FINAL REPORT

A DEMONSTRATION OF CMOS VLSI CIRCUIT PROTOTYPING IN SUPPORT OF THE SITE FACILITY USING THE 1.2 μm STANDARD CELL LIBRARY DEVELOPED BY NATIONAL SECURITY AGENCY

by

Edwyn D. Smith
Associate Professor of Electrical Engineering
The University of Toledo
2801 West Bancroft Street
Toledo, Ohio 43606

prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Lewis Research Center
Cleveland, Ohio

Space Electronics Division
Monty Andro, Technical Officer

Grant NAG 3-1036

March 1991

ACKNOWLEDGMENT

It is a pleasure to acknowledge the support provided by this grant to The University of Toledo. The chip designs created while pursuing the objective of the grant served well as foundations upon which four Master of Science in Electrical Engineering theses were constructed. From the University perspective, that is good `mileage´ indeed. As the project was concieved as a demonstration of standard cell CMOS ASIC implementation of low-cost logic, the fact that totally inexperienced students could produce working designs on first silicon would be convincing evidence. Let us hope it will prove to be so. The P.I. would like to thank the NASA Technical Officer, Mr. Monty Andro, for assistance, for numerous helpful suggestions, and much patience in answering questions from several struggling designers. Finally, I should like to speak a word of thanks to NASA on behalf of Professor Arthur R. Thorbjornsen whose untimely death in September, 1990, left the project and the University poorer.

INTRODUCTION

It was proposed to design two Si CMOS ASIC's, a Data Generator chip and a Data Checker chip. The effort was made more specific by some guidelines and constraints, summarized following. The logical design of these circuits was supplied by Lewis Research Center. It was agreed that the designs would be implemented to the extent possible using only cells from the scalable CMOSN Standard Cell Library developed by the National Security Agency [1]. It was further agreed that the designs be scaled to 1.2 μm technology with the understanding that a small number of prototype chips be fabricated through MOSIS [2]; the cost of fabrication to be borne separately by Lewis Research Center. Preliminary and full-functional testing was to be shared between The University of Toledo and Lewis Research Center as appropriate.

SUMMARY OF EFFORT AND PERSONNEL INVOLVED

The official period of performance was 14 April 1989 through 15 June 1990, including one no-cost extension. The informal period of performance began approximately November, 1988 and continued through the present (March, 1991). Permit me to explain what is meant by `informal period of performance.' Five international graduate students elected to contribute to this grant activity and use these same contributions as major portions of their M.S. theses. These students were not supported at any time by monies charged to this grant. I was given to understand by both the grant Technical Officer and the Office of University Affairs (NASA LeRC) that there was no prohibition or objection to these students being so involved. It follows that they constituted a volunteer labor force which accomplished most of what is reported herein.

Professor Arthur R. Thorbjornsen participated as co-P.I. in the grant activity from its inception to his death in September, 1990. All activity has remained continuously under the general supervision of Dr. E. D. Smith, P.I. The student volunteers were/are by name:

Mr. Anurag Gupta
Ms. Shobha R. Mallarapu
Mr. Anil S. Kapatkar
Mr. Sathyanarayana K. Rao
Mr. Kin Lui

SUMMARY OF ACCOMPLISHMENTS TO DATE

The CMOSN standard cells when read into the Univ. of Calif., Berkeley, layout editor MAGIC cause a minor but annoying design rule violation to be reported. Dr. Thorbjornsen modified many of the cells to remove the violation—the infraction occurs in the second layer of metal.

Dr. Thorbjornsen designed and submitted to MOSIS a simple test chip, intended to verify the library-MAGIC-MOSIS pipeline at 1.2 μm geometry. The prototypes were fully functional and indicated no difficulties. This work was accomplished during Spring quarter, 1988-89, and the first ten weeks of Summer, 1989. These prototypes were fabricated at University expense.

Mr. Anurag Gupta and Dr. E. D. Smith assumed responsibility for partitioning the total task into subunits and assigning the subunits to student volunteers. It quickly became apparent that the total project, Data Generator plus Data Checker could not be accommodated on a single chip. Thus, the first partition was to make the Data Generator and the Data Checker separate projects. The Data Generator was given first priority. Anurag Gupta agreed to be responsible for the logical design of the Data Generator as well as continuing to help coordinate the overall undertaking. Ms. Shobha Mallarapu agreed to be the principal layout designer for the Data Generator. As the layout approached completion, Anurag and Shobha cooperated to extract the interconnect wiring capacitances which Anurag used in hand-calculated estimates of the propagation delays—a first-order procedure to validate his initial timing analysis. At the time, we did not have a functional digital simulator capable of accepting the CMOSN library cells directly.

Well after the Data Generator layout was complete, we learned there was a major mistake in the logical design. The error concerned the all-zeros output data word—both as to when and how often it appears in the sequence of data words and the fact that the all-zeros data word has to have risetimes and falltimes essentially as fast as any other data word. The first Data Generator layout which I call Chip1 was put on a 40-pin MOSIS standard frame, the smallest available at 1.2 μm geometry. Chip1 very nearly filled the 40-pin frame. After a good bit of discussion, it was decided that a redesign would need to go to the next larger frame size, a 64-pin. The MOSIS price schedule revealed that a 64-pin chip at 1.2 μm geometry would cost about ten times as much as a 40-pin chip. Lewis technical officer requested we use two 40-pin chips rather than one 64-pin. The additional circuitry required was put on a second 40-pin layout, Chip2. A small

amount of rework was done to Chip1. The major change was the replacement of small pull-up transistors by large pull-down transistors to speed up the transition times for the all-zeros data words. The capacitance of sixteen large transistor gates connected in parallel was thought to be too big to drive comfortably with the output of a standard input buffer. A nonelegant but workable solution was to omit the input buffer and drive the line directly from the output pad driver on Chip2. As this connection between the two chips is presumed to be short, dedicated, and isolated, not going anywhere else, little ringing or other signal corruption is anticipated. The design and layout of Chip2, a relatively uncomplicated piece of work, was done mainly by Anil Kapatkar with some assistance from Anurag and Shobha. Technical details of the Data Generator, Chip1 and Chip2, are included as Appendix A; most of the material was excerpted from M.S. theses by Anurag and Shobha [3]-[4].

Anil Kapatkar assumed responsibility for the logical design of the Data Checker. It was immediately apparent that the I/O requirements could not be accommodated by a 40-pin frame even had the necessary circuitry been able to fit within the available area which it didn't. Again, by request of the Lewis technical officer, Anil began considering ways in which the Data Checker could be designed so as to allow the complete circuit to be divided between two 40-pin frames. As the design effort progressed, it became clear that it would be risky if even possible to attempt a scheme whereby all the necessary processing steps would be accomplished within a single period of a 13.2 MHz clock. By suggestion of the Lewis technical officer, the strategy for logical design was redirected towards a pipeline approach—a technique retaining synchronous data flow but permitting the processing of a single data packet to extend over more than one clock period.

Anil proposed a design which seemed to meet all pertinent criteria. Basically, it divided the 16-bit input data words into a 12-bit portion and a 4-bit portion. A block of circuitry consisting of input registers for the 12-bit portion, XOR gate comparators, a 12-bit population counter, and a small amount of control circuitry was designated to go on one 40-pin frame called (not too originally) Chip1. It was presumed that the remainder of the Data Checker could be put on a second 40-pin frame, Chip2; a rough check done by simply comparing the total area of the cells required to the core area available indicated it would be a tight fit. The Lewis technical officer having given permission to proceed, Anil performed a hand-calculated timing analysis which indicated the proposed design to be workable and, hopefully, robust.

Here, too, we lacked a simulator capable of doing critical timing analyses while working directly with CMOSN standard cells. Anil proceeded to do the layout of Data Checker, Chip1, a relatively simple piece of circuitry.

Mr. Sathy Rao agreed to undertake the layout of Data Checker, Chip2. As of the date of this report, this work is still ongoing. Our present estimate is that the circuitry dedicated to Chip2 will fit on a 40-pin frame, but only a completed or nearly completed design will confirm that. Whenever a finished layout becomes available, we shall be glad to pass it along to NASA LeRC. Technical details of the Data Checker, Chip1 and Chip2, to the extent they are available, are included as Appendix B. The bulk of this material is excerpted from the M.S. thesis by Anil Kapatkar [5].

Mr. Kin Lui is in process of bringing online for us some logic simulation capability: One tool will be a much improved version of the simulator RSIM capable of working with fully expanded CMOSN library cells. Another tool will be the proprietary Digital Logic Simulation module of PSpice using some subcircuit definitions created by Ms. Kalpana Vijayakumar [6]. The subcircuits implement logic macros with one-to-one correspondence to a subset of the CMOSN standard cell library. We are even now in the process of joining the Massachusetts Microelectronics Center; among other benefits, we hope to gain access to the HILO simulator, a product of the GenRad Corporation. As I now read it, we should be able to take a design based on CMOSN library cells, expand it in CIF, read it into Octtools, and then translate it into the HILO `language.'

The Data Generator and Data Checker chip sets are convenient examples of small to medium complexity CMOS ASIC's designed using the CMOSN standard cell library. Hopefully, in due time, Kin should be able to examine one to all of the chips using each of the simulators mentioned above. This should provide us not only with a comparison of the merits of the several simulators but should also give a firm decision whether the chips are functionally correct. As any new information becomes available, we shall be glad to pass it along.

CITATIONS

[1] CMOSN Cell Notebook: Scalable 2.0 and 1.2 Micron CMOS/Bulk Cell Family, developed by the National Security Agency, distributed by MOSIS, USC Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90292-6695.

[2] The MOSIS Service, USC/Information Sciences Institute 4676 Admiralty Way, Marina Del Rey, CA 90292-6695

[3] Anurag Gupta, "A Design Exercise: Transforming a Block of ExistingCircuitry into a CMOS ASIC using Standard Cell Methodology," M.S. thesis, The University of Toledo, Toledo, Ohio, June, 1989.

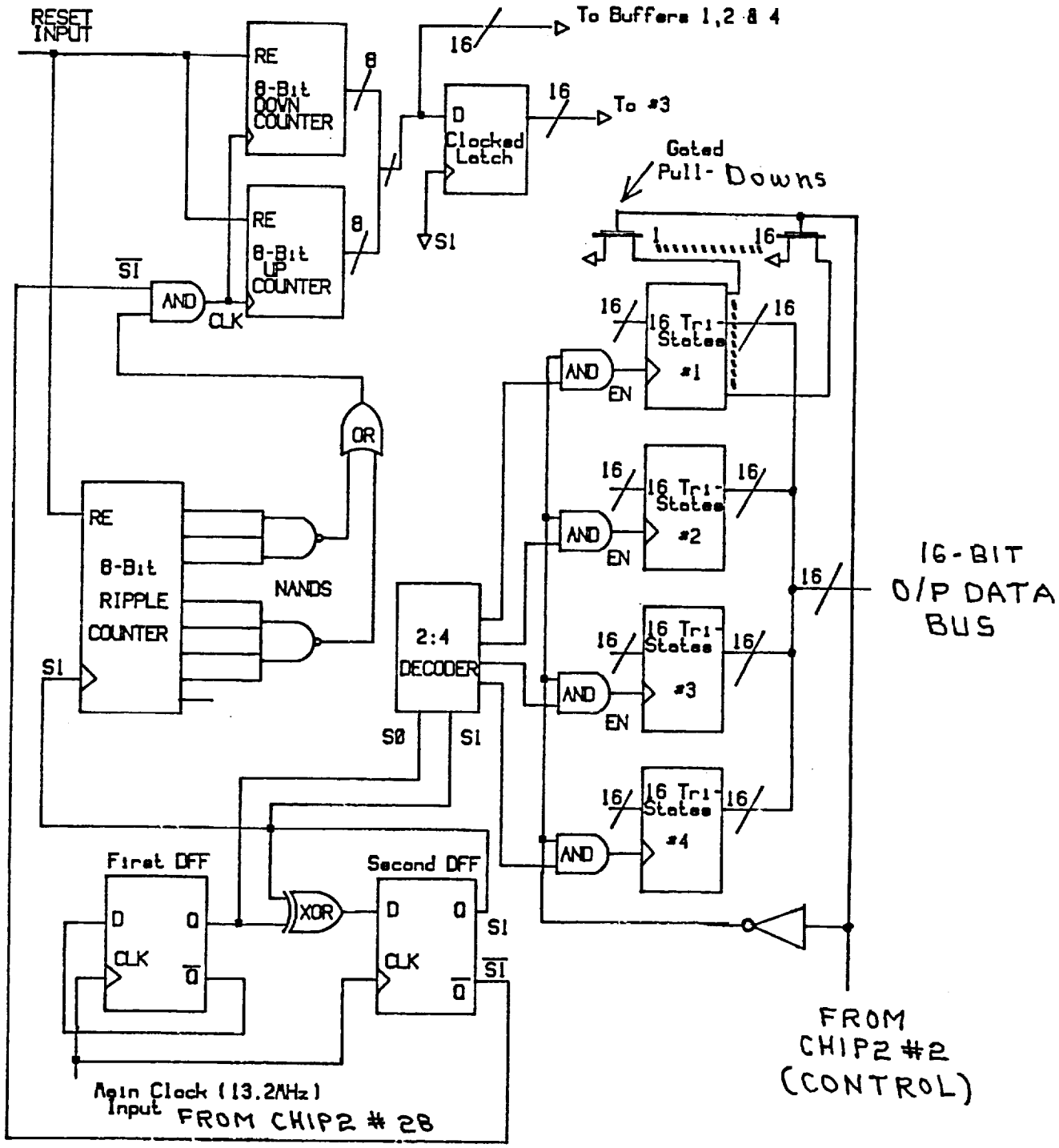
[4] S.R. Mallarapu, "A Demonstration of CMOS VLSI Circuit Prototyping in Support of the SITE Facility using the 1.2 μm Standard Cell Library," M.S. thesis, The University of Toledo, Toledo, Ohio, June, 1989.

[5] Anil Kapatkar, "A CMOS ASIC Implementation of an Existing Block of Circuitry using the Standard Cell Approach," M.S. thesis, The University of Toledo, Toledo, Ohio, in progress.

