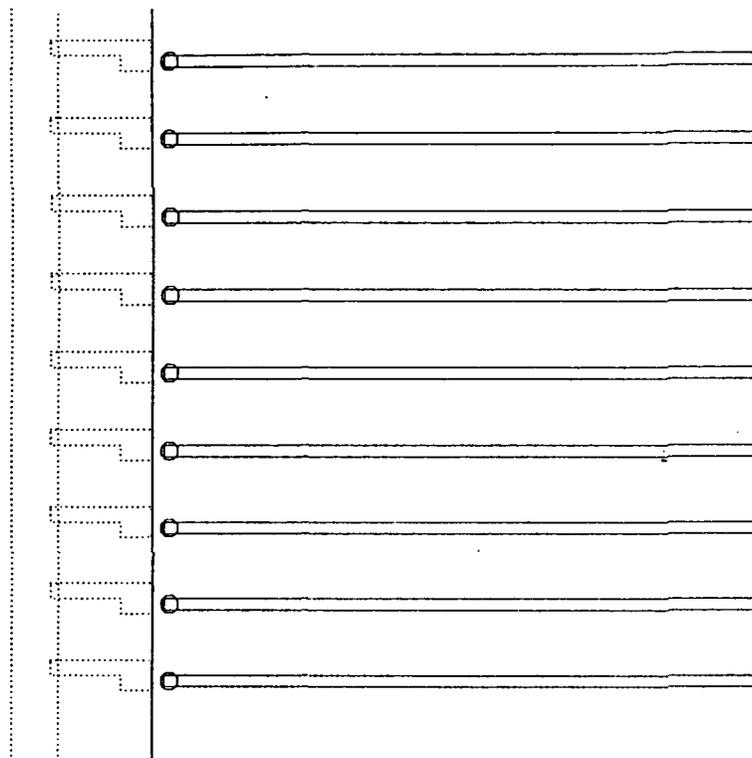


Figure 4: Layout of Garnet Grooving Structure for Stripe Stabilization

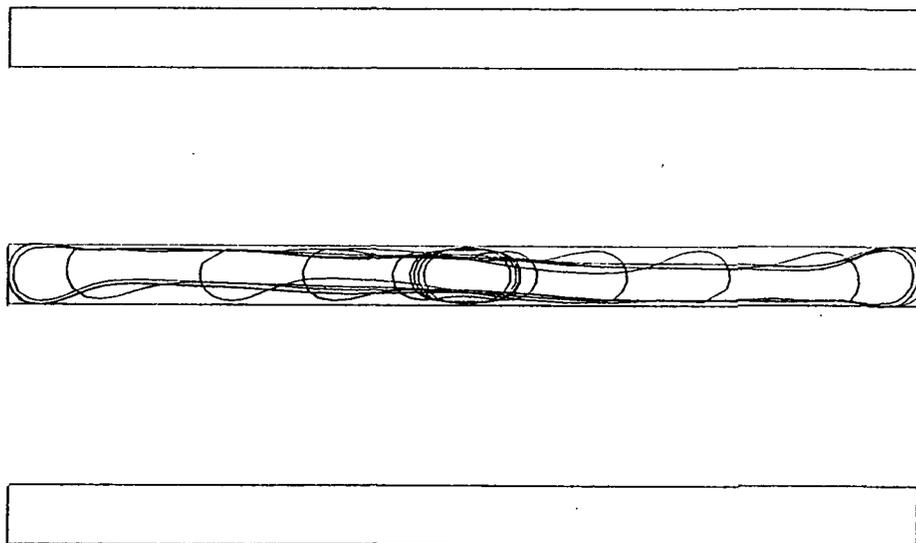


Figure 5: Supercomputer Time-Evolution Simulation of Stable Stripe Formation

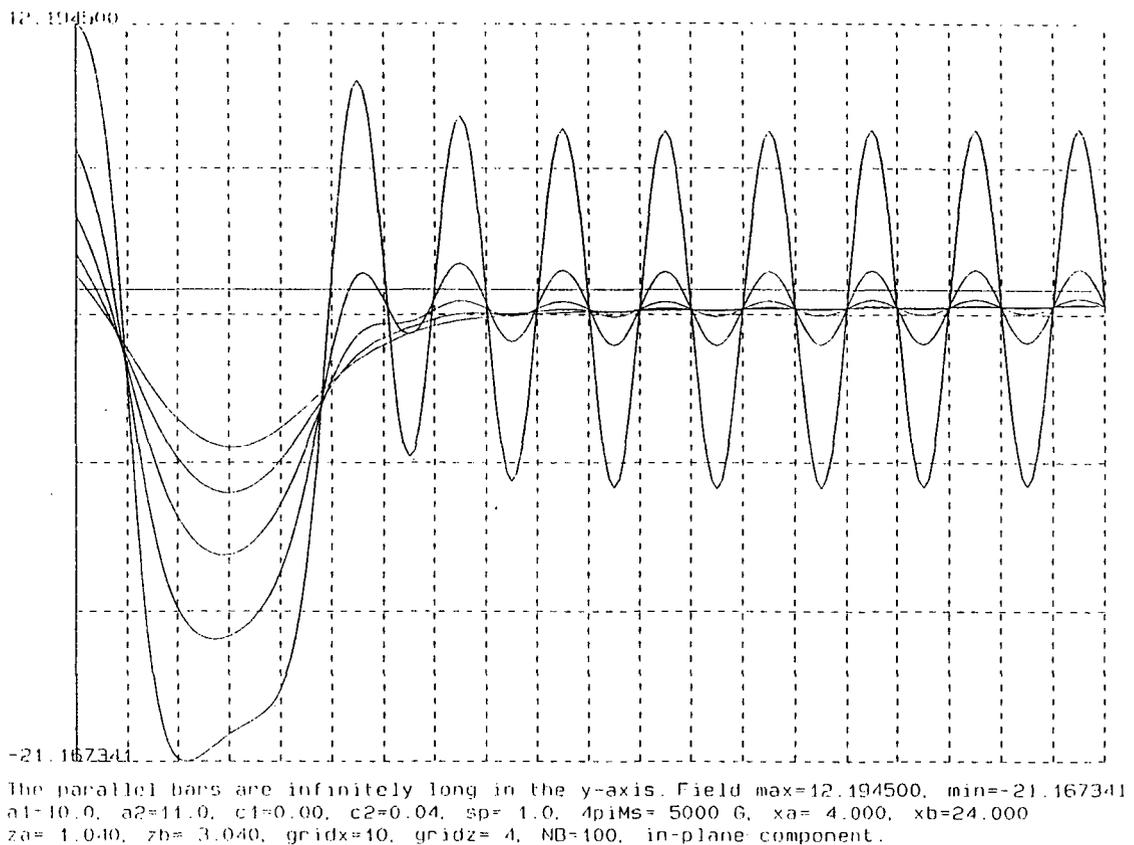


Figure 6: Computed Periodic Magnetic Field Profile of VBL Bit-Stabilization Cells

Bit stabilization is used to stabilize VBL pairs along the stripe. A periodic potential is placed along the stripe by an array of CoPt bars. For the CoPt, the saturation magnetization, coercivity, geometry, and spacing from the garnet are chosen to provide a sufficient field of approximately 5 Oe at the VBL stripe. This field value is currently considered to be enough to provide fields which create potential wells for the VBL pairs without disrupting the VBL pairs and moving stripes away from their groove-stabilized positions. The computed bit stabilization field profile at the end of an array is shown in Figure 6. The distance between the CoPt and the garnet film is a parameter. The periodicity in the field profile is clearly evident.

Propagation of the VBL pairs, around the bit cells and to the read/write gates, can be achieved in two ways. First, a vertical pulse field can be applied which presses, or rocks, the stripe against the groove wall which gyrotropically causes VBL pairs to propagate down the stripe. Second, an in-plane field can be applied which directly causes VBL pairs to advance along the stripes' walls.

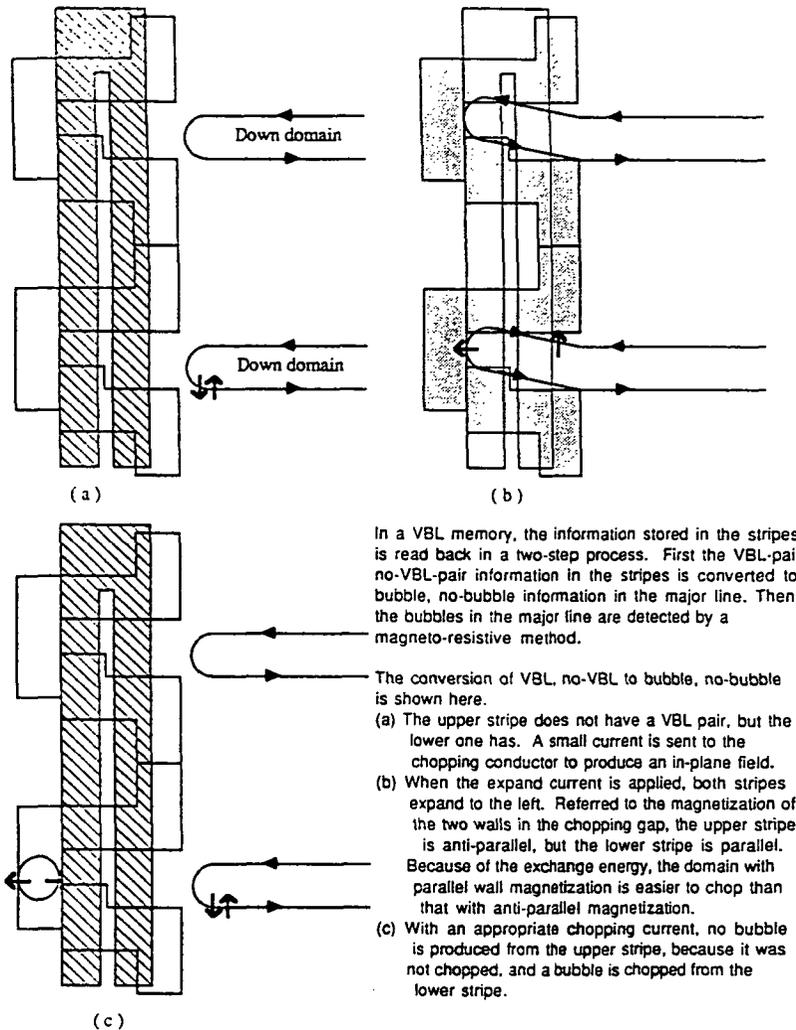


Figure 7: Depiction of VBL read process

3 Read/Write Gates

Read/write gates are used to convert VBL pairs in stripes into bubbles during the read process, and convert bubbles into VBL pairs during the write process. It is necessary to read and write both "1's" and "0's" correctly. Nondestructive readback is achieved by using a current in conductors to bring the end of a stripe out of its groove into the read write gate and into the presence of another conductor, as shown in Figure 7. If no VBL pair is present at the end of the stripe, the sense of the magnetization direction in the stripe wall causes the stripe to be difficult to chop with the field from a conductor because of exchange energy. The stripe is then returned to its stable position in the grooving. However, if a VBL pair is present, one VBL would be brought into the read/write gate while the other VBL would remain in the grooving, so the chirality of the stripe wall would be in the same direction which would readily allow it to be chopped. The chopped portion of the stripe becomes a bubble which can be propagated to the output for sensing, while the stripe returns to the grooved region and recreates a VBL pair leaving the information intact.