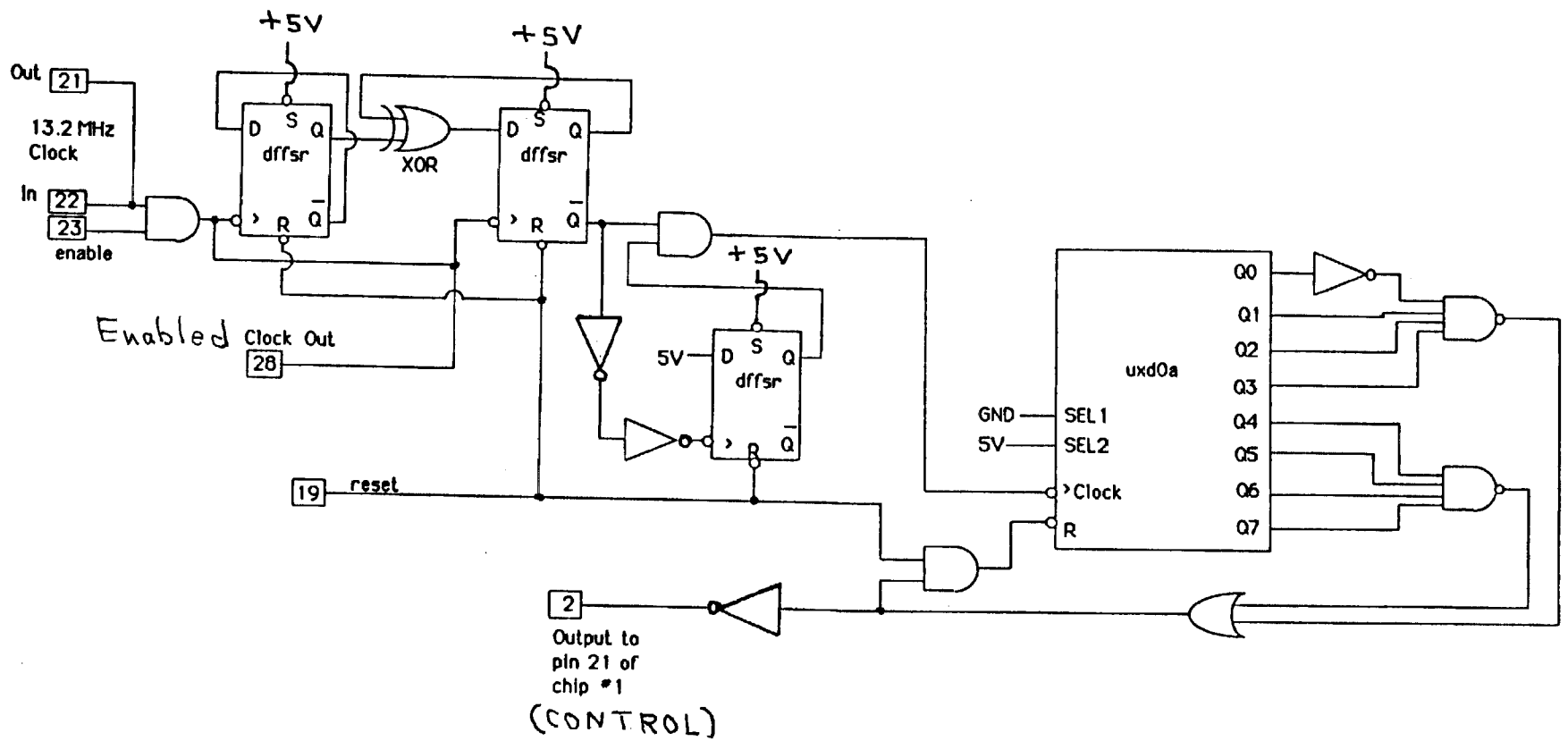
[6] Kalpana Vijayakumar, "Extension of the Proprietary PSpice Digital Simulation Library to Include the Scalable CMOS Standard Cell Library Developed by the National Security Agency," M.S. thesis, The University of Toledo, Toledo, Ohio, in progress.

APPENDIX A

Technical Details of the Data Generator, Chip1 Plus Chip2.



Chip1 Functional block diagram for Data Generator



NASA DATA GENERATOR CHIP NUMBER 2

BLOCK DIAGRAM MANUALLY EXTRACTED

FROM THE CHIP LAYOUT BY ART THORBJORNSEN

FEBRUARY 12, 1990

The output of the Data Generator originates from an 8-bit up counter and an 8-bit down counter; each pair of counter settings yields four 16-bit data words. The convention is here followed: 0 refers to the LSD and 7 refers to the MSD. Pin assignments and bit mappings are summarized in the table below.

TABLE 1

Output Data Mapping and Pin Assignments

Package Pin Number	O/P Bus Line Number	Up/Down Counter Connections			
		Word 1	Word 2	Word 3	Word 4
33	1	7D	1U	2U	0D
31	2	3D	7D	5U	5D
30	3	7U	4D	2D	3D
29	4	2U	7U	6U	2D
27	5	6D	6D	4U	6U
18	6	4D	1D	3U	5D
17	7	6U	5U	0D	1D
13	8	3U	0D	0D	4D
11	9	2D	6U	3D	4U
10	10	1U	2D	1U	1U
9	11	1D	5D	4D	7U
7	12	3U	0U	0U	2D
2	13	6U	1D	2U	5U
40	14	4U	4U	7D	0D
39	15	5D	6D	7U	3D
37	16	3D	0U	5U	6D

Data Generator, Chip1 Plus Chip2, Package Pin Assignments and System—Chip1—Chip2 Interconnect Wiring

The pin assignments of Data Generator—Chip1 are shown in the drawing on page A23.

The pin assignments of Data Generator—Chip2 are given below.

I/O Power.....Pins 10, 17
I/O Ground.....Pins 1, 30
Core Circuitry Power.....Pins 5, 35
Core Circuitry Ground.....Pins 20, 27
System Clock In.....Pin 22
System Enable In.....Pin 23
System Reset In.....Pin 19
Control Out.....Pin 2
Enabled Clock Out.....Pin 28
Pass-Through Clock Out.....Pin 21
All other pins are unused.

System Reset is supplied to both Chip1 and Chip2. System Clock and System Enable are supplied only to Chip2. Enabled Clock Out and Control Out are passed from Chip2 to Chip1. The wire list is tabulated below.

System	Chip1	Chip2	(Function)
Clock		Pin 22	
Enable		Pin 23	
Reset	Pin 20	Pin 19	
	Pin 21	Pin 2	Control
	Pin 23	Pin 28	Enabled Clock

The pin assignments for the output data bus are given in TABLE 1, page A3, and in the drawing on page A23.

Chapter III

The Data Generator

System Specification

This thesis documents the conversion of the Data Generator circuitry, made available to us in TTL technology, into a pair of CMOS ASIC chips using the 1.2 um standard cell library made available by the National Security Agency (NSA). The redesigned functional block diagram of Data Generator Chip 2 is shown on page A1.

Given a clock of 13.2 MHz frequency, we are to generate a 16-bit word by using an 8-bit UP and an 8-bit DOWN counter, at one-fourth the given frequency (3.3 MHz), then do a 16-bit to 64-bit mapping and finally generate four, 16-bit words sequentially that are delivered to the 16-bit output bus. Thus, we are generating four words (16-bit wide) from a single word (16-bit wide) by accomplishing the 16-bit to 64-bit mapping that was already specified to us. Table 1 shows the bit by bit mapping of both the 8-bit UP and DOWN counters. We are also to skip a 3.3 MHz clock pulse (254^{th} or 255^{th}) thereby generating the same word twice for that pulse. The output is pulled to an all-zeros state under control of circuitry on Chip 2. Thus, the counters would not increment or decrement for that particular clock pulse. So, with all of this in mind, let us divide the logic diagram of the Data Generator into four major sections:

1) Logic for generating an internal clock of 3.3 MHz from the given 13.2 MHz clock.
2) Logic for skipping the $254^{th}/255^{th}$ clock pulse, i.e., generate the same word twice for that pulse.
3) Logic to generate four words sequentially using sixty-four tristate buffers in four blocks of sixteen each.
4) Control circuitry to disable the tristate buffers, and pull the outputs low.

Generating the 3.3 MHz clock

Two edge-triggered D Flip-Flops (DFFs) are used as frequency dividers, each dividing the frequency by two, hence generating a 3.3 MHz clock from the given 13.2 MHz clock. The DFF has a single data input D, and it operates such that the logic level present at the D input is transferred to the Q output only on the positive or the negative going edge of the input clock signal. In our case, the D input is transferred on the falling edge of the clock signal. The D input has no effect on Q at any other time.

The given clock of 13.2 MHz is applied to the clock (CLK) input of the first DFF (refer to page A1), with Q bar fed back to the D input. Thus, the output Q generates a clock whose frequency is one-half the frequency of the given 13.2 MHz clock.

Now, the Q output of the first DFF (i.e. 6.6 MHz clock) is applied to one of the inputs of an Exclusive OR (X OR), the other input of which is connected to the Q output of the second DFF. The 6.6 MHz clock also serves as the S_0 bit for the 2:4 line decoder, which will be discussed later in the chapter. The given clock of 13.2 MHz is also applied to the CLK input of the second DFF and the output of the X OR is fed to the D input of the same DFF. Thus, with this configuration, we get a 3.3 MHz clock at the Q output of the second DFF. In short, the two flip-flops and the exclusive OR are simply interconnected to form a 2-bit synchronous up counter -- this is a standard design for synchronous up-down counters. To see the actual waveforms, please refer to the timing diagrams illustrated in chapter four. So, at this point we have the 3.3 MHz clocks S_1 and S_1 bar. The same clock also serves as the S_1 bit for the 2:4 line decoder, discussed later in the chapter.

Skipping the 254th/255th clock pulse

The control circuitry to inhibit the $254^{th}/255^{th}$ clock pulse requires an 8-bit ripple counter. The operational mode of the counter is determined by the SELECT input and as it is used as an UP counter in our circuitry, the SELECT input is connected to a logic low (GND). The DATA input of the ripple counter is connected to the 3.3 MHz clock S_1 generated at the Q output of the second DFF. All outputs for the counter occur on the negative edge of the clock.

The outputs (Q_7-Q_1) are connected to the four input and three input NAND gates. This is done because on the 254^{th} and 255^{th} pulses, all outputs (Q_7-Q_1) are high and as we want the outputs of the two NAND gates to be low for these two pulses, all inputs to the NAND gates must be high. The two outputs from the NAND gates are ORed and the result, which is the output of the OR gate, is connected to one of the inputs of the 2-input AND gate.

Due to the propagation delay across the 8-bit ripple counter, the zero bit, which is supposed to disable the AND gate and hence stop the counters from incrementing/decrementing, is delayed by approx - imately 55 nsec. This amount of time is sufficient for the AND gate to allow the leading edge of the 254^{th} clock pulse to pass but which will end 55 nsec later when the zero bit disables the AND gate. The falling edge of the pulse in turn increments/decrements the counters and this constitutes the 254^{th} count even though we "wanted" to skip it. But, having a logic low at the output of the OR gate for two successive pulses keeps the AND gate disabled for both the pulses and so even though the 254^{th} pulse is counted by the counters, the next pulse i.e., the 255^{th} is skipped because of the logic low at one input of the 2-input AND gate. This allows us to inhibit the 255^{th} clock pulse from incrementing/ decrementing the UP/DOWN counters respectively. The logic low at the output of the OR gate disables the AND gate, the output of which is applied to the clock inputs of the UP and DOWN counters, thereby preventing the counters from incrementing/decrementing for that clock pulse.

Thus, the previous word is repeated, but the outputs are pulled low by the control circuitry on Chip 2. The timing diagrams in chapter four illustrate the waveforms of all these clock pulses in question. The RESET of the 8-bit ripple counter is given a logic high during normal operation.

Generating four words (16-bit wide) sequentially

Looking at the functional block diagram of the Data Generator (page A1), we see that the 16-bit word generated by the UP/DOWN counters is mapped bit-by-bit to the sixty four tristate buffers. The sixty-four tristate buffers are divided into four groups of sixteen each. Thus, it is convenient to speak of our tristate buffers, each sixteen bits wide. The ENABLE input of each buffer is connected to the output of one of four 2-input AND gates. One input of each of the four AND gates is connected to one of the four outputs of the 2:4 line decoder. The inputs to the 2:4 line decoder are the S_0 and S_1 signals, generated at the outputs of the first and second DFFs respectively. The 2:4 line decoder enables the four tristate buffers one at a time in a sequence which repeats indefinitely. The four tristate buffers being enabled one-by-one (sequentially), generate four data words sequentially at the output for each state of the UP/DOWN counters. The timing diagrams in chapter four show how the four tristate buffers are enabled sequentially. The DATA inputs of the tristate buffers are derived by the 16-bit to 64-bit mapping from the outputs generated by the 8-bit UP counter and the 8-bit DOWN counter.

Control Circuitry

Sixteen gated Pull-downs are connected in parallel with the outputs from the tristate buffers; they pull the data outputs low when activated by the control signal from Chip 2. These devices are simple N-channel MOS transistors with their sources tied to GND, their drains connected to the output bus in parallel with the outputs from the tristate buffers; the gates are connected together and controlled by the control signal from Chip 2. Please refer, pages A1 and A2. Hence, to disable the tristates, the control is given a logic high which forces a low output from the AND gates; this in turn, imposes a high impedance state on the tristate buffers. Additionally, the logic high at the gates of the pull-downs turns the transistors on, thus pulling all the outputs low. A logic low at the control input is used for the normal operation of the Data Generator.

This completes the description of the logic operation of the Data Generator circuitry. The propagation delays and the timing diagrams for the circuitry are discussed in the next chapter.

Timing Diagrams

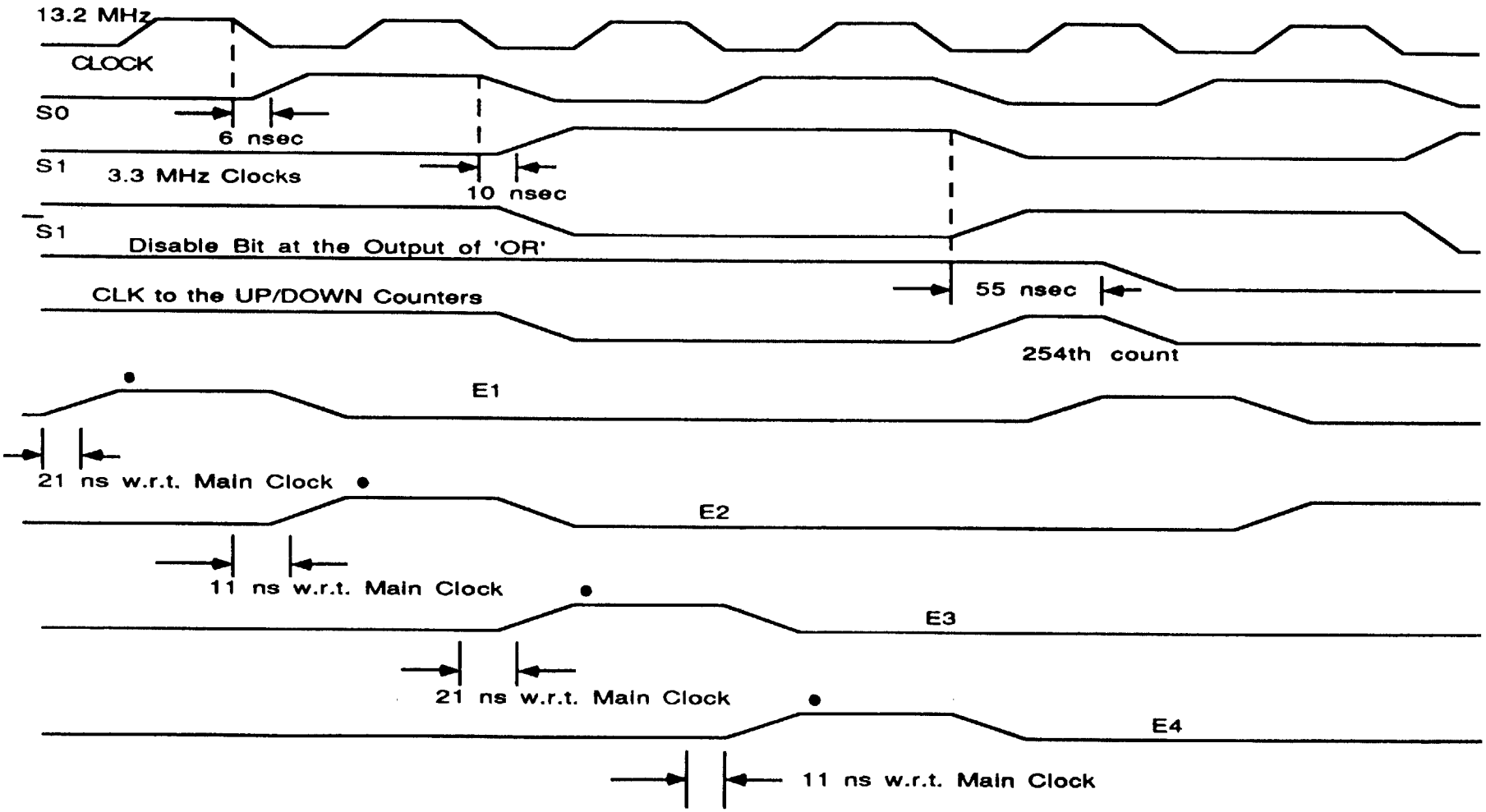
Figure 16 illustrates the timing diagrams for the Data Generator, Chip 1, circuit with approximate rise and fall times and the propagation delay of one clock with respect to another. It was only after looking at these diagrams, that we realized the need for a clocked latched at the input of the third buffer (refer to page A1).

Actually, what was happening was that the UP counter and the DOWN counter were being incremented/decremented precisely at the time the third (#3) tristate buffer was being enabled.

It was thus questionable whether the 16-bit data word output from the number three buffer (actually the last because the readout order is 4-1-2-3) would remain valid long enough to satisfy system requirements.

Timing information of standard cells
used in the Data Generator circuit.

CELL NAME	EQUATIONS USED (for Worst Case at 125 C)		Pd (0-1) ns	Pd (1-0) ns	Tris ns	Tfal ns
2-Input NAND	Pd(0-1) = 1.59 + (2.60)CL Pd(1-0) = 1.11 + (1.98)CL	Tris = 2.82 + (6.71)CL Tfal = 1.79 + (4.44)CL	2.45	1.76	4.86	3.26
2 - Input AND	Pd(0-1) = 0.72 + (2.53)CL Pd(1-0) = 0.75 + (2.16)CL	Tris = 1.18 + (5.10)CL Tfal = 1.11 + (4.14)CL	3.52	2.82	6.75	4.94
8_bit UP/ DOWN COUNTER	Pd(0-1) = 11.08 + (1.88)CL Pd(1-0) = 11.60 + (1.50)CL	Tris = 4.50 + (3.90)CL Tfal = 3.68 + (2.39)CL	11.42	11.87	5.20	4.11
2:4 Line DECODER	Pd(0-1) = 3.79 + (1.83)CL Pd(1-0) = 5.36 + (1.53)CL	Tris = 1.61 + (4.01)CL Tfal = 2.14 + (2.37)CL	4.40	5.87	2.30	2.30
D - FLIP FLOP	Pd(0-1) = 4.82 + (1.61)CL Pd(1-0) = 5.36 + (1.34)CL	Tris = 2.36 + (3.93)CL Tfal = 2.32 + (2.34)CL	6.00	6.30	5.10	4.00
Tri-State Buffer	Pd(0-1) = 2.16 + (2.13)CL Pd(1-0) = 2.60 + (1.91)CL	Tris = 1.57 + (4.49)CL Tfal = 1.38 + (3.54)CL	3.35	3.67	4.08	3.36
8_bit RIPPLE COUNTER	Pd(0-1) = 5.46 + (1.82)CL Pd(1-0) = 6.56 + (1.53)CL	Tris = 2.78 + (3.75)CL Tfal = 2.68 + (2.34)CL	6.67	7.68	4.06	3.47
2-Input NOR	Pd(0-1) = 1.58 + (1.62)CL Pd(1-0) = 1.60 + (1.24)CL	Tris = 3.09 + (3.82)CL Tfal = 2.46 + (2.45)CL	2.30	2.20	5.0	3.60
2-Input OR	Pd(0-1) = 0.77 + (1.33)CL Pd(1-0) = 0.76 + (1.24)CL	Tris = 1.29 + (2.90)CL Tfal = 1.09 + (2.55)CL	4.00	3.50	8.0	5.6



• Enable Bits for Tri-State Buffers. (E1, E2, E3 & E4)

Fig.16 Timing diagram for Data Generator.

In fact, in a worst case scenario, whether the word would even be valid at all. Introducing a clocked latch ensured a valid word at the output of the third tristate buffer. As the latch stored the previous word, the fact that the counters and the buffer were enabled at the same time had no effect on the valid word (word stored by the latch) being generated at the output. Thus, there is no doubt about the four words being generated sequentially at the output. It is also true now that every output word has exactly the same timing as every other one.

Another helpful application of the timing diagrams was made use of in connection with the 254^{th} pulse that we wanted the counters to skip. In fact, due to the propogation delay across the 8-bit ripple counter, the zero bit, which is supposed to disable the AND gate and hence stop the counters from incrementing/decrementing, is delayed by approximately 55 nsec. This amount of time is sufficient for the AND gate to allow the leading edge of the 254^{th} pulse to pass; the pulse is terminated 55 nsec later when the zero bit disables the AND gate. The falling edge of the pulse in turn increments/decrements the counters and this constitutes the 254^{th} count even though we originally intended to skip it.

This lead us to design the skip circuitry such that the zero bit disables the AND gate for two successive clock pulses (254^{th} and 255^{th}); it then happens that even though the 254^{th} pulse is counted we do end up skipping the 255^{th} pulse, which is what we wanted to do.

Thus, we see that the timing information and the timing diagrams play a very vital role in the logical design of the circuit. A brief description of the physical layout and design of the Data Generator chips is provided following.

CHAPTER IV

PHYSICAL DESIGN OF THE DATA GENERATOR

Layout design is the process of translating a schematic diagram representation into a corresponding representation consisting of geometric features on several independent mask layers. All present day (MOS) IC chips contain some unavoidable overhead: bonding pads, output pad drivers, input and output electrostatic discharge protection circuitry, (occasionally) input receivers, and level shifters. Dictated by packaging requirements, there is a semi-standard set of chip sizes and rectangular shapes. A chip conforming to one of these semi-standard sizes and shapes and containing the necessary associated overhead features is called a standard frame.

In the standard cell approach, the layout design of the circuitry within each cell is pre-accomplished; layout drawing consists of placement of the cells and detailed design of the interconnect wiring, the so called "place and route" problem. The first task in creating the layout is to create a floorplan for the chip. The chip is broken into blocks, and by looking at their relative sizes and the wiring between them, decisions are made where the blocks should be placed. To draw the plan, we need an estimate of the size and shape of the cells that will be used to make up the blocks plus the amount of space that will be left between them for wiring channels. When the floorplan is complete, the actual to-scale place and route is undertaken.

Design Methodology

Fig.7 shows a possible methodology for VLSI circuits [7]. It is a combination of top-down and bottom-up approaches. It is top-down because it considers

the overall functions to be performed and how these functions interact with each other and with the system. At the same time, low-level details (concerning capabilities, etc.) are considered to enable the designer to make informed, intelligent trade-offs. This thesis deals with the physical design starting from chip block diagram level. The part prior to it can be found in the M.S. thesis done by Mr. Anurag Gupta[9].

Floorplanning

It was decided to put five rows of cells on the chip. Part of the analysis that led to the choice depends on how the data and control signals flow and how it is possible to place the standard cell blocks and still remain within the constraints of actual space available. These considerations dominate the process of deciding where and how far apart to locate the various buliding blocks -- a process better known as floorplanning.

As the floorplan and the functionality of the individual blocks become better defined, the actual cells to be used to implement the blocks must be determined. These must be chosen from the cells available in the CMOSN standard cell library. The floorplan of the Data Generator is shown in Fig.8.

The size of the chip is 3917x3917 lambda units. The estimated area available for the placement of cells after placing the pad drivers, pad protectors, power and ground pads is 2550x2550 square lambda units. As a standard cell approach is used in this design, the height of the cells is standard at 250 lambda units. Hence for five rows of cells, 1250 lambda units has already been used. The rest of the space can be used for wiring.

The placement of cells has to be decided depending on the interconnections to be made between the cells. As each large buffer consists of 16 one-bit tristate buffers, all the sixteen tristate buffers have to be placed next to each other. In addition to this consideration, the outputs of the four large buffers must be connected to the same bus.

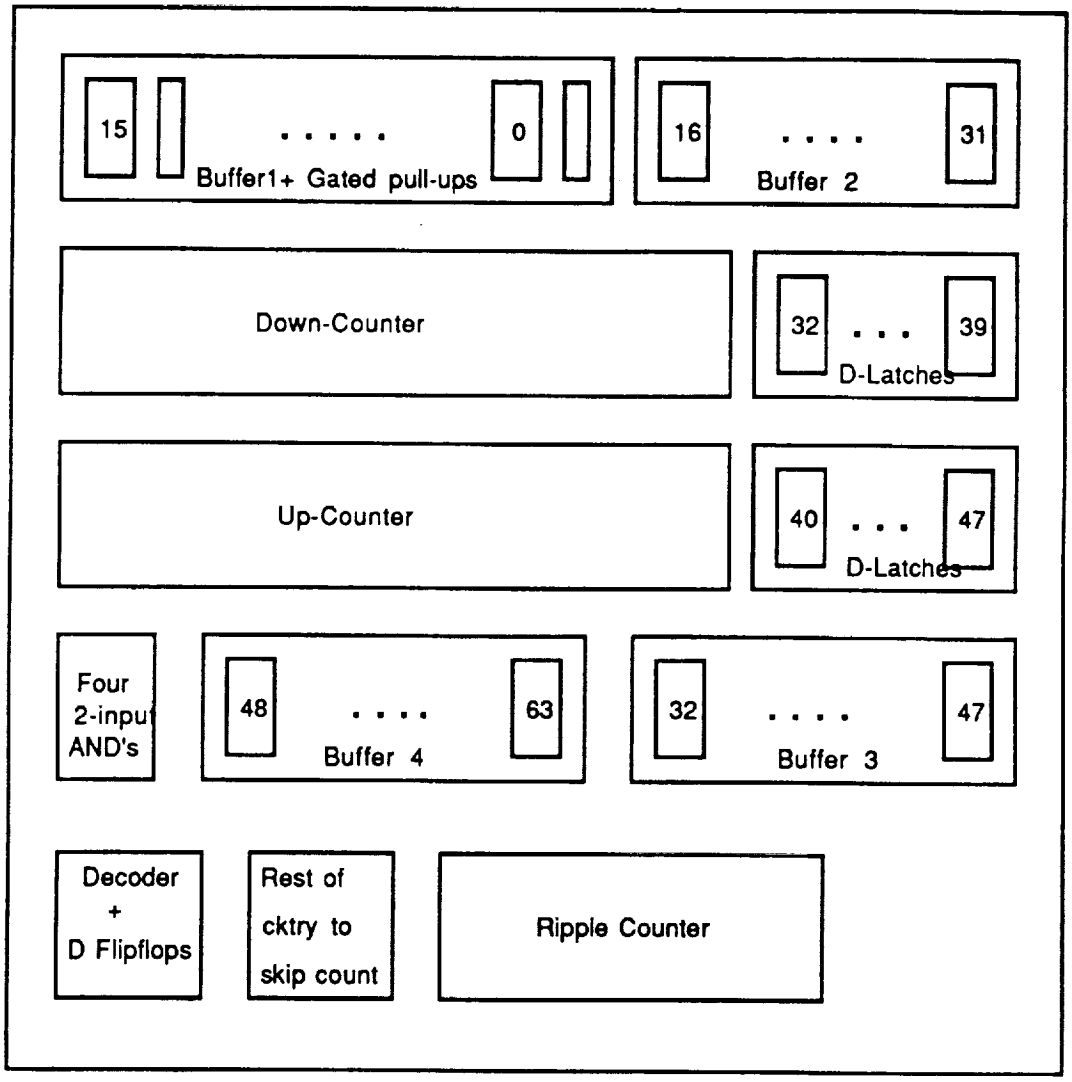


Fig.8 Floorplan for the Data Generator

Corresponding bits of each large buffer plus an active pull-down are connected together and to a rail of the bus. Keeping this in mind, buffers 1 and 2 are placed side by side and similarly buffers 3 and 4. The gated pull-downs are placed with buffer 1. Because the input to the third buffer has to be passed through data latches from the up-down counters, the data latches are placed directly above buffer 3 and next to both counters. The four AND gates used to enable the four large buffers are placed next to buffer 4 because space is available there and that location brings them close to other buffers. The rest of the Data Generator circuitry is placed together in a row because there are many more connections among that group of cells than to the cells placed in other rows.

The cells are also arranged such that all the inputs are to the left and the outputs to the right. But as there are sixteen outputs, they had to be distributed around the chip. There are only three inputs -- Control form Chip 2, Reset and the Clock. The cell rows are centered leaving equal space on either side of each row of cells for VDD and VSS buses. A space of 250 lambda was left between the rows of cells for wiring. At the bottom of the last row, only 80 lambda units was left as there are only a few connections to be made through that channel.

Although the wiring was considered in the floorplan, as the actual layout process progressed, the 16-bit to 64-bit interconnect wiring layout was changed to accommodate the required positions of individual wires as the design evolved and the necessary positions became clear. The wiring section of the layout is discussed in detail later in this chapter.

The Workstation

The process of developing a VLSI chip demands a great deal of interaction between the designer and the evolving design. Workstations support computer-aided engineering tools that organize and expedite the interaction. From schematic creation to simulation and physical layout, workstation tools help manage all interrelated design tasks associated with VLSI development and

allow easy updates to schematics and layouts. The workstation used to create this design was a Sun 3/160 running a 4.0 operating system.

One of the CAD tools available on the Sun workstations is called MAGIC. It is an interactive layout editor and design rule checker. It does not have the capability of placing the cells automatically but has the capability of routing automatically using netlists. In order to use netlist routing, one has to specify the nodes that have to be connected to each other. Then it automatically connects the nodes taking care of any crossovers hat occur in the midst of its routing. However, netlist routing was not used in this design in order to optimize the area of the chip. When the netlist router is used, it is hard to predict how it will route - where it will place wires and how long wires it will use. On the other hand, routing manually allows us to visualize the routing and run the wires as desired. In order to avoid crossovers and contacts, all horizontal wires are run in metal1 and all vertical wires in metal2. The technology used is 1.2um scalable CMOS and the design rules are lambda based as has already been discussed in Chapter 1.