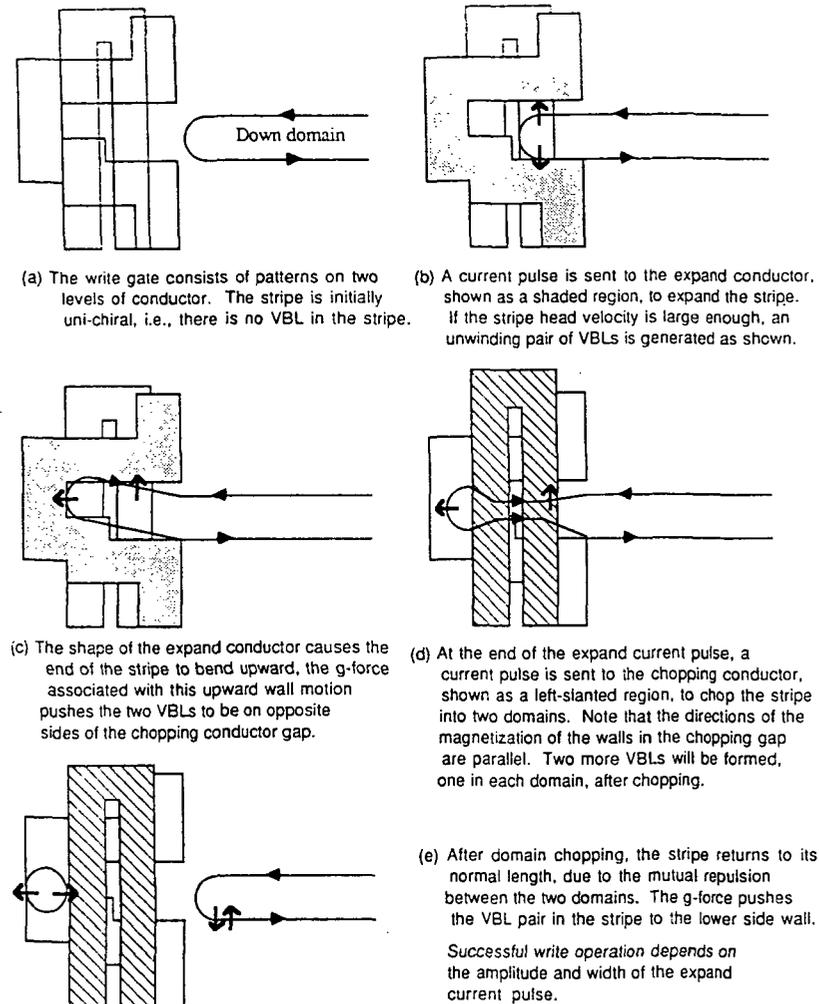


Figure 8: Depiction of VBL pair writing process

Writing is achieved by bringing a bubble from the nucleator and major line to the desired read/write gate. The writing process is shown in Figures 8 and 9. If a bubble is present, when the stripe is subjected to a field to bring it into the read/write gate, the stripe does not get drawn into the read/write gate because of magnetostatic repulsion between the bubble and the end of the stripe. Therefore, no writing to the stripe occurs. But if a bubble is not present, the stripe is brought rapidly into the read write gate which inserts a VBL pair into the stripe, and the stripe is then allowed to relax into the grooving.

3.1 Major Line and Output Detector

The major line consists of a bubble nucleator for converting input signal currents into bubbles, a track for propagating bubbles from the nucleator to the read write gates and from the read write gates toward the output, an output detector for converting the demagnetizing field from bubbles into output voltages, and an expander which is used to stretch a bubble to a desired length to provide a satisfactory signal-to-noise ratio at the output. These structures are now discussed.

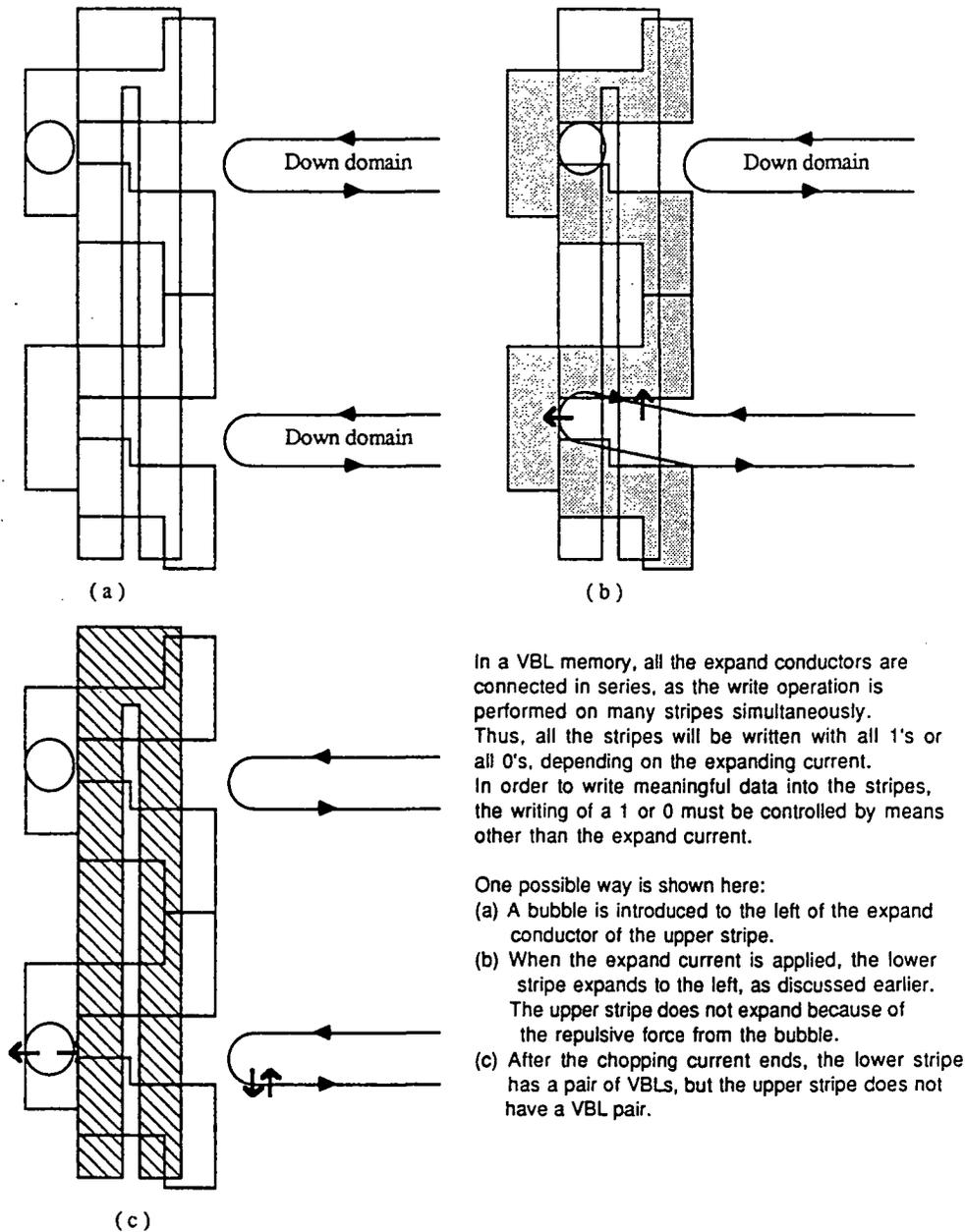
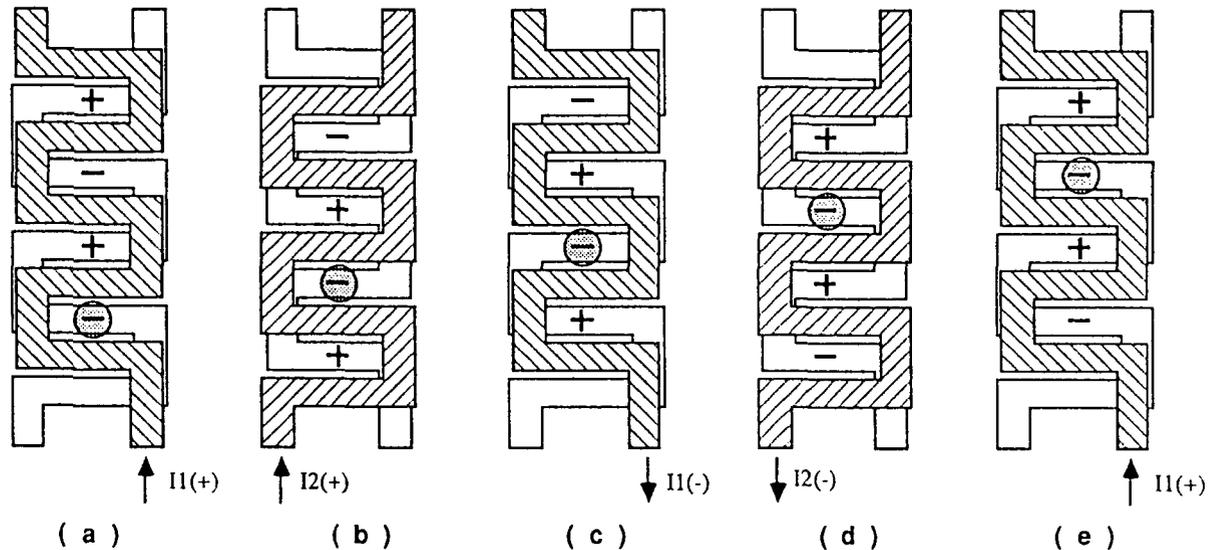


Figure 9: Depiction of binary VBL write process



A current in a zig-zag conductor, as shown in (a), produces an alternating magnetic field pattern which attracts or repels bubbles in sites along the conductor. By staggering two conductor patterns as shown, and alternating the current sent to each conductor, a track is defined which can propagate bubbles. Due to the low inductance of the conductor sheet, such a propagator can be operated at a very high frequency.