40 pin Standard Frame

A 40-pin tiny chip can accommodate the Data Generator. The frame of the chip has been redesigned to meet the requirements of the Data Generator -- the number of inputs and outputs, etc. Each input pin of the chip requires a pad protector and each output pin requires a pad driver. As we need only three inputs and sixteen outputs, the remaining 21 pins could be used for power and ground. It is a good practice to provide separate power and ground pads and buses for the I/O circuitry and the core circuitry. The big benefit of dong this is that it prevents power supply and ground noise produced by the pad drivers from being distributed directly to the core circuit. A good bit of bypassing can be done at the package pins. The power and ground pads for the pad drivers and pad protectors are called power and ground. The VDD and VSS pads for the circuit are called power circuit and ground circuit. This approach involves isolating the pad power and ground circuits.

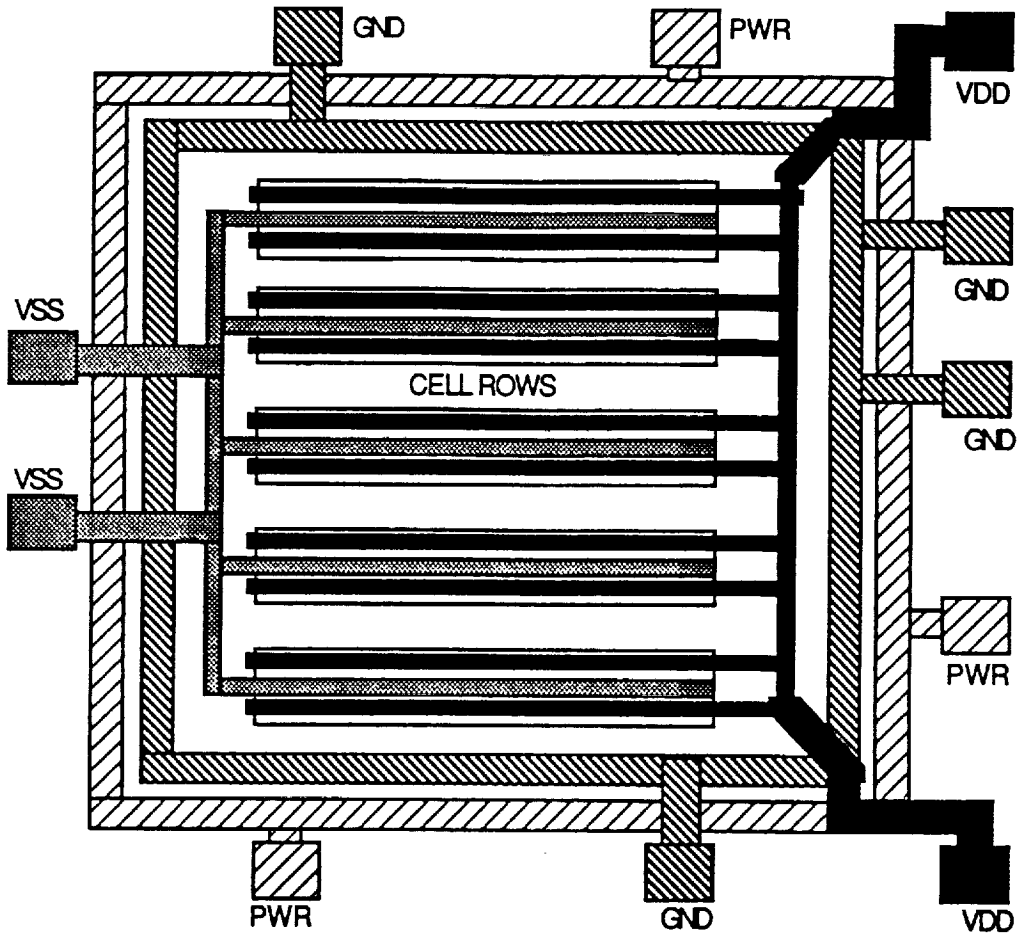


Fig.9 Chip Power Distribution for the Data Generator, Chip 1

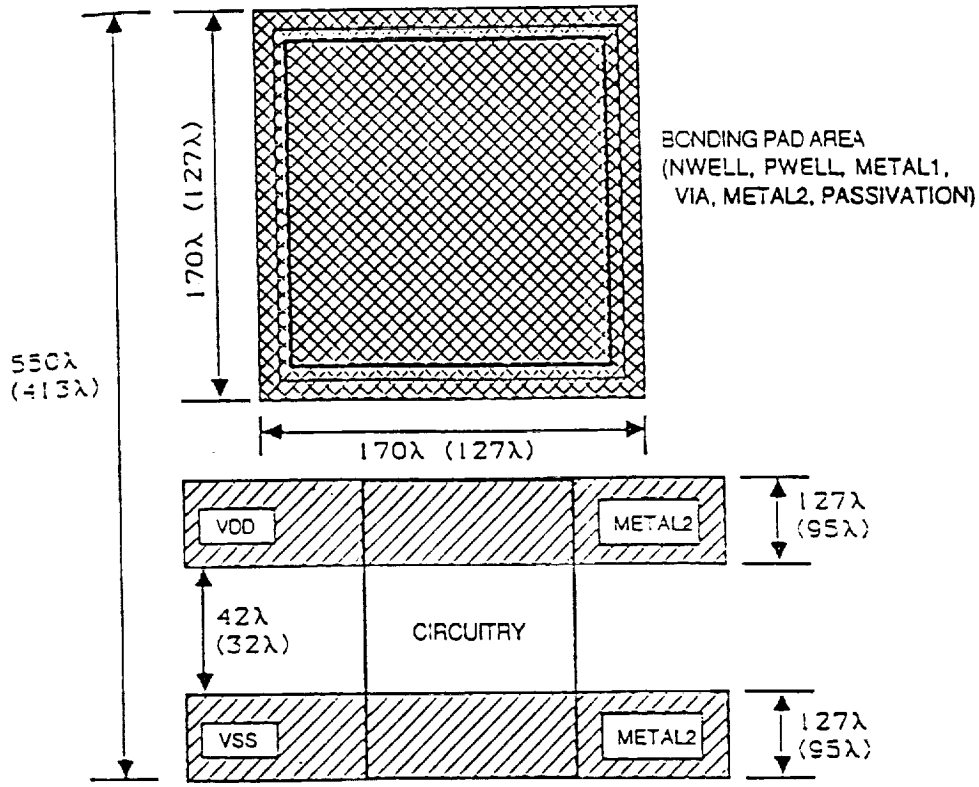


Fig.10 I/O Pad Layout Structure

As can be seen in the Fig.10, the bonding area in the pad is 170x170 square lambda units where as the area of each pin in the frame is 175x175 square lambda units. This disables the bonding area of the pad to be symmetric on the pin when placing the pad on the pin. Therefore, to be uniform throughout the chip, all pads are overlapped on the pins with 2 units on the top, 3 units at the bottom, 3 units on the left and 2 units on the right. To place the pads on the pins, the following procedure is adopted:

1). The MAGIC file of the 40-pin chip is opened.

magic 40pc22x22

2). One of the pads - input, output, power or ground pad is called on to the screen.

:getcell x2ipd

will bring the input pad onto the screen.

3). The pin where the pad has to be placed is chosen and its glass layer is selected by pressing 's' two or three times. The command **:what** will give the name of the layer that has been selected. Once the glass layer is selected, the lower left coordinates of the box around it is obtained by pressing 'b' which is a macro for box. It gives the lower left coordinates as **ll=(x_1,y_1)**.

Similarly the glass layer of the bonding area of the pad is selected and its lower left coordinates are obtained **(x_2,y_2)**.

The pad now has to be moved **(x_1-x_2-3, y_1-y_2-3)** units in both directions so as to be placed on the pin with an overlap of 3 units. After selecting the whole pad by pressing 'f', the following commands would place the pad exactly on the pin as desired:

:move east (x_1-x_2-3) or **move west (x_1-x_2-2)**

:move north (y_1-y_2-3) or **move south (y_1-y_2-2)**

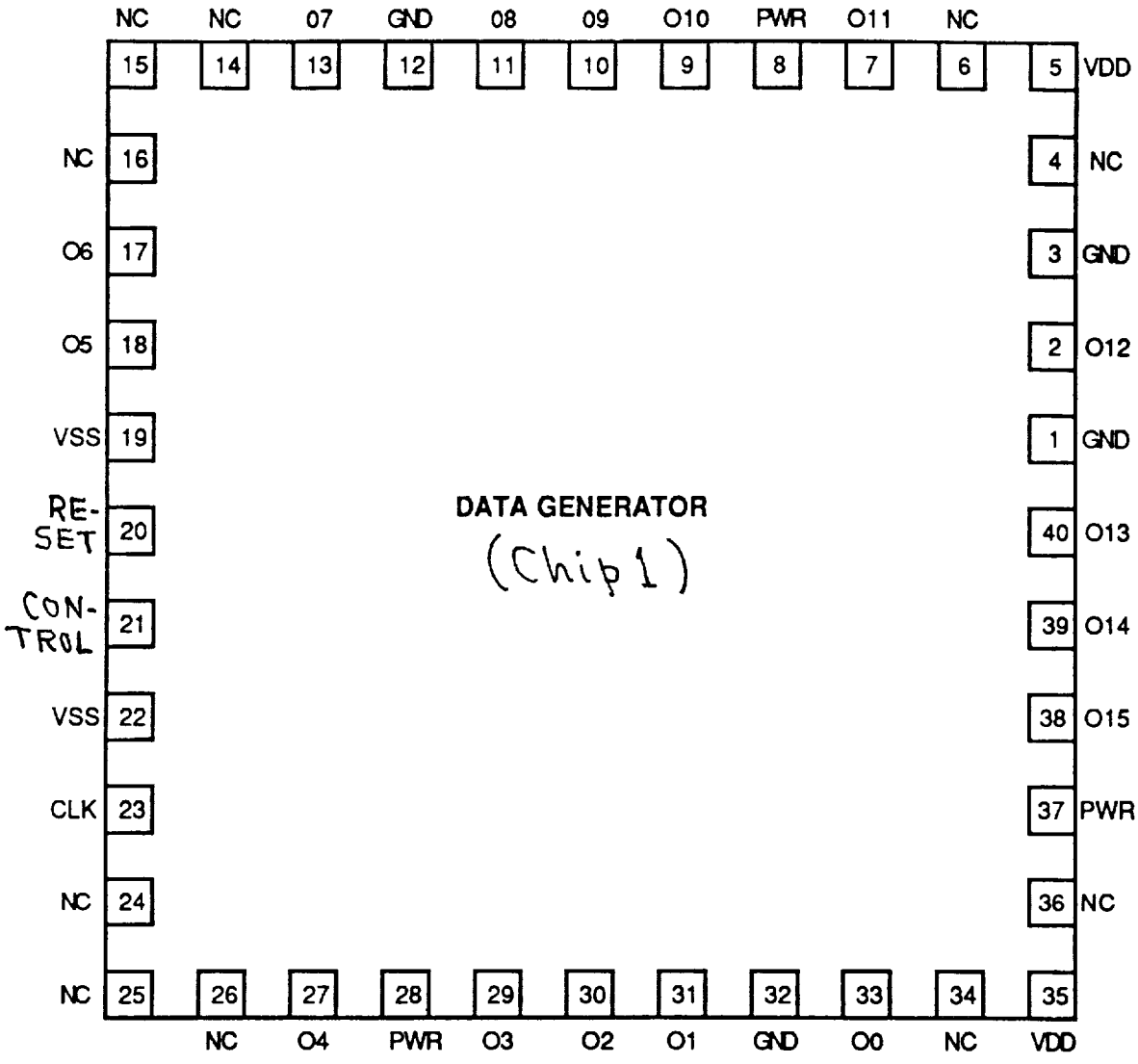
Similarly all the pads are placed on the frame.

To avoid overlapping of the pads, the pins next to the corner pins are left unconnected. The pin diagram of the Data Generator is shown in Fig.11.

Input protection and banding

To protect against damage caused by electrostatic discharge and to reduce the possibility of latchup, special structures have been included in the I/O pads. The input protection circuit [3] is shown in Fig.12. The input resistor is composed of ten squares of polysilicon having a resistance of approximately 500Ω. After the input resistor, the signal line is connected in metal to a P+/N-diode and an N+/P- diode. The diode areas are large to enhance their capability to handle current. The P+/N- diode is surrounded by an N+ ring tied to VDD and the N+/P- diode is surrounded by a P+ ring tied to VSS.

To reduce the possibility of latch-up, a P+ active area guard ring surrounds the N-channel transistor and an N+ active area guard ring surrounds the P-channel transistors in the I/O pads.



O - OUTPUT PIN
I - INPUT PIN
NC - NOT CONNECTED
GND - PAD GROUND
PWR - PAD POWER
VDD - CIRCUIT POWER
VSS - CIRCUIT GROUND

Fig.11 Pin diagram of Data Generator, Chip1

buses from the cell power buses in order to reduce power and ground noise in the functional cells. The more the number of power supply pads, the better is the power and ground noise in the pad area. Also, because this noise results from the switching of output buffers, large strips of adjacent output pads are avoided as much as possible. The chip power distribution for the Data Generator is shown in Fig.9.

The basic I/O pad layout structure is shown Fig.10.

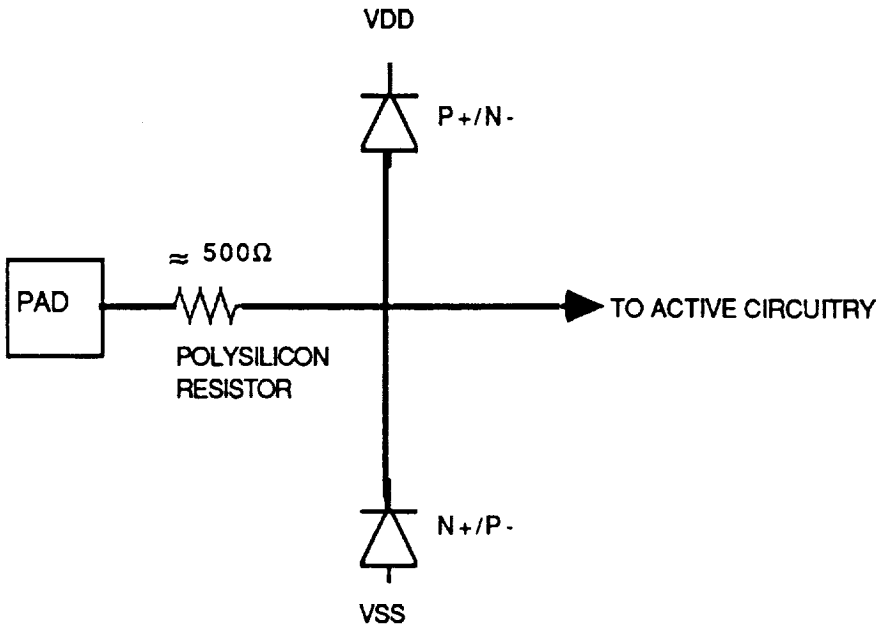


Fig.12 Input Protection Circuitry

Placement and Routing

The placement of the cells has been done in a bottom-up fashion. This is true mainly because the last row of cells had very few connections at the bottom and the connections at the top were mostly interconnections among cells in that row. So, leaving a space of 80 lambda units from the bottom I/O pads and 250 lambda on the left, the decoder is placed using the command

:getcell dec3 .