Figure 10: Principle of bubble propagation in a dual-conductor major line

A hairpin conductor is used for the nucleator. When a current is applied, such a conductor produces a magnetic field which is concentrated at the center of the hairpin. This field is used to generate bubbles which are used to transmit binary information to the VBL stripes via the read/write gates.

The propagation track in the major line consists of two levels of conductors. Each conductor is a serpentine arrangement of hairpin conductors which provide local magnetic field variations which form "waves" of stable positions for the bubbles down the track. The conductors are physically phase shifted by 90 to effect propagation. The principle of operation of the major line is shown in Figure 10. The layout of the major line track, along with the bubble nucleator and two read/write gates and grooves, is shown in Figures 11 and 12. Figure 12 shows the functional, two-metal major line and Figure 11 shows the major line with just one metal to help visualization.

The output detector consists of a rectangular strip of permalloy which is magnetoresistive. When a magnetic field as from a magnetic bubble is placed near the sensor, the resistance changes. When a reference current is issued to the sensor, the presence or absence of a bubble induces two different voltage levels to be generated which define binary

8.3.12

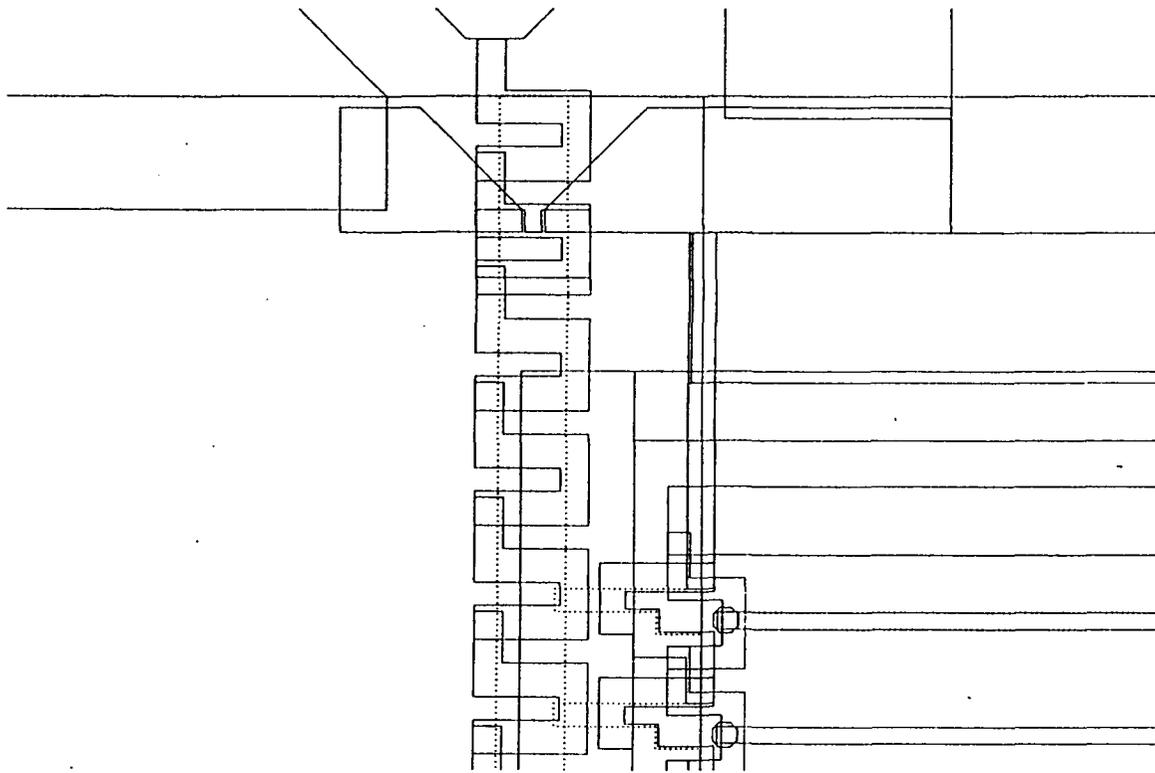


Figure 11: Layout showing nucleator, major line and two read/write gates and tow stripe grooves. only one major line conductor is shown for visibility

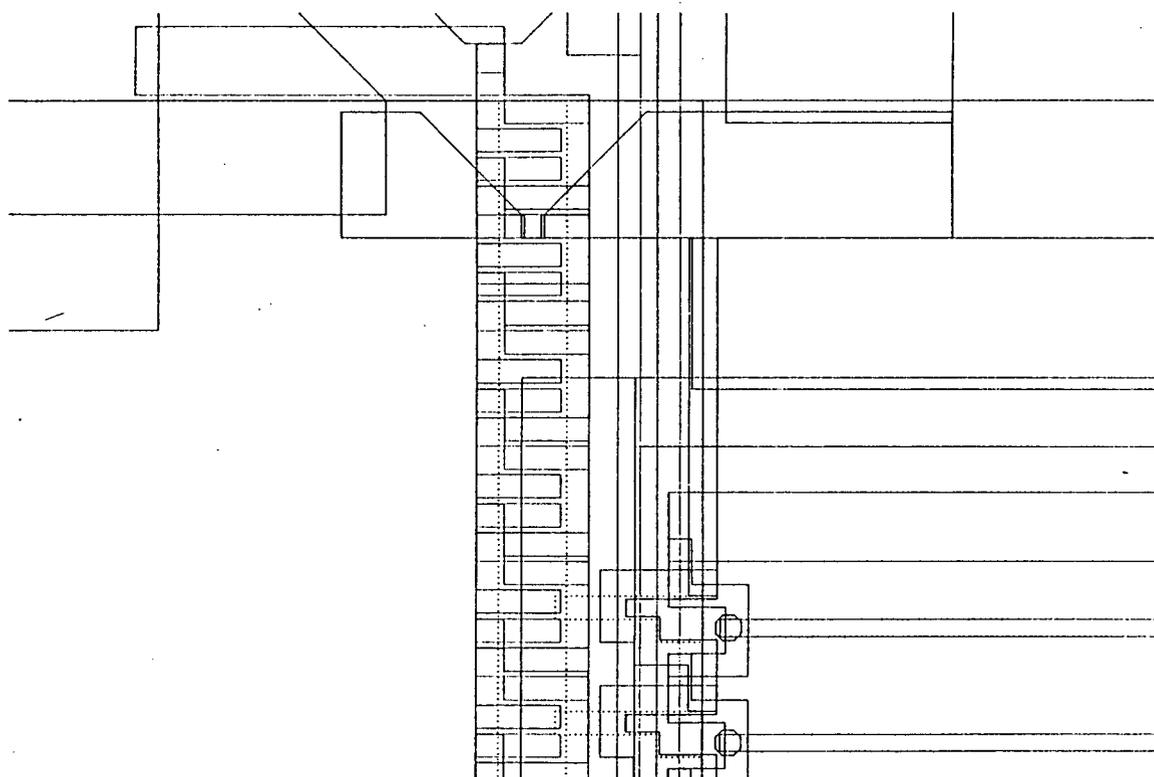


Figure 12: Layout showing nucleator, major line and two read/write gates and tow stripe grooves. Both major line conductors are shown

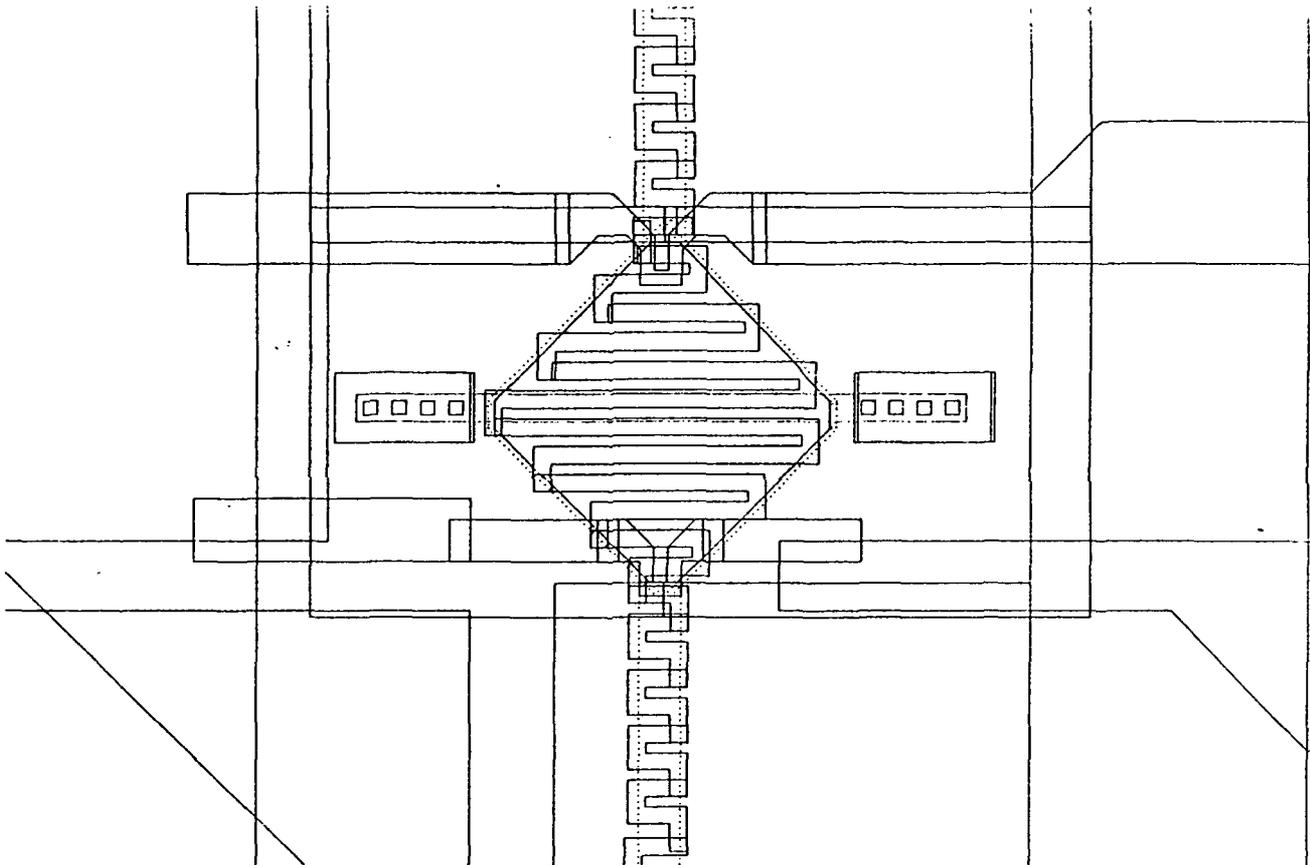


Figure 13: Layout of a VBL expander and output detector

"1's" and "0's." If it is necessary to maximize common-mode rejection from stray magnetic fields, two magnetoresistive sensors, including the actual sensor and a dummy sensor, are used which can be measured differentially. Since the signal from the magnetoresistive sensor is increased if longer elements with greater electrical resistance are used, the sensor length can be increased. Bubbles can be stretched in length to provide fields for the lengthened sensor by widening the bubble track as the bubble approaches the sensor. The combined output sensor and expanded major line are shown in Figure 13, where only one metal is shown in the major line for clarity. A supercomputer simulation, shown at 40 nsec intervals after being computed at 1 nsec intervals, of a bubble being stretched and then unstretched as it passes the expander and detector is shown in Figure 14.

4 Experimental Results

Sample experimental results of chip functions from past designs are now presented. [6],[7] These data were taken on test chips using a $4.76\mu\text{m}$ thick $(\text{BiYSmLu})_3(\text{FeGa})_5\text{O}_{12}$ garnet

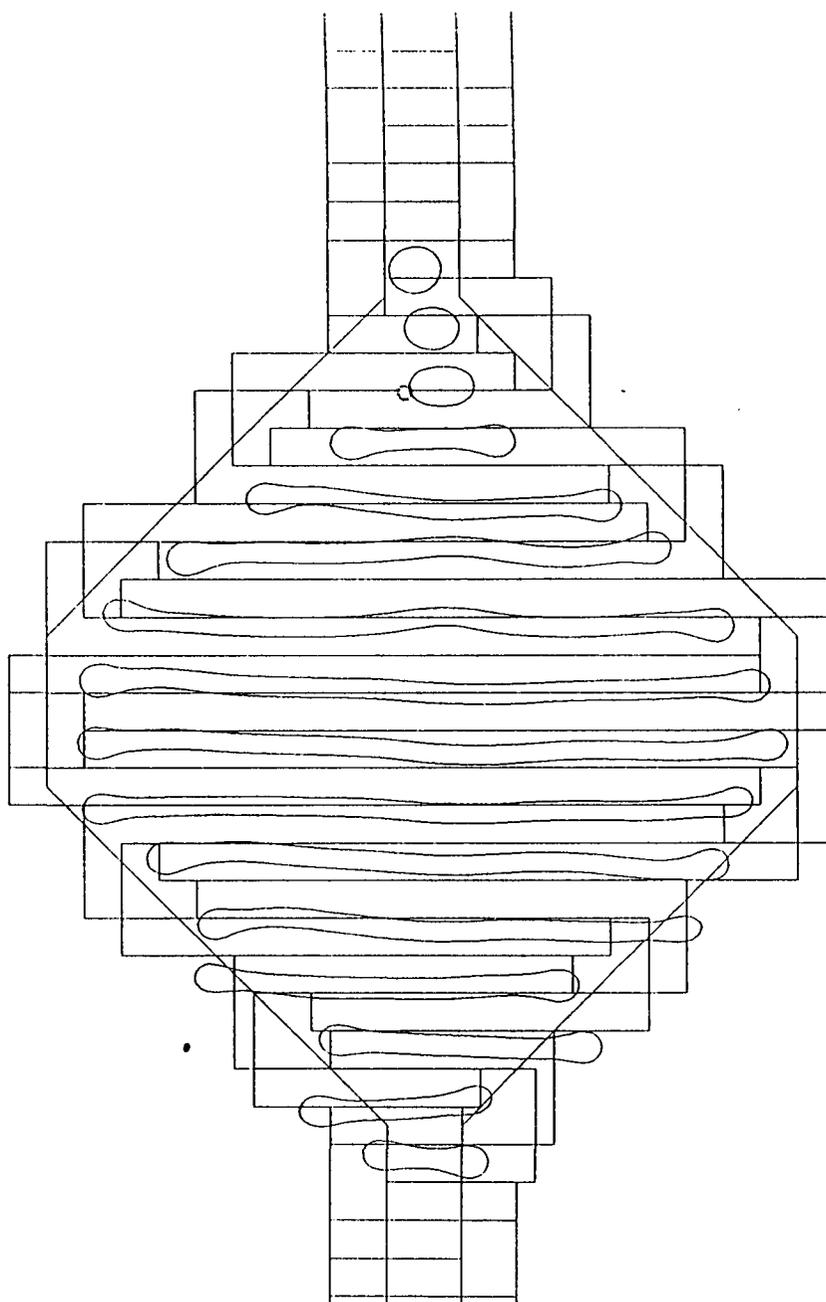


Figure 14: Supercomputer time evolution simulation of an expanding bubble in a major line expander and output detector

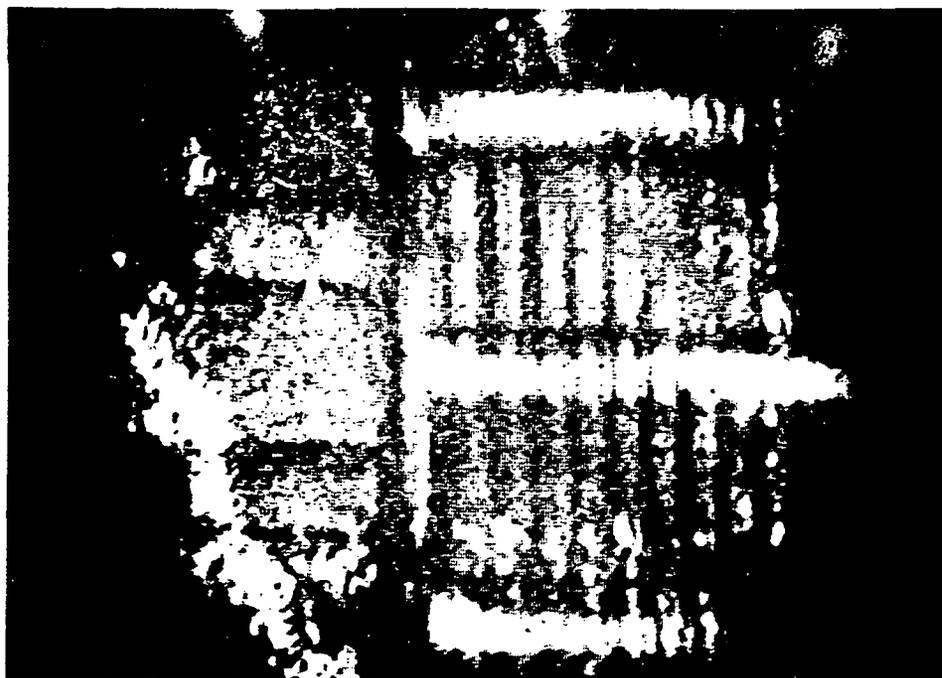


Figure 15: Photograph of stable stripes near the read/write gate end in a VBL test chip

film with a saturation magnetization of $200Oe$, zero-field stripe width of $4.67\mu m$, characteristic length of $0.61\mu m$, domain wall mobility of $350cm/sec/Oe$, anisotropy field of $1350Oe$, coercivity of $1.1Oe$, and collapse field of $100Oe$.

Figures 15, 16, and 17 show stabilized stripes in grooves, respectively, near read/write gates at the end of a groove, near the center of the stripe groove, and near the other end of the stripe away from a read/write gate. The stripes are observed with the magneto-optic Faraday effect as white strips against a green background caused by the illuminating laser light source. The dark bars running perpendicular to the stripes are cobalt bit stabilization bars. Figure 16 shows the conductor that was used to nucleate the stripes.

Figure 18 shows, in white, a nucleated bubble and a second bubble propagating along the major line. The nucleated bubble is large, because of the large reverse magnetic field which produces the bubble. As the bubble propagates away from the nucleator, the bubble diameter reduces to a nominal diameter dictated by the film and the applied bias field. Figure 19 shows bubbles in the major line further down the track. Bubbles and stretched bubbles in the expander portion of the major line near the output detector are shown as white spots and strips Figure 20.