Similarly, the other cells in that row are placed with a spacing of 10 lambda between cells. Before placing the next row of cells, the interconnections among these cells are completed. In addition to the wiring area already occupied, a spacing of 100 lambda is left for the connections to the next row of cells. The number 200 lambda is obtained by estimating that 16 wires are needed for sixteen outputs and another three for the inputs. So at the rate of 4 lambda for the spacing between two wires, each wire will require a width of 8 lambda units. The spacing of wires including both the thickness and the gap between them is often called the wiring pitch; i.e., so many lambda per wire. The wiring pitch in poly or metal1 or metal2 is not always the same.

The fourth row of cells from the top contains four AND gates which are placed with a distance of 20 lambda between each pair of them and thirty-two tristate buffers which constitute buffer 3 and buffer 4. Because all thirty-two buffers are the same, an array command is used instead of calling the cells thirty-two times. The command **:getcell trib1** will bring a tristate buffer onto the screen. A box is placed around trib1 with its right edge extended by 5 units. Now the command **:array 16 1** will create an array of 16 tristate buffers with a spacing of 5 units between each pair of them. The same procedure is followed to create buffer 4, but at a distance of 30 lambda from buffer 3 to distinguish easily between the two buffers. All tristate buffers of an array are selected or deselected together.

To select an up or down counter, the select bits s_1 and s_2 have to be supplied with a 1 and a 0. To select up-counter, $s_1 s_2 = 10$ With s_1 connected to VDD and also to the input of the inverter, s_2 is connected to 0 which is the output of the inverter. Similarly s_2 of down-counter is connected to VDD and s_1 to the output of the inverter. The connections from the 8-bit up-down counters are made only after placing buffers 1 and 2 along with the gated pull-downs and the data latches. Before proceeding any further with the connections, the VDD and VSS connections of the cells are completed. First, the VDD and VSS between cells are connected. Then VSS is brought out on the right hand side and connected to two power circuit pads at the corners.

To make the 16-bit to 64-bit mapping convenient, all the terminals of each counter are labelled as F(feed through), D(data), or Q(output), etc. The feedthrough's were helpful especially when a wire in one row has to be taken two rows above or below it.

Corresponding bits of all four buffers plus a gated pull-down plus a wire connecting to an output pad driver are connected together; this six-fold connection is repeated sixteen times constituting the on-chip output bus. To achieve this, the outputs of the gated pull-downs are connected to the corresponding outputs of the tristate buffer 1. They are also connected to the corresponding outputs of buffer2. In the same way, the outputs of buffers 3 and 4 are connected. All these outputs are brought out on the right hand side of the chip and connected together to get the final 16-bit output.

The completed designs are presented in Figs. 13 and 14. Fig. 13 shows the wiring with the pads and the cells unexpanded. Fig. 14 shows the fully expanded version of the layout design of the Data Generator.

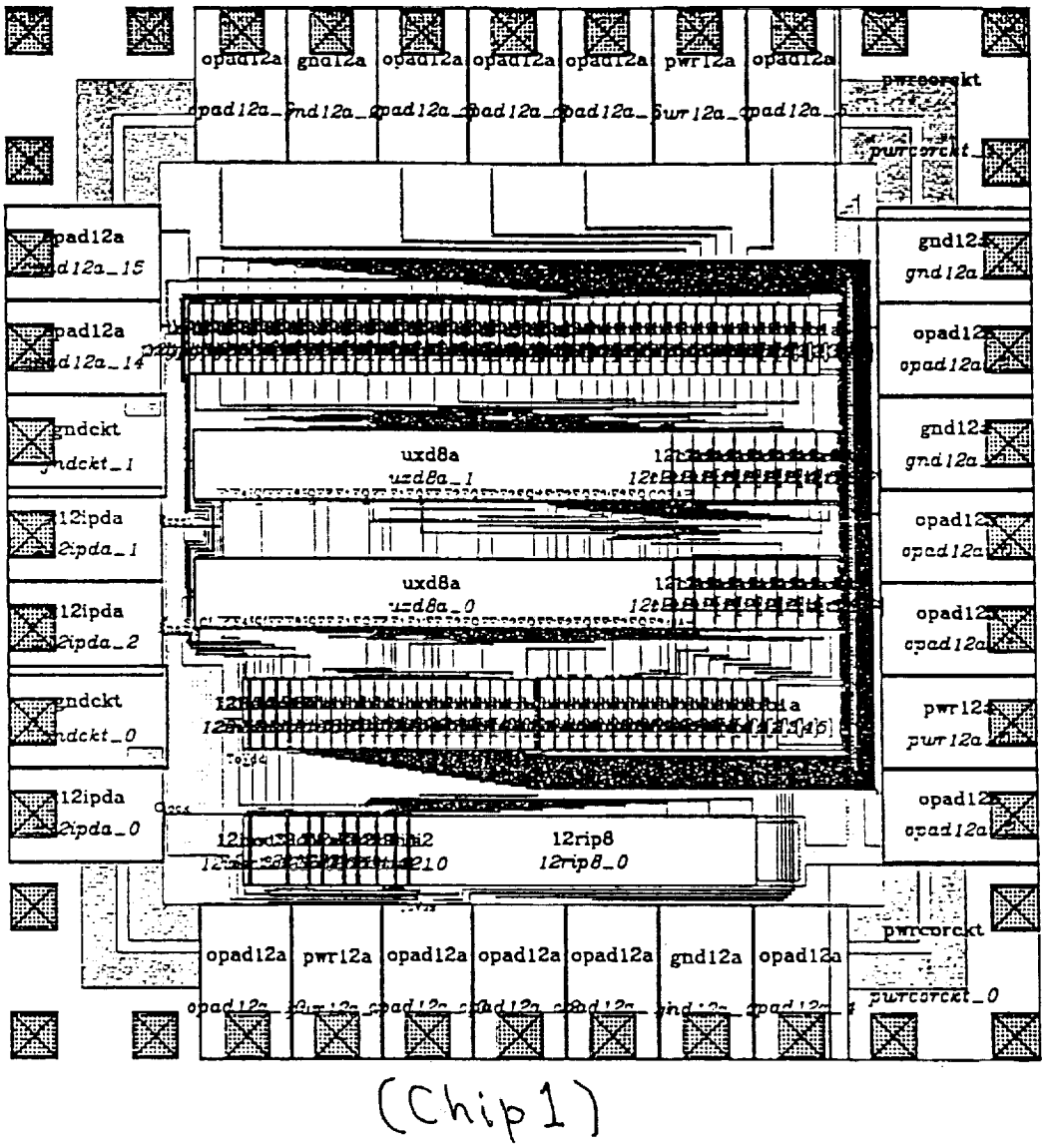
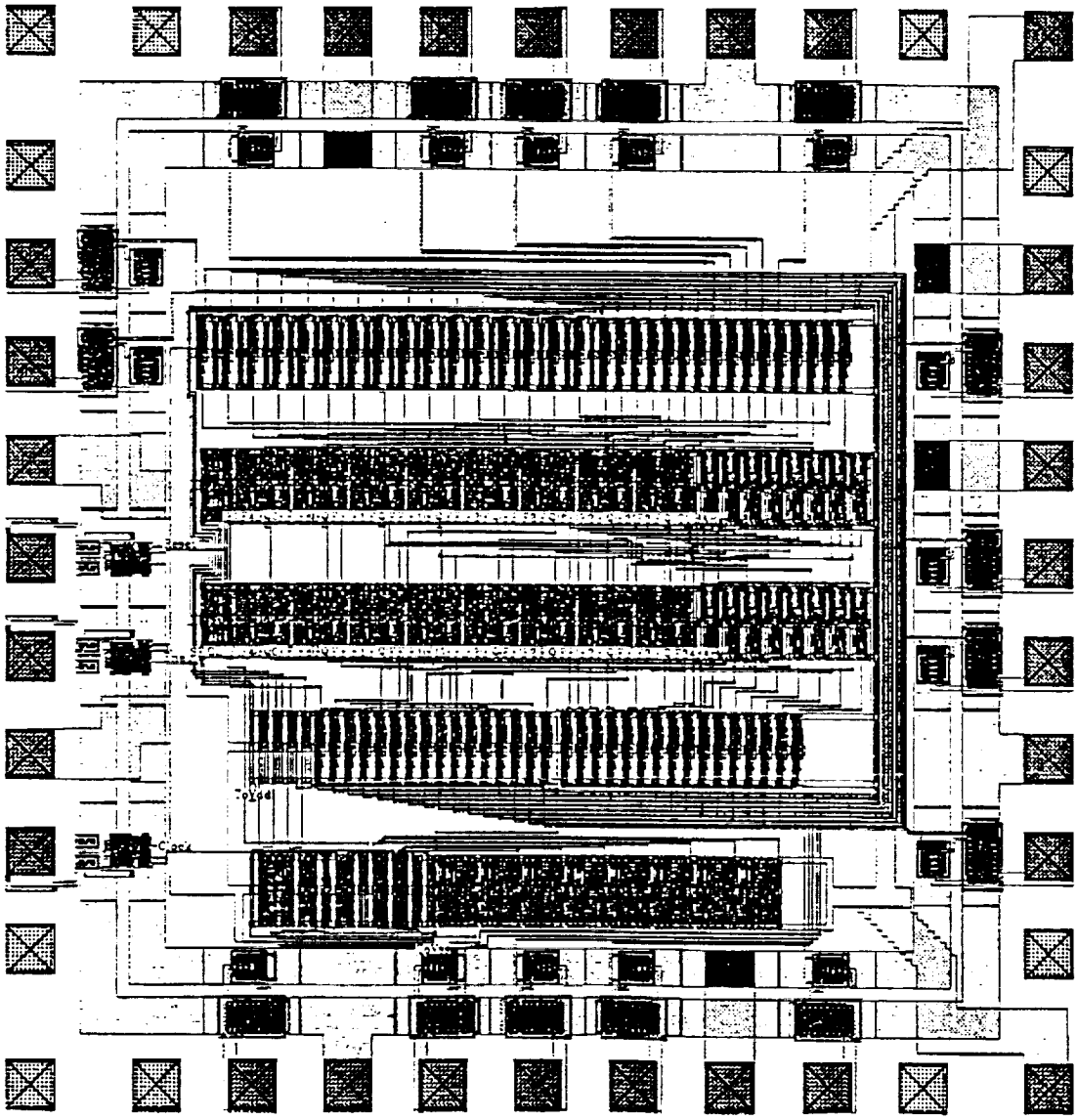


Fig.13 Layout of the Data generator with unexpanded cells and pads



(Chip 1)

Fig.14 Layout of the Data generator with expanded cells and pads

APPENDIX B

Technical Details of the Data Checker, Chip1 Plus Chip2.

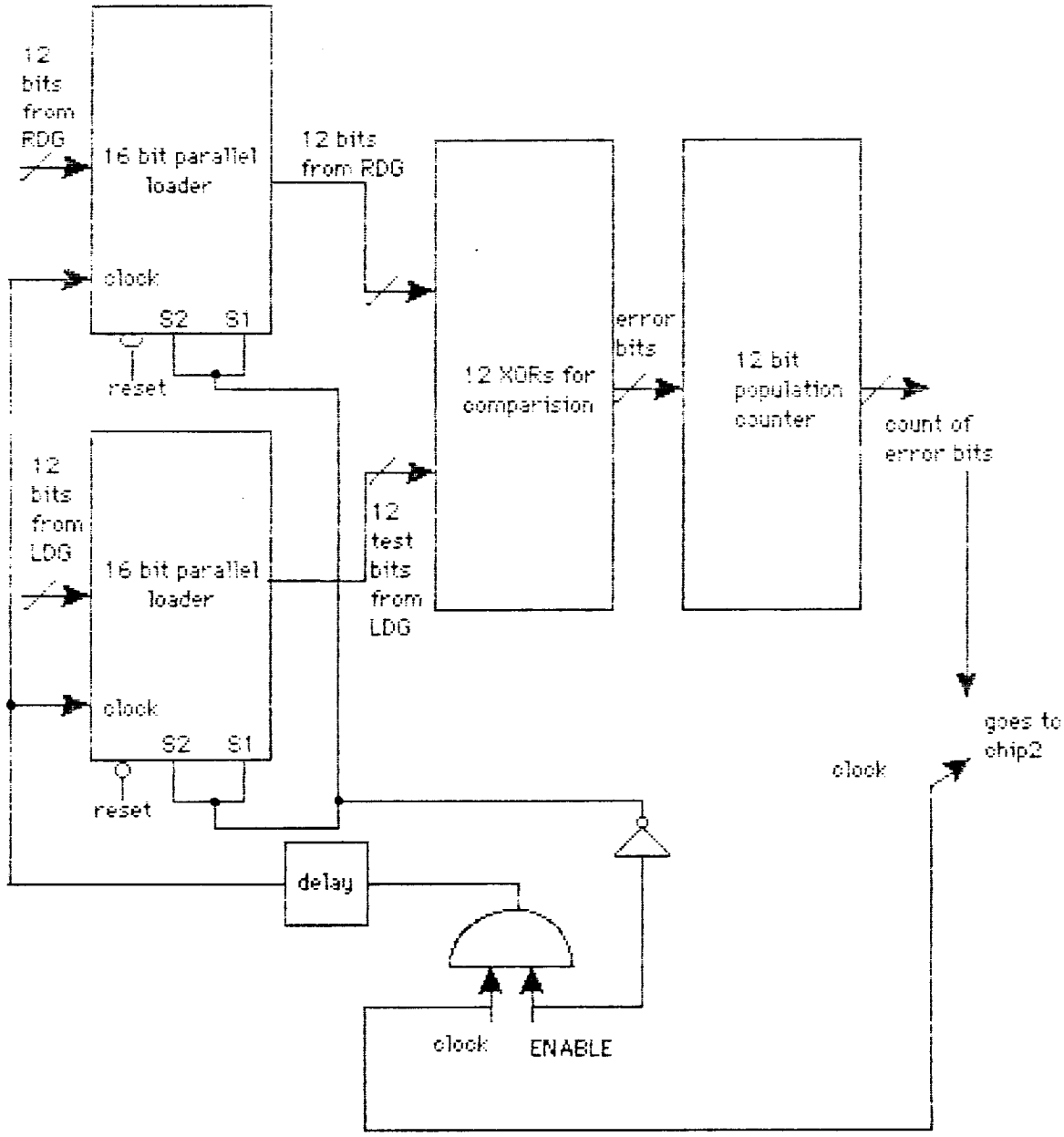


Fig. 3.
Block diagram of chip1.