5 Conclusion

VLSI designs, simulation results, and experimental results have been presented which describe current work on the storage and input/output functions for solid state, high density, nonvolatile, radiation hard, block access Vertical Bloch Line (VBL) memory. Such a memory offers the potential of achieving $1Gbit/cm^2$. Three metallizations are used along with three ion implantation steps, one permalloy magneto-resistive sensor deposition, and one

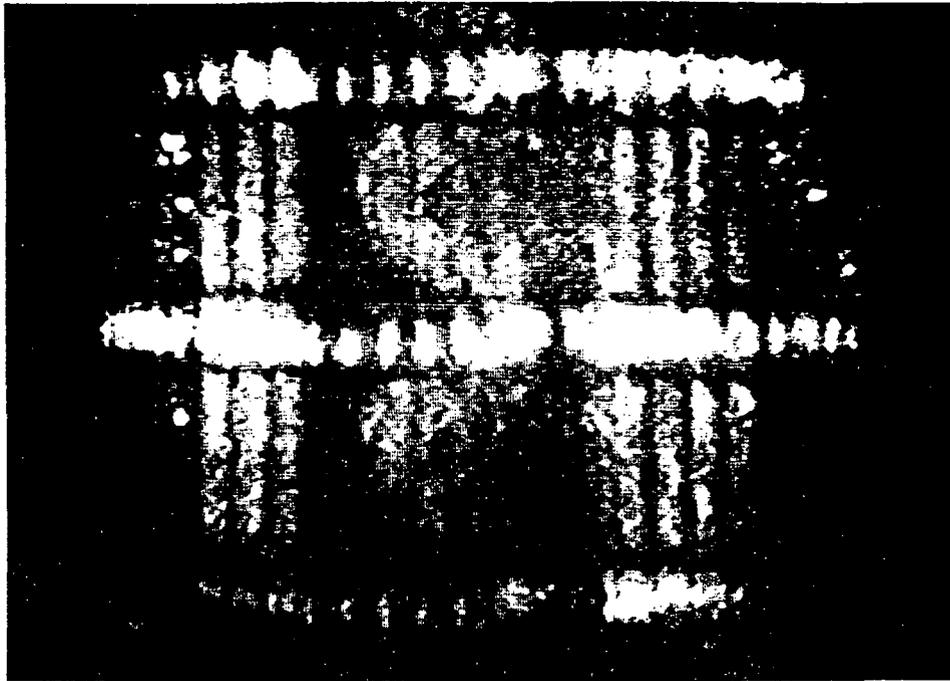


Figure 16: Photograph of stable stripes at the stripe's center near the stripe nucleator in a VBL test chip

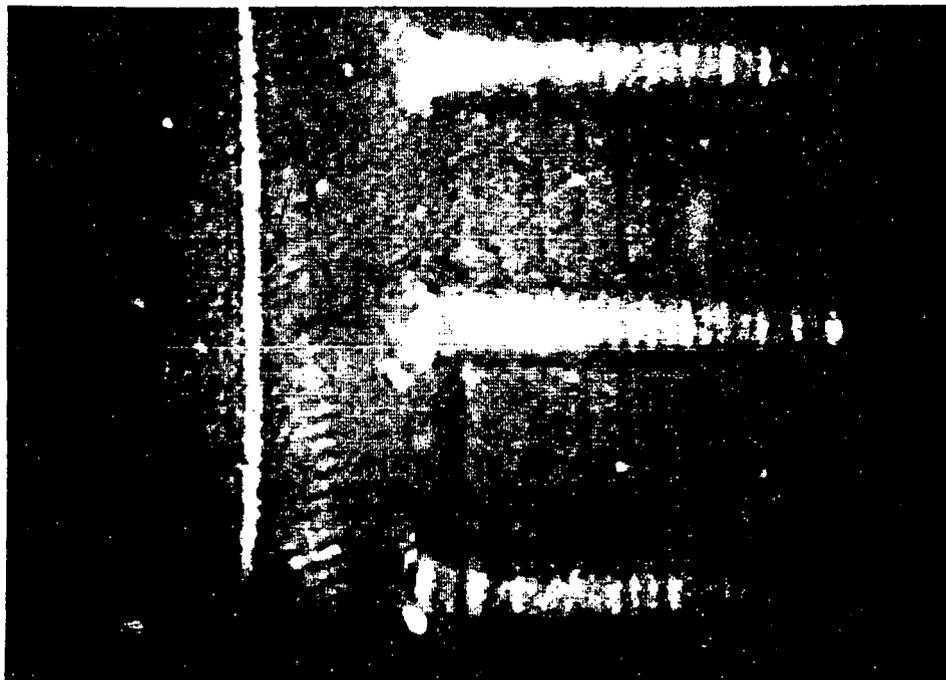


Figure 17: Photograph of stable stripes at an end away from read/write gates in a VBL test chip

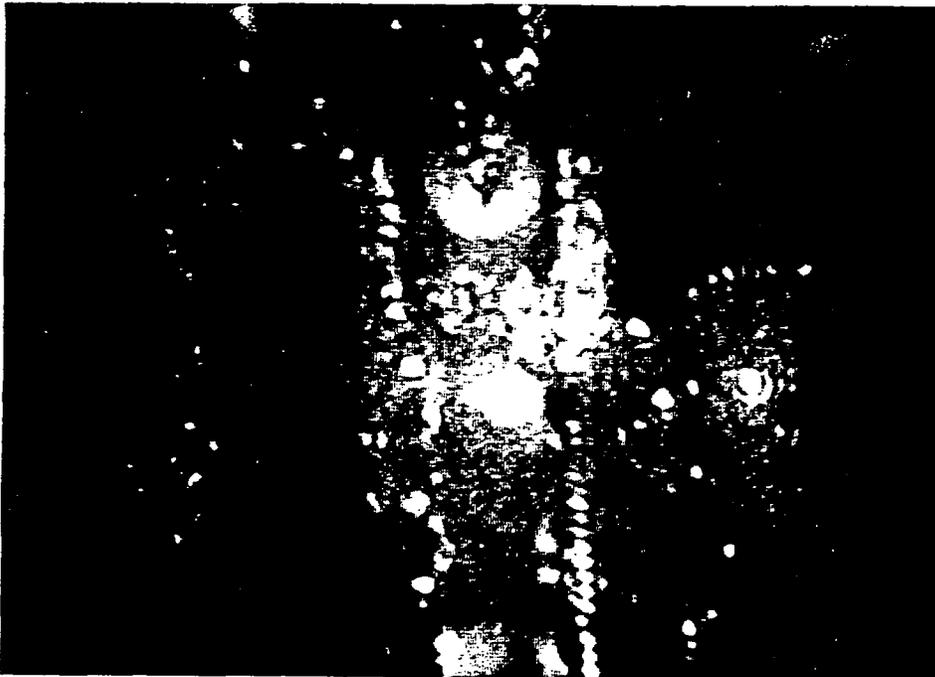


Figure 18: Major line and nucleator under operation

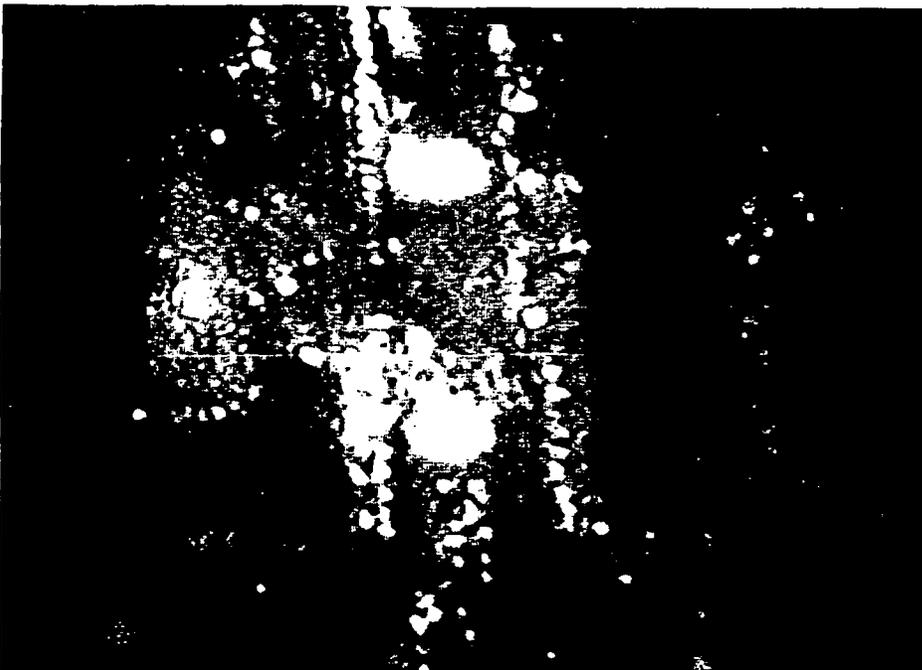


Figure 19: Major line under operation near the center of the major track



Figure 20: Major line and the major line expander under operation

CoPt bit-stabilization deposition. Previous and current experimental results and super-computer simulations indicate that individual storage, read/write gate, and input/output functions are feasible. Present work is aimed at integrating all necessary memory functions on a single chip to achieve simultaneous operation, a unique operating point, and a fully-functional single chip memory.

Acknowledgements:

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 in part by the Strategic Defense Initiative Organization Innovative Science and Technology Office. The use of the JPL/Caltech CRAY X-MP/18 for performing portions of this work is acknowledged.

References

- [1] S. Konishi. "A New Ultra-High-Density Solid State Memory: Bloch Line Memory", *IEEE Transactions on Magnetics*, Vol. 19, No. 5, pp. 1838-1840, September, 1983.
- [2] T. Suzuki, et.al. "Chip Organization of Bloch Line Memory," *IEEE Transactions on Magnetics*, Vol. 22, No. 5, pp. 784-789, September, 1986.
- [3] H. Matsutera, et.al. "Design and Characteristics for Vertical Bloch Line Memory Using Ring-Shaped Domain," *IEEE Transactions on Magnetics*, Vol. 23, No. 5, pp. 2320-2325, September, 1987.

- [4] J. C. Wu. "Solid-State Magnetic Devices: Magnetic Bubble Logic Devices and Vertical Bloch Line Memory," Ph.D. Thesis, Carnegie Mellon University, August, 1986.
- [5] N. Hayashi and K. Abe. "Computer Simulation of Magnetic Bubble Domain Wall Motion," Japanese Journal of Applied Physics, Vol. 15, No. 9, pp. 1683-1694, September, 1976.
- [6] J. C. Wu, et.al. "Major Line Operation in Vertical Bloch Line Memory," Paper FP-10, Accepted for Presentation, 1990 Magnetism and Magnetic Materials Conference, San Diego, CA, October, 1990.
- [7] J. C. Wu, et.al. "Stripe Stabilization in Vertical Bloch Line Memory," Paper FP-9, Accepted for Presentation, 1990 Magnetism and Magnetic Materials Conference, San Diego, CA, October, 1990.

High Performance Pipelined Multiplier with Fast Carry-Save Adder

Angus Wu

Department of Electrical and Computer Engineering
Washington State University
Pullman, Washington 99164-2752

Abstract - High-performance pipelined multiplier is described. Its high performance is resulted from the fast carry-save adder basic cell which has a simple structure is suitable for the Gate Forest semicustom environment. The carry-save adder computes the sum and carry within two gate delay. Results show that the proposed adder can operate at 200 MHz for a 2-micron CMOS process and expected to have better performance in a Gate Forest realization.

1 Introduction

Multiplication is one of the most common and important operation in signal processing [1]. But, the operation is time consuming. Many efforts have been invested in improving the multiplication speed. Pipelined multipliers with different architecture were proposed to enhance the performance [2-5]. However, most of the works has been devoted to improve the performance in system level. Little work has been done to enhance the system performance in reducing the propagation delay of the basic computation cell in circuit level. It is well known that the performance of a digital system not only depends on its architecture, but also on its implementation complexity. To enhance the performance, it is necessary to reduce the implementation complexity as much as possible.

For pipelined multiplier, the essential component is the carry-save adder. The maximum clock speed of the multiplier is determined by the delay time of the basic carry-save adder cell to form and add the partial product, and generate the carry. A carry-save adder with simple implementation complexity will shorten these operation time and enhance the maximum throughput rate of the multiplier directly. To achieve this goal, a high performance pipelined multiplier with fast carry-save adder cell is proposed.

2 System Description

The pipelined multiplier is constructed in as a semi-systolic array [3]. Schematic of the pipelined multiplier array is shown in figure 1. It has three basic components, the carry-save adder, half adder and register. They are organized as three basic arrays of the pipelined multiplier. Two phase clocking scheme is used to control the data flow. It is well known

that the throughput rate of a pipelined multiplier is determined by the delay of its basic cells, in this case, the carry-save adder. To enhance the system performance, it is necessary to reduce the delay of the adder. It can be done either in device technology or reducing the circuit complexity. The first approach is expensive and can not be controlled by the circuit designer. Instead, the second approach is used in this paper.

A fast carry-save adder is designed with simple circuit structure; and, it is suitable for the Gate Forest semicustom environment [6]. Unlike traditional implementations, the adder cell does not use multiple-level of nand gate or complex logic gate to evaluate the sum and carry.

3 Circuit Implementation

Figure 2 shows the circuit schematic of the proposed adder cell. It consists of three functional units: sum evaluation unit, carry evaluation unit and partial product generation unit. The sum is computed by an exclusive-or and equivalence dual pair. Each of the gates has only five transistors constituting one level of logic. Taking the advantage of charge re-distribution, this arrangement has the best performance than using two exclusive-or gate for sum evaluation. At the precharge phase, Φ_1 is asserted. The exclusive-or gate is isolated from the equivalence gate since the inputs of the equivalence gate are pull to ground by the Nfets at node 1 and node 3. At the same time, node 2 is precharged through the Pfet controlled by Φ_2 which is not asserted. When Φ_2 is asserted, charges at node 2 may be re-distributed to node 1 depending on the inputs signal \bar{D} of the equivalence gate. This charge re-distribution phenomena* can speed up the sum evaluation. By carefully design, charges stored at node 2 can be equally sharing with node 1 if the capacitances at node 1 and node are made to be equal. For inputs AB as '01' or '10' together with \bar{D} as '1', the logic '1' signal from either A or B has to propagate to node 2 through the exclusive-or gate and equivalence gate; so that, node 1 and node 2 will be charged to logic '1'. The time needed for both node 1 and 2 to be charged to logic '1' would be reduced significantly. Since node 1 and node 2 are sharing equal charges (at voltage $V_{dd}/2$ ideally) when \bar{D} is at '1'. Thus, the worst case propagation delay of sum evaluation will be reduced. Such delay reduction is also true for the case when logic '0' has to propagate from input signals A or B to the output node of the equivalence gate.

For carry computation, a Manchester carry chain is used to compute the carry. This technique has shown to be very effective for carry propagation in parallel adder. Because of the simple structure, the carry can be computed in two gate delay at most. The partial product term is formed by an And gate embodied in the carry-save adder to eliminate the extra propagation delay needed for the partial product term which is usually generated in the carry-save adder before sum and carry evaluations. In other words, the partial product term generated here is to be used by the next carry saved adder cell. This product term generation scheme has been proved to be very effective [7] in improving the multiplier throughput rate as the critical path delay reduced to the delay of the full adder only.

The half adder is implemented with the exclusive-or gate and And gate. It has the

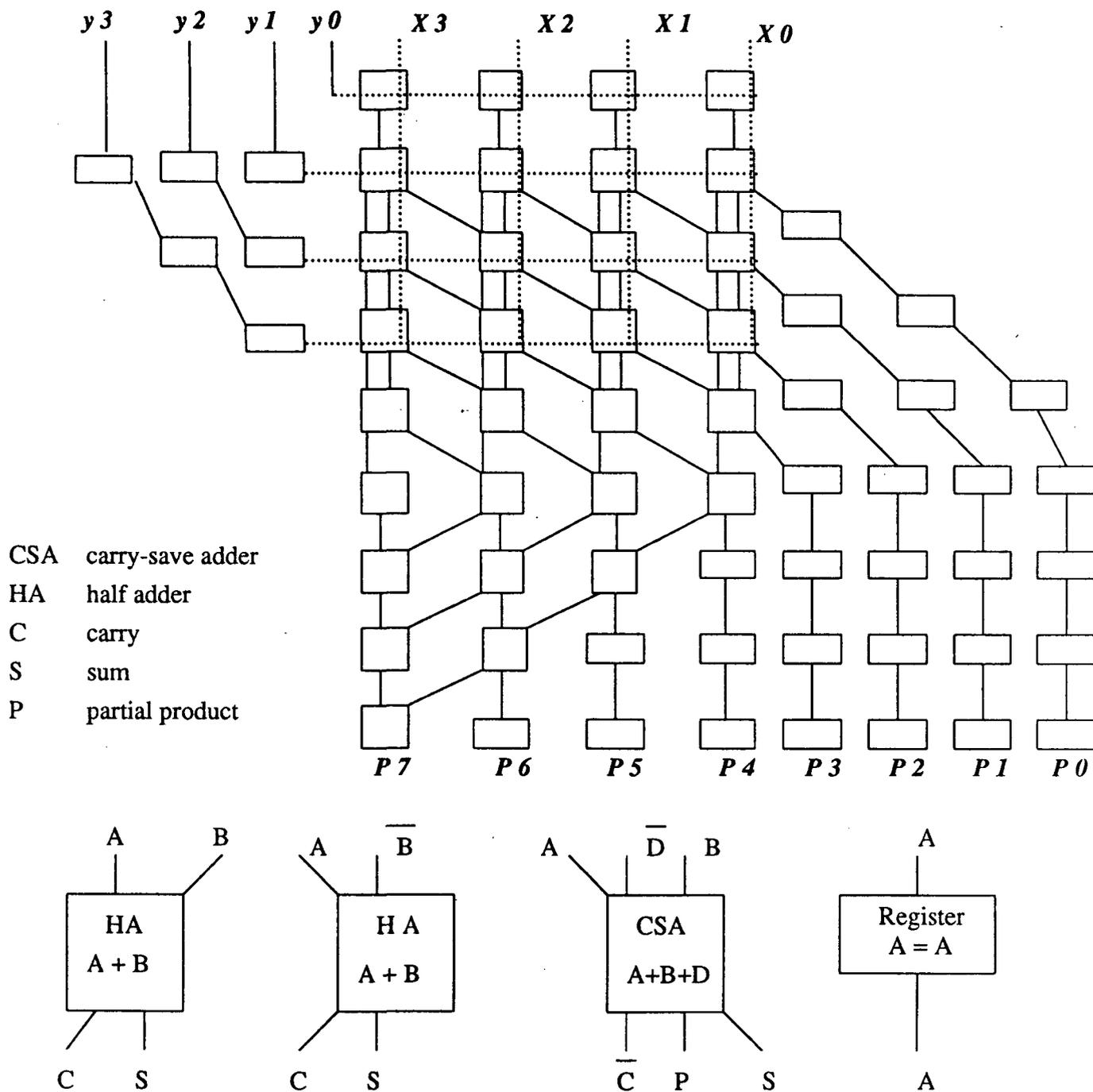


Figure 1: Schematic of 4 × 4 pipelined multiplier

8.4.4

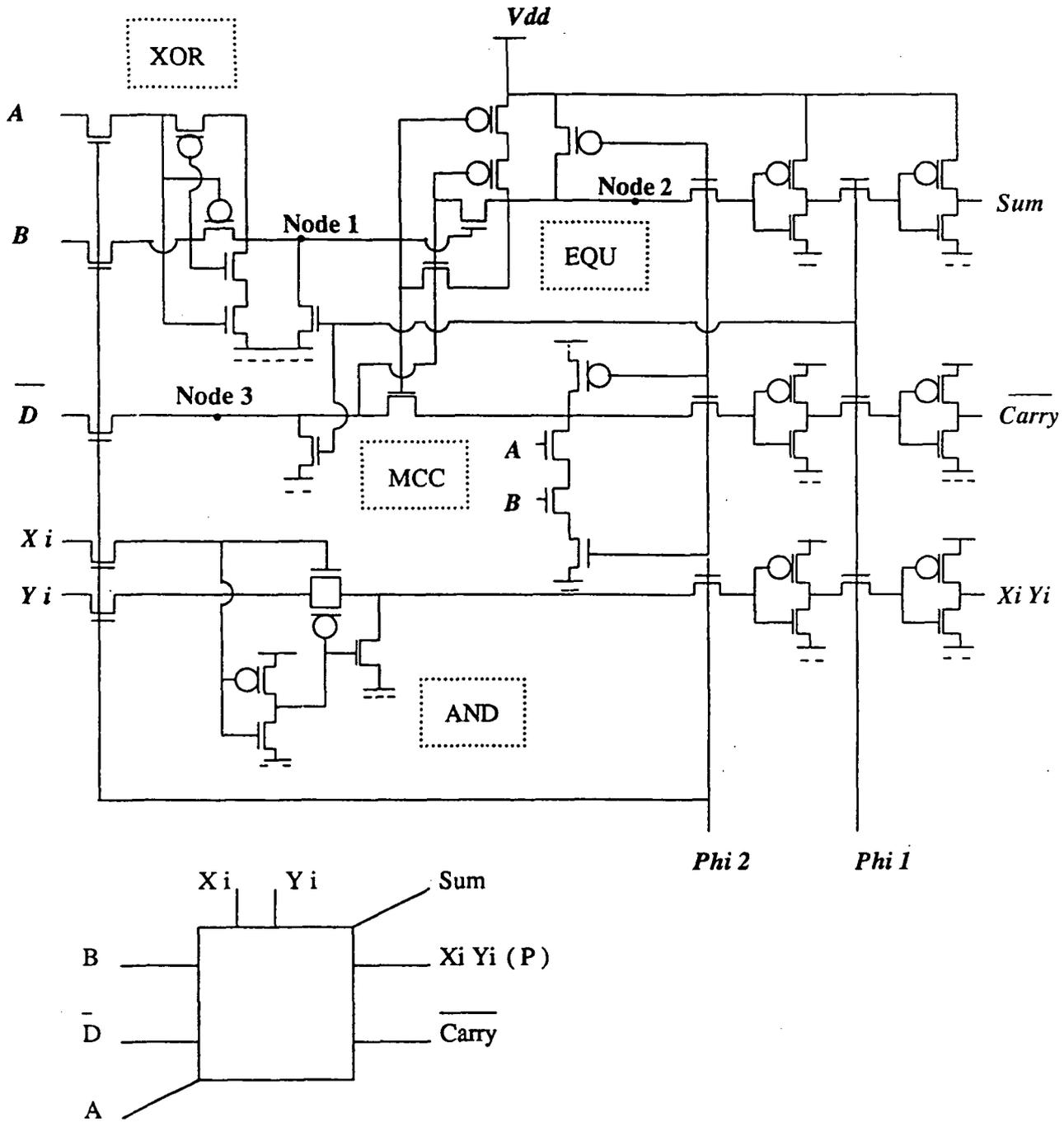


Figure 2: Circuit schematic of the carry-save adder. The basic cell precharges at $\Phi 1$, evaluate sum, carry and partial product