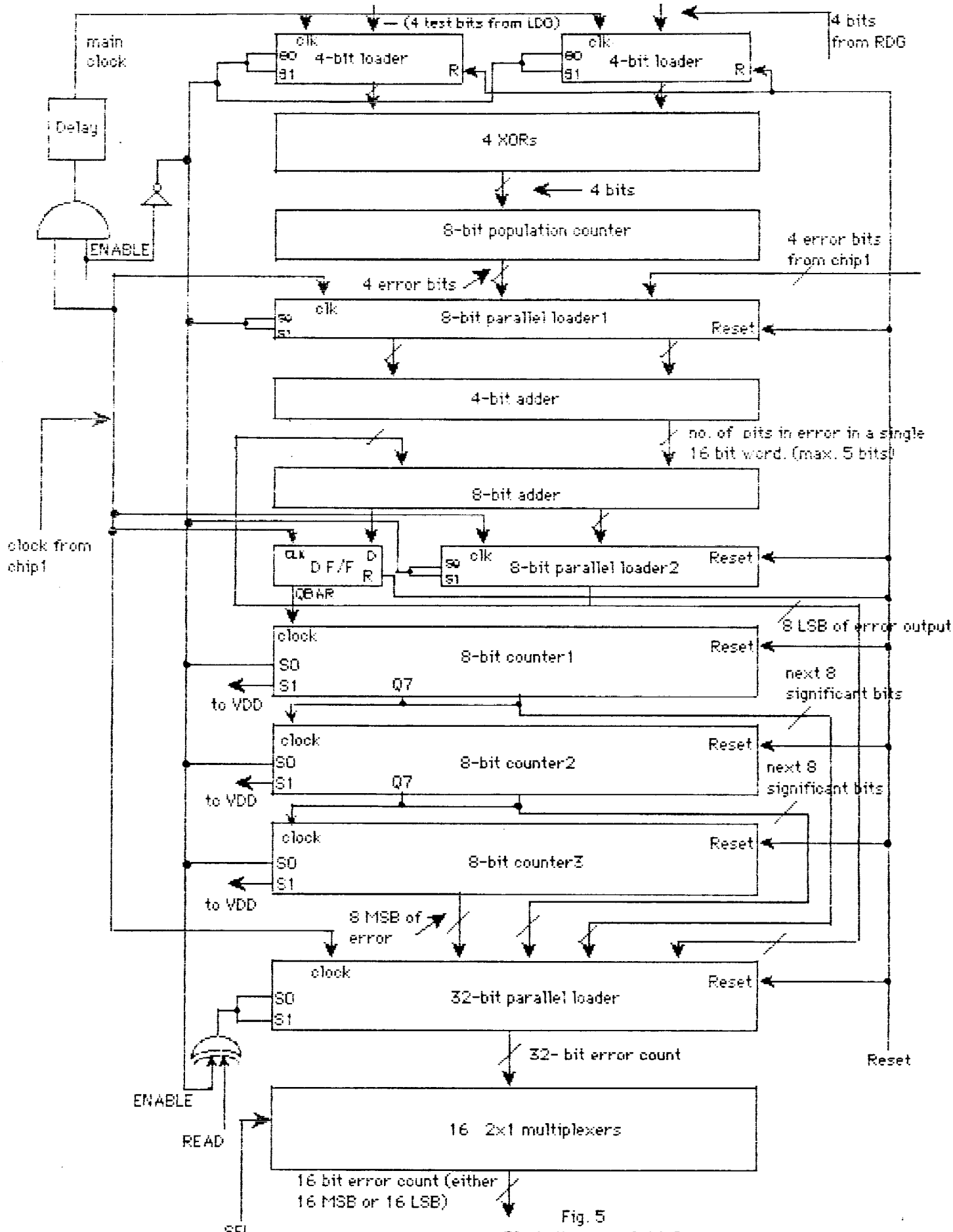


Fig. 5
Block diagram of chip2.

ORIGINAL PAGE IS
OF POOR QUALITY

Pin Assignments of Data Checker—Chip1

I/O Power	Pins 5, 35
I/O Ground	Pins 16, 26
Core Circuitry Power	Pins 15, 25
Core Circuitry Ground	Pin 34
Data I/P—A0	Pin 12
Data I/P—A1	Pin 13
Data I/P—A2	Pin 14
Data I/P—A3	Pin 17
Data I/P—A4	Pin 18
Data I/P—A5	Pin 19
Data I/P—A6	Pin 20
Data I/P—A7	Pin 21
Data I/P—A8	Pin 22
Data I/P—A9	Pin 23
Data I/P—A10	Pin 24
Data I/P—A11	Pin 27
Data I/P—B0	Pin 7
Data I/P—B1	Pin 6
Data I/P—B2	Pin 4
Data I/P—B3	Pin 3
Data I/P—B4	Pin 2
Data I/P—B5	Pin 1
Data I/P—B6	Pin 40
Data I/P—B7	Pin 39
Data I/P—B8	Pin 38
Data I/P—B9	Pin 37
Data I/P—B10	Pin 36
Data I/P—B11	Pin 33
Error Sum Out—S0	Pin 31
Error Sum Out—S1	Pin 32
Error Sum Out—S2	Pin 29
Error Sum Out—S3	Pin 30
System Clock In	Pin 8
Clock Out	Pin 28
System Reset In	Pin 10
Reset Out	Pin 11
System Enable In	Pin 9

NOTE: The data inputs are labeled A0—A11 and B0—B11 only as a matter of convenience; corresponding bits of two data words should be connected to the same numbered A and B inputs, but it is not necessary that the LSB's be connected to A0 and B0.

Tentative Pin Assignments of Data Checker—Chip2

I/O Power	Pin 40
I/O Ground	Pins 5, 26
Core Circuitry Power	Pins 15, 25
Core Circuitry Ground	Pin 35
Data I/P—A12	Pin 31
Data I/P—A13	Pin 32
Data I/P—A14	Pin 33
Data I/P—A15	Pin 34
Data I/P—B12	Pin 27
Data I/P—B13	Pin 28
Data I/P—B14	Pin 29
Data I/P—B15	Pin 30
Error Sum In—S0	Pin 39
Error Sum In—S1	Pin 38
Error Sum In—S2	Pin 37
Error Sum In—S3	Pin 36
Error Count Data Out—o0 (LSB)	Pin 24
Error Count Data Out—o1	Pin 23
Error Count Data Out—o2	Pin 22
Error Count Data Out—o3	Pin 21
Error Count Data Out—o4	Pin 20
Error Count Data Out—o5	Pin 19
Error Count Data Out—o6	Pin 18
Error Count Data Out—o7	Pin 17
Error Count Data Out—o8	Pin 16
Error Count Data Out—o9	Pin 14
Error Count Data Out—o10	Pin 13
Error Count Data Out—o11	Pin 12
Error Count Data Out—o12	Pin 11
Error Count Data Out—o13	Pin 10
Error Count Data Out—o14	Pin 9
Error Count Data Out—o15 (MSB)	Pin 8
Chip1 Clock In	Pin 7
Chip1 Reset In	Pin 6
System Clock In	Pin 3
System Enable In	Pin 2
System Select (I/P)	Pin 1
System Read (I/P)	Pin 4

NOTE: The Read command is active low. Select, when low, selects the two most significant bytes; when high, the two least significant bytes.

The tentative interchip wire list for the Data Checker chip pair is given below.

System	Chip1	Chip2	(Function)
Clock	Pin 8	Pin 3	
Enable	Pin 9	Pin 2	
Reset	Pin 10		
Read		Pin 4	
Select		Pin 1	
	Pin 28	Pin 7	Clock Out/In
	Pin 11	Pin 6	Reset Out/In
	Pin 31	Pin 39	Error Sum—S0
	Pin 32	Pin 38	Error Sum—S1
	Pin 29	Pin 37	Error Sum—S2
	Pin 30	Pin 36	Error Sum—S3

NOTE: The language Error Sum refers to the errors detected, if any, produced by any single comparison of the the least significant 12 bits of the data words done on Chip1; it does not refer to the running total of errors counted as maintained in the 32-bit output register.

CHAPTER II

LOGIC DESIGN OF DATA CHECKER

System specifications

The Data Checker works in conjunction with another chip called the Data Generator, as shown in Fig. 2. The Data Generator was previously designed at the University of Toledo [5]. It generates a 16-bit near random data stream at the rate of 13.2 MHz. This data stream is employed as a test signal in a simulated satellite relay link. The data is returned by the simulated satellite after processing. The Data Checker receives the 16-bit data from the satellite along with a duplicate of the original 16-bit data which is provided by the Local Data Generator, compares them, and keeps a running count of the total number of bits in error accumulated during the evaluation interval. The accumulated error count is stored in a 32-bit register. When interrogated, the register contents are multiplexed to a 16-bit bus as a double precision word. The Data Checker circuit also operates at 13.2 MHz.

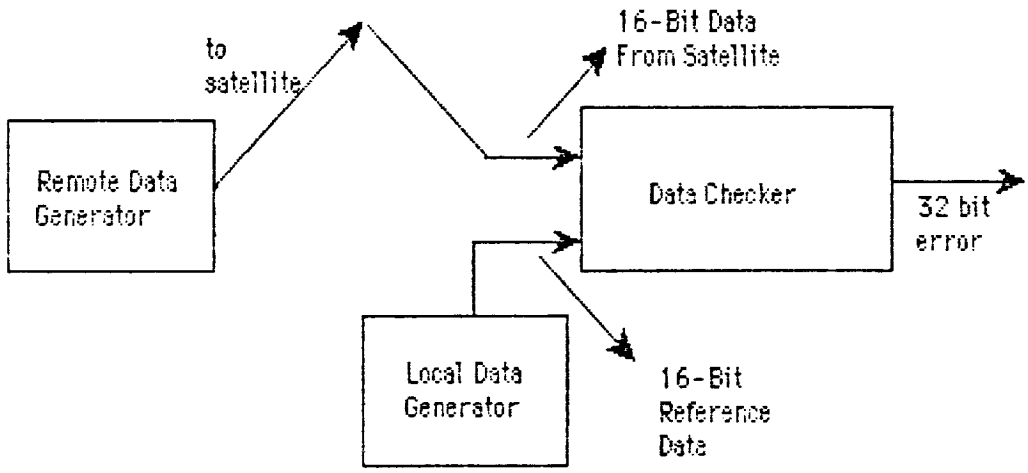


Fig. 2

Working of the Data Generator and Data Checker System

Initially the Data Checker is reset and disabled. The Local Data Generator (LDG) functions exactly in the manner of the Data Generator transmitting words to the sstellite (called the Remote Data Generator or RDG). The LDG generates the first word and is then disabled. This word is present at the input of the checker circuitry waiting for the word from the RDG to arrive. The arrival of a valid word from the RDG is indicated by an ENABLE signal which goes high when a valid word arrives. The ENABLE signal is provided by an independent controller. Once the ENABLE goes high, immediately all the blocks are enabled and the clock is applied. Accordingly, the first word will be latched in. At the same time the LDG is also enabled and it starts generating the second word. So by the time the second word from the RDG is at the input of the checker, the LDG has outputted the second word and it is available to the checker for comparison. Thus, while the checker is checking the Nth word the LDG is generating the (N+1)th word. When the data from the RDG ceases to be valid, the ENABLE signal automatically goes low disabling both the LDG and the checker. Again, when valid data are received, both circuits start functioning from the state in which they were halted.

Partitioning of the design

Because the Data Checker requires 32 input pins, 16 output pins, and some additional pins for signals like RESET, ENABLE, Clock, Power, and Ground, a 64-pin chip would be required. The cost of fabrication for a 64-pin prototype is approximately $58,000 while the cost of a 40-pin chip is about $5800. Due to this enormous difference in cost it was decided to use 40-pin chips. Since the design could not be accommodated on a single 40-pin chip, it was partitioned into two parts, and each part was implemented on a 40-pin chip. After a careful analysis the Data Checker was partitioned as follows.

1) Input the first 12 (least significant) bits of the 16-bit words from the RDG and the LDG to the first chip, find the number of bits in error and then output this number as an input to the second chip.

2) Use as inputs the 4 most significant bits from the RDG and the LDG. Determine the number of error bits occurring here. Add this to the number of error bits from chip1 to get the total number of bits in error produced by a single 16-bit word comparison. Additional circuitry is implemented in the second chip to keep a running count of errors and to read out the 32-bit output register as a sequence of two 16-bit words.

Both chips have identical control circuitry with the second chip having an extra command signal for reading the 16-bit multiplexed output plus a select signal to output first either the most significant 16 bits or the least significant 16 bits.

The complete design of both chip1 and chip2 can be divided into four major parts.

1) The logic to detect the number of bits in error resulting from a single 16-bit word comparison.
2) The logic to keep a running count of the total number of errors that have occurred since the last reset.
3) The output circuitry.
4) The control circuitry.

The first and fourth parts are common to chip1 and chip2 whereas parts 2 and 3 are implemented only within chip 2

Logic to Detect the Number of Bits in Error in a Single 16-bit Word

Chip 1

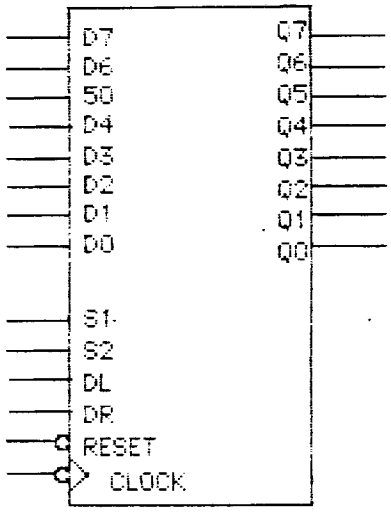
The complete logic diagram of the circuit on Chip 1 is shown in Fig. 3.* The least significant 12 bits from the RDG and the LDG are compared. Initially, the 12 bits from the LDG are at the input of the 16-bit parallel loader waiting for the 12 bits from the RDG to arrive.

A 16-bit shift register is configured as a 16-bit parallel loader. The configuration for an 8-bit shift register is shown in Fig. 4. The 16-bit shift register is configured like wise. When the select signals S1 and S0 are both low the inputs D0-D15 are parallel loaded on the falling edge of the clock. When the select signals S1 and S0 both go high then a hold is placed on the output; i.e., the output data remains the same irrespective of the inputs and the clock. When the word from the RDG arrives, the ENABLE signal goes high. Both parallel loaders are enabled and on the next falling edge of the clock the words are loaded.

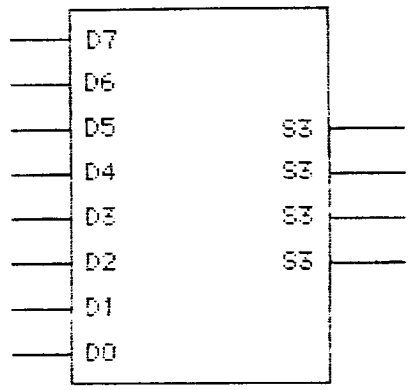
The comparison operation is done using twelve XORs. An XOR produces a zero output if both inputs are the same and an output of one if they are different. Thus, if the corresponding bits are not the same, i.e., in error, then the output of the corresponding XOR is one.

* Page B1

8 bit parallel loader



8 bit population counter



FUNCTION TABLE FOR 8 BIT PARALLEL LOADER

CLOCK	RESET	S1	S2	DR	D0-D7	DL	OPERATION	Q0-Q7
*	0	*	*	*	*	*	RESET	0 - 0
⌐¬	1	0	0	*	*	*	LOAD D0-D7	D0-D7
⌐¬	1	0	1	*	*	*	SHIFT RIGHT	DR-Q6(N-1)
⌐¬	1	1	0	*	*	*	SHIFT LEFT	Q1(N-1)-DL
⌐¬	1	1	1	*	*	*	HOLD	NO CHANGE

FUNCTION TABLE FOR 8 BIT POPULATION COUNTER

D0 - D7	S3	S2	S1	S0
M 1's = 0	0	0	0	0
M 1's = 1	0	0	0	1
M M 1's = 2	0	0	1	0
M M 1's = 3	0	0	1	1
M M 1's = 4	0	1	0	0
M M 1's = 5	0	1	0	1
M M 1's = 6	0	1	1	0
M M 1's = 7	0	1	1	1
M 1's = 8	1	0	0	0

Fig. 4
Block diagrams and function tables of the cells used in the design.

By counting the number of ones at the outputs of the XORs the total number of bits in error can be obtained. The error count can be put in binary form by using a population counter. An 8-bit population counter is shown in Fig. 4. The 12-bit population counter is similar to the 8-bit population counter in Fig. 4. Since there are 12 bits a 12-bit population counter is used. A population counter is a rather sophisticated digital circuit which produces a binary output equal to the number of true- or one-valued signals applied to its inputs. The counter employed here has 12 data inputs (D0-D11). The number appears at the outputs (S3-S0). The input data ones can appear in any order and still produce the correct result. However, any indeterminate input(s) will produce an unpredictable state at the output. This 4-bit error count is provided as inputs to the 8-bit parallel loader on the second chip.

In Chip2

The complete logic diagram of the circuit on chip2 is given in Fig. 5.* On chip2, the most significant 4 bits from the RDG and the LDG are compared. The operation is similar to the one on chip1. When the ENABLE goes high the 4-bit words are parallel loaded into 4 bit parallel loaders. These are derived from 4-bit shift registers and work exactly the same as the 16-bit parallel loaders explained earlier. The comparison is done using 4 XORs. If corresponding bits are not the same then the output of the corresponding XOR will be one. All the ones at the output of the XORs are counted by using an 8-bit population counter. Actually a 4-bit population counter could suffice. But as only an 8-bit population counter is available in the standard cell library, it is used. The remaining four inputs are permanently tied to logical zero (ground).

* Page B2

The 4-bit error count from chip1 and the 4-bit error count from chip2 are provided as inputs to the 8-bit parallel loader. The loader is introduced to avoid timing problems (discussed in Chapter 3). The outputs from the loader are provided as inputs to the 4-bit adder which adds the error count from chip1 (0-12) to the error count from chip2 (0-4) to give the 5-bit error count (including the carry) in a single 16-bit word.

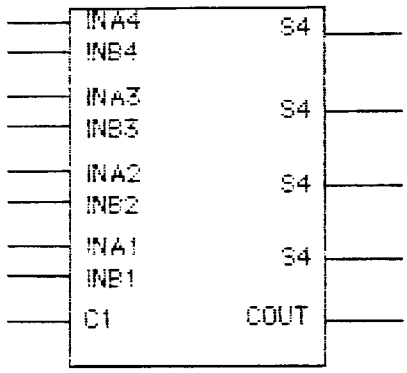
Logic to Keep a Running Count of the Total Number of Errors in an Interval

As mentioned previously, the total number of errors counted during a single comparison of two 16-bit words is obtained using a 4-bit adder. A 4-bit adder adds the two 4-bit numbers and produces a 5-bit output, a 4-bit sum and a carry bit.

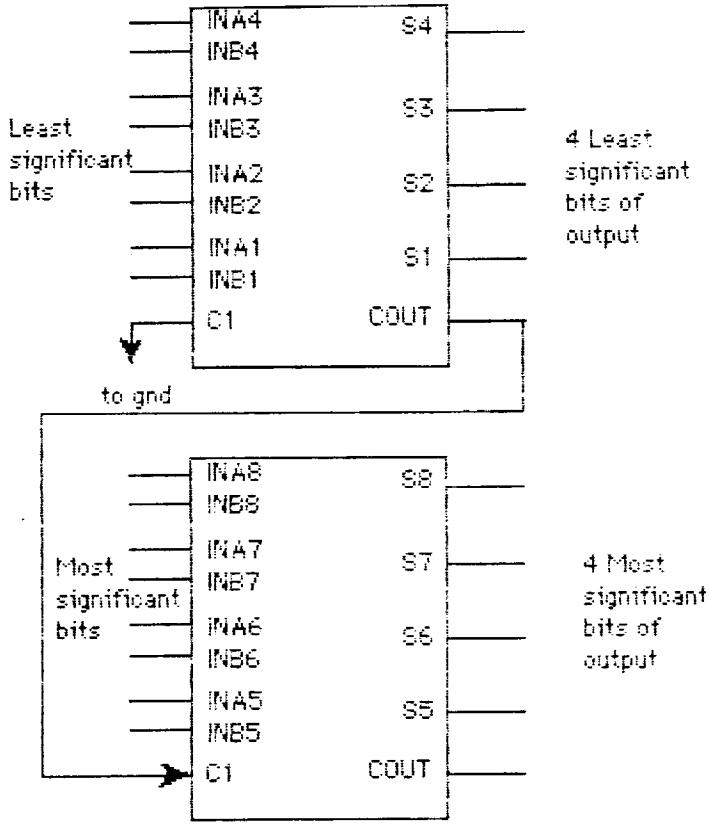
To maintain a running total of errors counted since the beginning of the current test interval, the error count from the most recent comparison is added to the previous total. This is done by using an 8-bit adder. The output of the adder is passed to an 8-bit register the output of which is then fed back as one input to the adder; the most recently determined error count is the other input. Because the immediate error count cannot exceed sixteen, only a 5-bit word is needed; the three most significant bits of this input are tied off to ground (logical zero).

An 8-bit adder is realized using two 4-bit adders as shown in Fig. 6. The least significant bits are added first using the 4-bit adder (upper block). Then the carry

4-BIT ADDER



8-BIT ADDER FROM TWO 4-BIT ADDER BLOCKS.

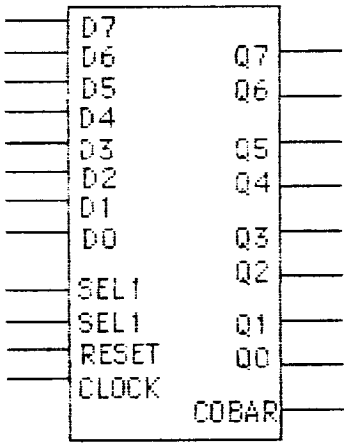


FUNCTION TABLE FOR 4 BIT ADDER

INAi	INAi	Ci	Si
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	0
1	0	1	0
1	1	1	1

Fig. 6
Block diagram and function tables for adder circuits used in the design.

8-BIT COUNTER



FUNCTION TABLE

CLOCK	D0 - D7	RESET	SEL1	SEL2	OPERATION
*	*	0	*	*	RESET
⌐	*	1	0	0	PARALLEL LOAD OF D0-D7
⌐	*	1	0	1	INCREMENT (COUNT UP)
⌐	*	1	1	0	DECREMENT (COUNT DOWN)
⌐	*	1	1	1	HOLD (STOP COUNT)

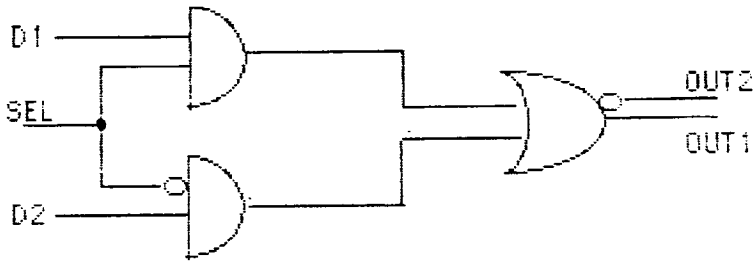
Fig. 7
Block diagram and function table of an 8-bit counter.

from that addition is supplied as carry-in input to the second 4-bit adder (lower block) which adds the 4 most significant bits. Thus we get an 8-bit sum and a carry bit as output.

The 8 bits are then parallel loaded into an 8-bit parallel loader. As in the case of the 16-bit parallel loader, an 8-bit shift register is used as a parallel loader by properly configuring the select signals S1 and S0. The stable and valid outputs from the loader are then fed back to the 8-bit adder as one set of inputs.

The outputs from 8-bit parallel loader 2 supplies the least significant 8 bits of the running total error count. The higher bits of the error count are obtained as follows.

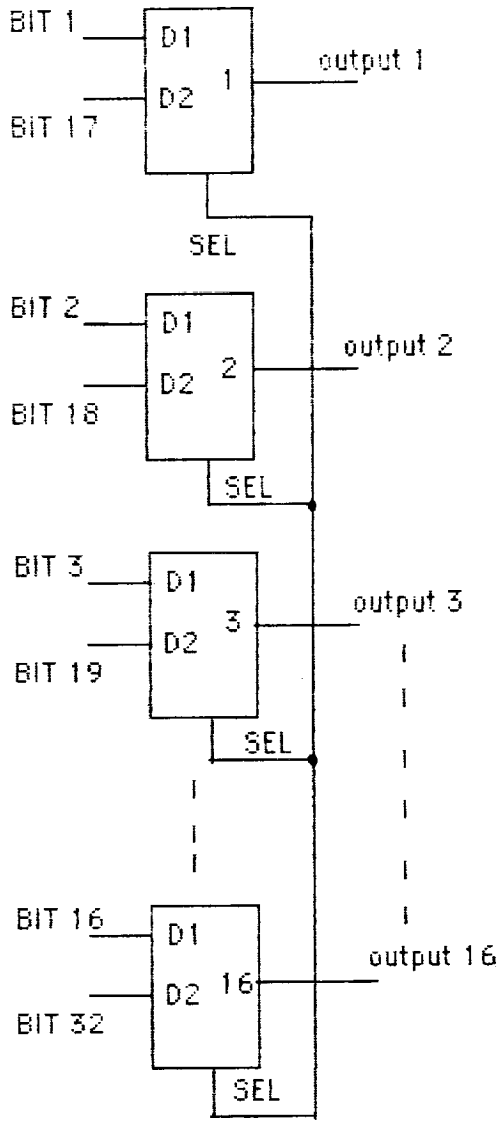
When the sum exceeds 8 bits a carry is generated by the 8-bit adder. The carry is fed to the data input of a D flip-flop. The Qbar output of the flip-flop is connected as a clock to 8-bit counter1 as in Fig. 7. The counter increments whenever it receives a falling edge at its clock input. Initially when the D flip-flop is reset, Qbar is at logic one. So the clock input to the counter is at logic high. When a carry is generated by the 8-bit adder, it is latched using the D flip-flop on the next clock cycle. So the Q output becomes high while the Qbar output goes low. Thus, the counter receives a falling edge and its count is incremented. The counter output thus gives the 8 next most significant bits of the error count. Every time the output of the 8-bit adder exceeds 255, a carry is generated which increments the count of 8-bit counter1. Thus the outputs from 8-bit counter1 plus those from 8 bit loader1 gives the 16 least significant bits of the total error count. The 16 most significant bits are obtained as follows.



FUNCTION TABLE

SEL	D1	D2	OUT1	OUT2
0	*	*	D2	$\overline{D2}$
1	*	*	D1	$\overline{D1}$

Fig. 8a
2X1 Multiplexer



When SEL is HIGH the 16 least significant bits are outputted.

When SEL is LOW the 16 most significant bits are outputted.

Fig. 8b
Illustration of how output is multiplexed.

The Q7 output of counter1 is applied as a clock to 8-bit counter2. During the total count sequence 0-255 the Q7 bit goes from low to high and from high to low once. Thus, whenever counter1 resets after the terminal count, Q7 goes from high to low which constitutes a falling clock edge for counter2 and it increments. The 8-bit output of this counter gives the 8 next most significant bits. The 8 most significant bits are similarly obtained by applying Q7 from counter2 as a clock to counter3.

The 32-bit output obtained in parts from the 8-bit adder and the three 8-bit counters is parallel loaded into a 32-bit parallel loader. The 32-bit parallel loader is achieved by using two 16-bit parallel loaders. Its function is similar to the parallel loaders used at the input.

Output Circuitry

The 32-bit output is to be outputted in two groups of 16 bits each. This can be done using 16 two-to-one multiplexers. A 2x1 multiplexer is shown in Fig. 8a. There are two inputs and either of them can be selected as the output using the select signal SEL. If SEL=0, D2 appears at output1. If SEL=1, D1 appears at output1. As we want 32 outputs to be multiplexed into two groups of 16 bits each, we need sixteen 2x1 multiplexers. Also, we want either the 16 least significant bits to be outputted first and then the 16 most significant bits or vice versa. So the inputs to the multiplexer are given as shown in Fig. 8b. When SEL=0, the 16 most significant bits are selected. When SEL=1, the 16 least significant bits are selected.

Control Circuitry

The control circuitry consists of two parts.

1) Circuitry to Disable the Data Checker and Halt its Operation

The clock signal is ANDed with the ENABLE signal and the output of the AND gate is applied as a clock to the rest of the Data checker circuitry on chip1 and chip2. As long as a valid word is at the input of the checker the ENABLE signal is nigh and the clock is applied to the checker circuitry. Whenever invalid data arrives, the ENABLE goes low preventing the clock signal from being applied to the circuitry. However, simply ANDing the clock signal with the ENABLE signal results in an unwanted falling edge if the clock is high when the ENABLE goes low. This falling edge will be applied to the circuit when actually we want to discard it. This would result in a wrong error count. To avoid this, the output of the AND gate is applied to the checker circuitry through a delay circuit. Also, the ENABLE signal is inverted and applied to the 16-bit parallel loaders at the input, to 8 bit parallel loader1 and to the 8-bit counters (counter1, counter2, counter3) and it is also applied at the input of the XOR gate which is used to control the read operation (explained later).

So, when the ENABLE goes low, a high at the output of the inverter is applied to the select inputs S1 and S0 of the 16-bit and the 8-bit parallel loaders. Also, a high is applied to the S1 input of the counters making their select inputs S1 and S0 high. Thus, the 16- and the 8-bit loaders along with the counters are forced into a halted state. So even if the clock signal is applied their outputs will not

change. So whenever the ENABLE goes low the loaders and the counters are stopped. As the output of the AND gate goes through the delay circuit, before the unwanted falling edge is received by the counters and loaders, they already are in a halted state. Thus, there is no effect from the unwanted falling edge. Again, when a valid word arrives at the input, the ENABLE goes high. So the loaders and counters again resume their normal operation before the clock is applied.