simple structure as the carry-save adder. The register simply consists of inverters for data storage and pass-transistor for data flow control.

4 Circuit Performance

The multiplier is designed and verified. The carry-save adder was simulated with MOSIS 2-micron N-well CMOS device parameters. Results showed that the proposed adder can operate upto 200 MHz (figure 3). Better performance is expected in Gate Forest realization as 0.45 nsec of internal gate delay of ring oscillators was documented [8].

5 Summary

A high performance pipelined multiplier has been described. The performance of the multiplier is enhanced by a novel design of carry-save adder which has a simple circuit structure is suitable for the Gate Forest semicustom environment. Sum and carry can be evaluated at most in two gate delay. With the simple circuit structure and advantage of charge re-distribution, the multiplier can operate upto 200 MHz when simulated with MOSIS 2-micron N-well CMOS device parameters. Better performance is expected when it is implemented in Gate Forest environment.

8.4.6

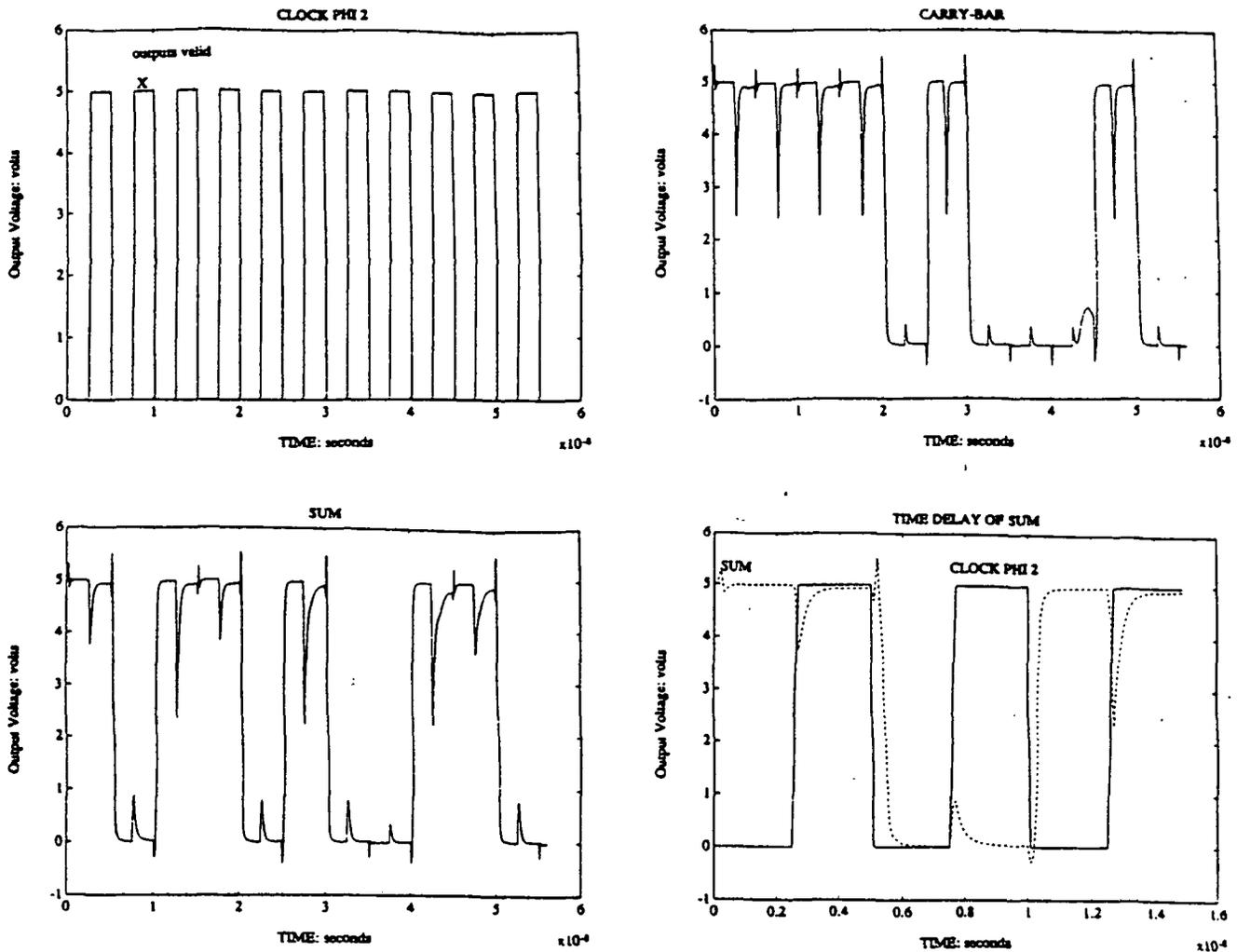


Figure 3: Simulation Results of carry save adder

Input Signals		Output Signals	
A	: 0 1 0 1 0 1 0 1	SUM	: 0 1 1 0 1 0 0 1
B	: 0 0 1 1 0 0 1 1	C	: 0 0 0 1 0 1 1 1
D	: 0 0 0 0 1 1 1 1	\bar{C}	: 1 1 1 0 1 0 0 0

References

- [1] S. L. Freeny, "Special-Purpose Hardware for Digital Filtering," Proc. IEEE, Vol. 63, April 1975, pp. 633-648.
- [2] J. V. McCanny, J. G. McWhirter, "Completely Iterative, Pipelined Multiplier Array Suitable for VLSI," Proc. IEEE., Vol. 129, Pt. G., no. 2, April 1982, pp. 40-46.
- [3] H. T. Kung, "Why Systolic Architectures?," Computers, Jan. 1982, pp. 37-46.

C-6.

- [4] J. R. Jump, S. R. Ahuja, "Effective Pipelining of Digital Systems," *IEEE Trans. Computers*, Vol. C-27, Sept. 1978, pp. 855-865.
- [5] J. Deverell, "Pipeline Iterative Arithmetic Arrays," *IEEE Trans. Computers*, Vol. C-24, March 1975, pp. 317-322.
- [6] M. A. Beunder, J. P. Kernhof and Bernd Hoefflinger, "The CMOS Gate Forest: An Efficient and Flexible High-Performance ASIC Design Environment," *IEEE Journal of Solid-State Circuits*, Vol. 23, no. 2, April 1988.
- [7] T. Noll, D. Schmitt-Landsiedel and G. Enders, "A Pipelined 330-MHz Multiplier," *IEEE Journal of Solid-State Circuits*, Vol. SC-21, no. 3, June 1988.
- [8] J. Kernhof, M. A. Beunder, B. Hoefflinger and Werner Hass, "High-Speed CMOS Adder and Multiplier Modules for Digital Signal Processing in a Semicustom Environment," *IEEE Journal of Solid-State Circuits*, Vol. 24, no. 3, June 1989.

Session 9
Included Papers

Implications of Tracey's Theorem to Asynchronous Sequential Circuit Design

S. Gopalakrishnan, G. Kim and G. Maki
NASA Space Engineering Research Center
for VLSI System Design
University of Idaho
Moscow, Idaho 83843

Abstract - Tracey's Theorem has long been recognized as essential in generating state assignments for asynchronous sequential circuits. This paper shows that 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.

Introduction

Most digital controllers in use are synchronous sequential circuits. As the integration density of VLSI circuits increase, synchronous circuits face clock and power distribution problems. An asynchronous circuit on the other hand does not use a clock and hence the clock distribution overhead is eliminated. Further, the modules within an asynchronous circuit switch over a period of time instead of at a clock edge thereby reducing instantaneous current requirement and the complexity of the power distribution problem. Moreover, asynchronous circuits usually produce the fastest possible circuits as they are not limited in speed by the clock [1]. These advantages would project asynchronous controllers as a viable alternative. The major factor preventing the widespread use of asynchronous sequential circuits is that they involve a complex design procedure which must account for problems like critical races and hazards.

Tracey introduced a fundamental theorem that define the necessary and sufficient conditions for realizing state assignments [2]. This work shows the relationship between Tracey's state assignment procedure and design equations, that can impact the process of generating design equations.

This research was supported in part by the NASA Space Engineering Research Center Program under grant NAGW-1406 and the State of Idaho Grant SBOE 89-041. Dr. Gopalakrishnan is with the California Design Center of Hewlett Packard; Gui-hang Moon Kim is with The Mutual Life Insurance Company

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

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

Asynchronous Sequential Circuit Fundamentals

It is assumed that the asynchronous sequential circuits operate in the fundamental mode [1] and are encoded with single transition time (STT) state assignments [2]. 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 [2,3,4].

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 [3,4]. Tracey's state assignment procedure, on the other hand, partitions the transition pairs under a given input column [2].