2) Circuitry to Read the Output

The output read operation may be a slow process and take more than one clock cycle. So there should be a stable valid output until the read operation is complete. At the same time the running count of errors should not be lost. This is achieved as follows. The output is obtained from a 32-bit parallel loader (realized using two 16-bit parallel loaders). The select signals S1 and S0 are obtained from the output of an XOR gate whose inputs are the inverted ENABLE signal and the READ signal.

Under normal operation the READ signal is high and the ENABLE signal is high. So the output of the XOR gate is low. Thus both S1 and S0 are set low and the block functions as a parallel loader. Whenever output is to be read, the READ signal is made low while ENABLE remains high; this forces the output of the XOR gate along with S1 and S0 of the 32-bit parallel loader to go high. The loader is thus stopped and maintains its data constant even though the clock is still being applied. Now using the SEL signal, either the 16 least significant bits or the 16 most significant bits are selected and dumped to the output bus. After the first selection and a high or a low word has been read, SEL is complemented, and the

second read is done. The operation is complete, READ returns to a high state, and the 32-bit loader resumes its normal operation. While the 32-bit loader is halted for a read, 8-bit parallel loader2 plus 8-bit counters 1, 2, and 3 maintain the running count correctly without interruption; this count is then passed along normally as soon as the big loader is again able to receive data.

Thus the logic design is completed. The next chapter describes the propagation delays and the timing diagrams for the circuitry.

CHAPTER III

DELAYS AND TIMING INFORMATION

Propagation delay plays a vital role in any ASIC design. Under ideal conditions with no propagation delay through the gates, the logic of the design may be functionally correct but under real world conditions may be far from ideal. Each signal is delayed by a certain amount of time as it passes through a logic level. Each signal has a finite rise and fall time associated with it. These can cause logic errors and may impose a limit on the speed of operation. So an accurate estimate of timing relationships is essential to make sure that the logic design performs correctly under practical conditions.

Conceptually, an ASIC consists of a number of functional blocks (parts or gates), each of which takes one or more input signals and produces one or more outputs. Each output drives the inputs of one or more other gates, on occasion, an external output. The sum of the capacitances of the driven gates and the external outputs is called the absolute fanout of the driving output. The propagation delay through the gates or the functional blocks comprising an ASIC is related to this absolute fanout.

The propagation delay depends on circuit design, transistor sizing, number of levels of logic, temperature, supply voltages, variations in process dependent parameters, and capacitive loading of the output of the gate. High temperature leads to reduced carrier mobilities, higher resistance, and longer delays. Also, low supply voltages and slow rise/fall times of driving inputs also lead to longer propagation delays.

The CMOSN cell notebook provides equations that predict the delays from input to output and the rise and fall times of the output signal. Since the design of the Data Checker is done using standard cells, the cell equations are used for calculating the delays.

The Following timing and cell equation information is taken from the CMOSN cell notebook.

Calculation of Delays for Chip1 and Chip2

By using the delay equations, the propagation delays for all the various logic blocks are calculated for the worst case process parameters at 125 degrees Celsius and typical case process parameters at 25 degrees Celsius. These delays for chip1 are listed in Table1 and Table 2 and for chip2 in Table 3, Table 4, Table 5, and Table 6. The delays through the individual blocks are then added to get the total delays from inputs to outputs.

a) For chip1-

The delay from the input to the output of chip1 is given by the sum of the delays through the input pad, 16-bit parallel loader, XOR gate, 12-bit population counter, and output pad. It comes to 56.9601 ns at 125 degrees Celsius (worst case) and 17.0488 at 25 degrees Celsius (typical case).

b) For chip2-

The propagation delays in chip2 are calculated in four stages.

1) Delay from the input to 8-bit parallel loader 1.

This delay is given by the sum of the delays through the input pad, the 4-bit

TABLE 1

Timing Information for Standard Cells Used in Data Checker Chip1 @ 125 Degrees Celsius

CELL NAME	EQUATIONS USED (@ 25 DEG. C)	Pd (0-1) ns	Pd (1-0) ns	Tris ns	Tfal ns
16-Bit* Parallel Loader	Pd(0-1) = 11.9 + (1.9)CL Pd(1-0) = 14.9 + (1.47)CL Tris = 3.42 + (3.96)CL Tfal = 4.18 + (2.13)CL	12.565	10.4145	4.806	4.9255
XOR	Pd(0-1) = 2.10 + (3.58)CL Pd(1-0) = 1.38 + (2.22)CL Tris = 3.35 + (8.53)CL Tfal = 2.33 + (4.70)CL	4.1048	2.6232	8.1268	4.962
12-bit population counter	Pd(0-1) = 32.79 + (1.85)CL Pd(1-0) = 31.23 + (1.74)CL Tris = 3.02 + (3.56)CL Tfal = 3.27 + (2.19)CL	35.01	33.318	7.292	5.898
Output Pad	Pd(0-1) = 2.29 + (0.08)CL Pd(1-0) = 2.04 + (0.06)CL Tris = 1.04 + (0.19)CL Tfal = 0.08 + (0.13)CL	6.45	5.16	10.92	6.84

* = The delay equation for Pd(1-0) & Tfal is the one between RESET and Q0 but not the one between CLOCK and Q0.

TABLE 2

Timing Information for Standard Cells Used in Data Checker Chip1 @ 25 Degrees Celsius

CELL NAME	EQUATIONS USED (@ 25 DEG. C)	Pd (0-1) ns	Pd (1-0) ns	Tris ns	Tfal ns
16-Bit* Parallel Loader	Pd(0-1) = 3.68 + (0.62)CL Pd(1-0) = 4.78 + (0.56)CL Tris = 0.90 + (1.4)CL Tfal = 1.29 + (0.90)CL	3.897	4.976	1.39	1.605
XOR	Pd(0-1) = 0.70 + (1.06)CL Pd(1-0) = 0.53 + (0.79)CL Tris = 1.11 + (2.55)CL Tfal = 1.02 + (1.61)CL	1.2936	.9724	2.538	1.9216
12-bit population counter	Pd(0-1) = 9.02 + (0.60)CL Pd(1-0) = 8.60 + (0.60)CL Tris = 0.87 + (1.25)CL Tfal = 0.94 + (0.91)CL	9.74	9.32	2.37	2.032
Output Pad	Pd(0-1) = 1.01 + (0.02)CL Pd(1-0) = 0.70 + (0.02)CL Tris = 0.43 + (0.06)CL Tfal = 0.45 + (0.05)CL	2.05	3.55	1.74	3.04

* = The delay equation for Pd(1-0) & Tfal is the one between RESET and Q0 but not the one between CLOCK and Q0.

TABLE 3

Timing Information for Standard Cells Used in Data Checker Chip2 @ 125 Degrees Celsius.

CELL NAME	EQUATIONS USED (@ 125 DEG. C)	Pd (0-1) ns	Pd (1-0) ns	Tris ns	Tfal ns
4-Bit * Parallel Loader	Pd(0-1) = 9.36 + (1.90)CL Pd(1-0) = 10.59 + (1.59)CL Tris = 3.41 + (3.97)CL Tfal = 3.32 + (2.45)CL	10.025	10.5565	4.799	4.1775
XOR	Pd(0-1) = 2.10 + (3.58)CL Pd(1-0) = 1.38 + (2.22)CL Tris = 3.35 + (8.53)CL Tfal = 2.33 + (4.70)CL	4.069	2.601	8.041	4.915
8-bit population counter	Pd(0-1) = 14.1 + (1.83)CL Pd(1-0) = 18.83 + (1.28)CL Tris = 1.26 + (4.20)CL Tfal = 0.98 + (2.54)CL	14.46	19.086	2.1	1.488
8-bit * parallel loader	Pd(0-1) =10.22 + (1.9)CL Pd(1-0) = 11.40 + (1.46)CL Tris = 0.43 + (0.06)CL Tfal = 0.45 + (0.05)CL	11.075	12.057	5.17	4.971
4-bit adder	Pd(0-1) =7.29 + (3.26)CL Pd(1-0) = 9.60+ (2.27)CL Tris = 4.13 + (8.41)CL Tfal = 3.97 + (6.16)CL	8.757	10.6215	7.91	6.742

* = The delay equation for Pd(1-0) & Tfal is the one between RESET and Q0 but not the one between CLOCK and Q0.

TABLE 4

Timing Information for Standard Cells Used in Data Checker Chip2 @ 125 Degrees Celsius.

CELL NAME	EQUATIONS USED (@ 125 DEG. C)	Pd (0-1) ns	Pd (1-0) ns	Tris ns	Tfal ns
8-Bit * adder	Pd(0-1) = 7.29 + (3.26)CL Pd(1-0) = 9.60 + (2.27)CL Tris = 4.13 + (8.41)CL Tfal = 3.97 + (6.16)CL	15.885	20.0172	11.62	10.404
8-bit parallel loader2	Pd(0-1) = 10.22 + (1.9)CL Pd(1-0) = 11.40 + (1.46)CL Tris = 3.37 + (4.0)CL Tfal = 3.99 + (2.18)CL	11.455	12.349	5.97	5.407
8-bit counter	Pd(0-1) = 11.08 + (1.88)CL Pd(1-0) = 12.50 + (1.42)CL Tris = 4.50 + (3.90)CL Tfal = 4.10 + (2.15)CL	11.7994	13.139	5.982	4.917
32-bit * parallel loader	Pd(0-1) =11.9 + (1.9)CL Pd(1-0) = 14.9 + (1.47)CL Tris = 3.42 + (3.96)CL Tfal = 4.18 + (2.13)CL	12.299	15.208	4.251	4.627
2X1 multiplexer	Pd(0-1) =4.63 + (3.52)CL Pd(1-0) = 3.50+ (1.81)CL Tris = 8.36 + (8.41)CL Tfal = 6.27 + (4.43)CL	8.854	5.672	18.45	11.586

* = The delay equation for Pd(1-0) & Tfal is the one between RESET and Q0 but not the one between CLOCK and Q0.

TABLE 5

Timing Information for Standard Cells Used in Data Checker Chip2 @ 25 Degrees Celsius.

CELL NAME	EQUATIONS USED (@ 25 DEG. C)	Pd (0-1) ns	Pd (1-0) ns	Tris ns	Tfal ns
4-Bit * Parallel Loader	Pd(0-1) = 2.79 + (0.63)CL Pd(1-0) = 3.06 + (0.59)CL Tris = 0.89 + (1.41)CL Tfal = 0.95 + (1.03)CL	3.0105	3.2665	1.383	1.3105
XOR	Pd(0-1) = 0.70 + (1.06)CL Pd(1-0) = 0.53 + (0.79)CL Tris = 1.11 + (2.55)CL Tfal = 1.02 + (1.61)CL	1.283	.9645	2.512	1.905
8-bit population counter	Pd(0-1) = 4.0 + (0.59)CL Pd(1-0) = 5.06 + (0.50)CL Tris = 0.30 + (1.52)CL Tfal = 0.25 + (1.14)CL	4.118	5.16	0.604	0.478
8-bit * parallel loader	Pd(0-1) =3.08 + (0.62)CL Pd(1-0) = 3.58 + (0.57)CL Tris = 0.90 + (1.4)CL Tfal = 1.19 + (0.94)CL	3.359	3.8365	1.53	1.613
4-bit adder	Pd(0-1) =2.10 + (1.01)CL Pd(1-0) = 2.76+ (0.97)CL Tris = 1.03 + (2.72)CL Tfal = 1.16 + (1.96)CL	2.5545	3.1965	2.25	2.042

* = The delay equation for Pd(1-0) & Tfal is the one between RESET and Q0 but not the one between CLOCK and Q0.

TABLE 6

Timing Information for Standard Cells Used in Data Checker Chip2 @ 25 Degrees Celsius.

CELL NAME	EQUATIONS USED (@ 25 DEG. C)	Pd (0-1) ns	Pd (1-0) ns	Tris ns	Tfal ns
8-Bit adder *	Pd(0-1) = 2.10 + (1.01)CL Pd(1-0) = 2.76 + (0.97)CL Tris = 1.033 + (2.72)CL Tfal = 1.16 + (1.96)CL	4.604	5.908	3.148	3.104
8-bit parallel loader2	Pd(0-1) = 3.08 + (0.62)CL Pd(1-0) = 3.58 + (0.57)CL Tris = 0.90 + (1.40)CL Tfal = 1.19 + (0.94)CL	3.493	3.9505	1.91	1.801
8-bit counter	Pd(0-1) = 3.32 + (0.63)CL Pd(1-0) = 3.97 + (0.56)CL Tris = 1.20 + (1.36)CL Tfal = 1.23 + (0.94)CL	3.5594	4.1828	1.716	1.587
32-bit * parallel loader	Pd(0-1) =3.68 + (0.62)CL Pd(1-0) = 4.78 + (0.56)CL Tris = 0.90 + (1.4)CL Tfal = 1.29 + (0.90)CL	3.8102	4.8976	1.194	1.479
2X1 multiplexer	Pd(0-1) =1.33 + (1.10)CL Pd(1-0) = 4.78+ (0.56)CL Tris = 2.27 + (2.64)CL Tfal = 1.85 + (1.63)CL	2.65	2.178	5.438	3.806

* = The delay equation for Pd(1-0) & Tfal is the one between RESET and Q0 but not the one between CLOCK and Q0.

parallel loader, the XOR gate, and the 8-bit population counter. It comes to 33.7115 ns at 125 degrees Celsius (worst case) and 9.7095 ns at 25 degrees Celsius (typical case).

2) Delay from parallel loader 1 to parallel loader 2

It is given by the sum of the delays through loader1, the 4-bit adder, and the 8-bit adder. It comes to 42.6957 ns at 125 Celsius and 12.9022 at 25 degrees Celsius (typical case).

3) Delay from loader 2 to the 32-bit parallel loader.

This is given by the sum of the delays through loader2, 8-bit counter1, counter2, and counter3. Its value is 51.766 ns at 125 degrees Celsius (worst case) and 18.7023 ns at 25 degrees Celsius (typical case).

4) Delay from the 32-bit parallel loader to the output.

It is the sum of the delays through the 32-bit loader and the 2x1 multiplexer. It comes to 24.0627 ns at 125 degrees celsius (worst case) and 7.5476 ns at 25 degrees celsius (typical case).

The clock period is 75.75 ns (frequency of 13.2 MHz). From the delay calculations above it is clear that the delay between successive stages is less than 75.75 ns. Since the stages are separated by parallel loaders which act as latches, inputs to the loaders are stable and valid at the time the clock is applied to them. Also data is present at the input to these loaders for the minimum data setup time before the clock is applied and for the minimum data hold time after the clock is applied. So, whenever the clock is applied to the loaders, the data at their inputs is stable and valid and this data is passed on to the next stage.