Definition 1 *The two block partitions $\tau_1, \tau_2, \dots, \tau_n$, induced by the internal state variables y_1, y_2, \dots, 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.

$$\begin{aligned}
\tau_1 &= \{CDF; ABE\} \\
\tau_2 &= \{BCE; ADF\} \\
\tau_3 &= \{EF; BCDA\}
\end{aligned}
\tag{1}$$

$\tau_1 = \{S_j \dots; S_k \dots\}$ implies that the internal state S_j 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 [2].

Tracey's theorem must be satisfied to eliminate critical races. Moreover, this theorem and the procedures used in this work [2] 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:

$$\begin{aligned}
Y_1 &= I_1(f_1^1) + I_2(f_1^2) + \dots + I_p(f_1^p) + \dots + I_m(f_1^m) \\
Y_2 &= I_1(f_2^1) + I_2(f_2^2) + \dots + I_p(f_2^p) + \dots + I_m(f_2^m) \\
&\vdots \\
Y_i &= I_1(f_i^1) + I_2(f_i^2) + \dots + I_p(f_i^p) + \dots + I_m(f_i^m) \\
&\vdots \\
Y_n &= I_1(f_n^1) + I_2(f_n^2) + \dots + I_p(f_n^p) + \dots + I_m(f_n^m)
\end{aligned}
\tag{2}$$

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.

$$\begin{aligned}
\eta_1^p &= \{CDEF; AB\} \\
\eta_2^p &= \{C; ABDEF\} \\
\eta_3^p &= \{EF; ABCD\}
\end{aligned}
\tag{3}$$

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.

$$\begin{aligned}
f_1^p &= y_1' y_3' \\
f_2^p &= y_1 y_2 \\
f_3^p &= y_3
\end{aligned}
\tag{4}$$

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.

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

$$\begin{aligned} \text{If } \tau_1 &= S_i S_j; S_1 \\ \tau_2 &= S_i S_j; S_2 \\ &\vdots \\ \tau_k &= S_i S_j; S_k \end{aligned}$$

where $S_1 \cup S_2 \cup \dots \cup S_k = S$, then $S_i S_j; S$ can be covered by the product term $y_1^* y_2^* \dots y_k^*$, where $y_i^* = y_i$ or y_i' .

Proof: If states $S_i S_j$ 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 $S_i S_j; S$. The product expression that covers $S_i S_j; S$ is $y_1^* y_2^* \dots y_k^*$.

Lemma 2

$$\begin{aligned} \text{If } \{\eta_1; S\} &\text{ is covered by product term } P_1 \\ \{\eta_2; S\} &\text{ is covered by product term } P_2 \\ &\vdots \\ \{\eta_m; S\} &\text{ is covered by product term } P_m \end{aligned}$$

then the partition $\{\eta_1 \eta_2 \dots \eta_m; S\}$ can be covered by a sum of products expression $P_1 + P_2 + \dots + P_m$.

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_j 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 [2] 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**

$y_1y_2y_3y_4y_5$		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

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:

$$\begin{aligned}
 I_1 & y_1, y_2 \\
 I_2 & y_3, y_4, y_5 \\
 I_3 & y_4, y_5
 \end{aligned}$$

		I_1	
	Y_1	0	1
Y_2		GH 00000	CD 10010
	0	EF 01100	AB 11111
	1		

		I_3	
	Y_4	0	1
Y_5		EG 01100	BC 10010
	0	DF 01001	AH 00011
	1		

		I_2			
	Y_3Y_4	00	01	11	10
Y_5		G 01001	C 11110	B 11110	F 01001
	0	E 10101	H 00011	A 10110	D 10101
	1				

Figure 1: State Tables for Each Input

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.

$$\begin{aligned}
 Y_1 &= y_1 I_1 + (y_4 y'_5 + y_3 y_5 + y'_4 y_5) I_2 + y_4 y'_5 I_3 \\
 Y_2 &= y_2 I_1 + y'_5 I_2 + y'_4 I_3 \\
 Y_3 &= y_2 I_1 + (y_4 y'_5 + y_3 y_5 + y'_4 y'_5) I_2 + y'_4 y'_5 I_3 \\
 Y_4 &= y_1 I_1 + (y_4 y'_5 + y'_3 y_4) I_2 + y_4 I_3 \\
 Y_5 &= y_1 y_2 I_1 + (y'_4 + y_5) I_2 + y_5 I_3
 \end{aligned} \tag{5}$$

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 [5]. Shown in Fig. 4 is the flow table and the design equations follow. The state assignment satisfies Tracey's theorem.

$(y_4 y_5 y_1 \dots y_3)$	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

$$\begin{aligned}
 Y_1 &= y_5 y_2 y_3 I_3 + y_5 y_1 y_3 I'_3 + y_1 y_2 y_3 \\
 &\quad + y_5 y_1 y_2 + y_4 y_1 \\
 Y_2 &= y_4 y_3 I_2 + y_5 y_2 y_3 I'_1 + y_4 y_1 I_0 + y_5 y_1 y_3 I_0 + y_1 y_2 y_3 I'_2 \\
 &\quad + y_5 y_1 y_2 (I_1 + I_3) \\
 Y_3 &= y_4 y_5 I_0 + y_5 y_3 I'_1 + y_5 y_2 y_3 I_2 + y_5 y_1 y_3 I'_3 + y_1 y_2 y_3 I'_1 \\
 &\quad + y_5 y_1 y_2 I_2 + y_4 y_1 I'_0 \\
 Y_4 &= y_4 y_5 + y_4 y_3 (I_0 + I_1) + y_5 y_2 y_3 (I_0 + I_1) + y_4 y_2 + y_5 y_1 y_3 I_3 \\
 &\quad + y_5 y_1 y_2 I_0 + y_4 y_1 I_0 \\
 Y_5 &= y_4 y_5 I'_0 + y_4 y_3 I'_0 + y_5 y_3 I'_0 + y_4 y_2 I'_0 + y_5 y_1 y_3 I'_0 \\
 &\quad + y_1 y_2 y_3 I'_0 + y_5 y_1 y_2 I'_0 + y_4 y_1 I'_0
 \end{aligned} \tag{6}$$

The partition variables for each input are:

$$\begin{aligned}
 I_1 & y_1, y_2 \\
 I_2 & y_3, y_4, y_5 \\
 I_3 & y_4, y_5 \\
 I_4 & y_4, y_5
 \end{aligned}$$

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.

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 [6]. 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.

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 [6], 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 3 *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 4 *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.

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.

$$\begin{aligned}
 Y_1 &= I_1(y_1) + I_2(y_3) + I_3(y_5) \\
 Y_2 &= I_1(y_2) + I_2(y_4) + I_3(y_6) \\
 Y_3 &= I_1(y_1) + I_2(y_3) + I_3(y_6) \\
 Y_4 &= I_1(y_2) + I_2(y_4) + I_3(y_5) \\
 Y_5 &= I_1(y_2) + I_2(y_3) + I_3(y_5) \\
 Y_6 &= I_1(y_1) + I_2(y_4) + I_3(y_6)
 \end{aligned} \tag{7}$$

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 5: Flow table for Safe Design

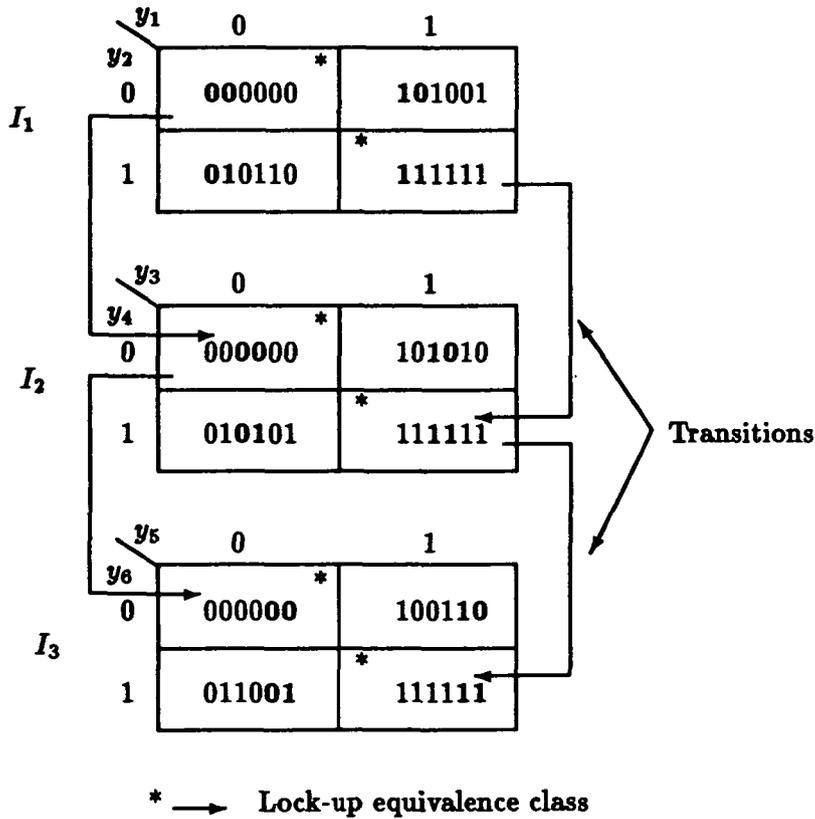


Figure 2: Safeness analysis

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 be 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.

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.

References

- [1] S. Unger, *Asynchronous Sequential Switching Circuits*, New York, NY, Wiley-Interscience, 1969.
- [2] J. Tracey, "Internal State Assignments for Asynchronous Sequential Machines", *IEEE Transactions on Electronic Computers*, Vol. EC-15, pp. 551-560, Aug. 1966.
- [3] C. Liu, "A State Variable Assignment Method for Asynchronous Sequential Switching Circuits," *J. ACM*, Vol 10, pp. 209-216, Apr. 1963.
- [4] C. Tan, "State Assignments for Asynchronous Sequential Machines", *IEEE Transactions on Computers*, Vol. C-20, No. 4, pp. 382-391, April 1971.
- [5] 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.
- [6] R. Wickersham and G. Maki, "Safe Asynchronous Sequential Circuits", *IEEE Transactions on Computers*, Vol. C-23, No. 5. May 1974.

9.1.12

Acknowledgement– The professional children and grandchildren of Dr. James H. Tracey would like to acknowledge his contribution to the theory of circuit design. Moreover, we would like to express our appreciation for his personal encouragement in our lives. We regard him as a dear friend and sincerely admire the dynamic enthusiasm expressed through his life.