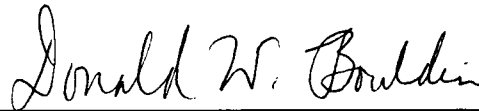


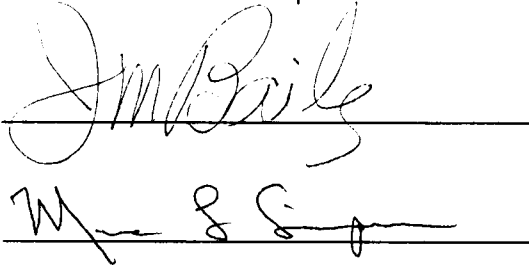
To the Graduate Council:

I am submitting herewith a thesis written by Gary Allen Armstrong entitled "Design and Implementation of a Moment Invariant Algorithm in VLSI for Pattern Recognition Applications." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.



Donald W. Bouldin, Major Professor

We have read this thesis
and recommend its acceptance :



Accepted for the Council :



Vice Provost
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Gary A. Armstrong
Date 11/30/89

DESIGN AND IMPLEMENTATION OF A MOMENT INVARIANT ALGORITHM IN VLSI FOR PATTERN RECOGNITION APPLICATIONS

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Gary Allen Armstrong
December 1989

ACKNOWLEDGMENTS

The design described in this thesis was accomplished while the author was a member of the engineering staff of Oak Ridge National Laboratory (ORNL) at Oak Ridge, Tennessee. This facility is operated by Martin Marietta Energy Systems, Inc., for the U.S. Department of Energy.

The author wishes to thank the members of his Committee, Dr. D. W. Bouldin, Dr. M. L. Simpson, and Dr. M. L. Bailey for their leadership, support, and advice through the development of the moment generator chip and the writing of this thesis. Dr. D. W. Bouldin contributed excellent advice aiding the design and architectural implementation of the chip. Dr. M. L. Simpson provided significant leadership in describing the mechanics of the moment invariants algorithm. And special thanks go to C. L. Britton and D. F. Newport of ORNL for their insight into the design of high performance CMOS devices and the use of simulation tools for verification of CMOS designs.

ABSTRACT

VLSI technology provides the capability to place from 10 up to 100,000 transistors on a single silicon circuit, referred to as an application-specific integrated circuit (ASIC). This allows incorporating into a single chip many algorithms that were previously limited to designs primarily implemented by special-purpose computers or dedicated processor boards. With CMOS transistor internal switching, speeds reaching 100 to 120 MHz, the algorithms incorporated into VLSI ASICs also achieve a substantial performance improvement over discrete device implementations of von Neumann architectures. In addition to the performance improvements offered, VLSI design can be customized to fit the specific needs of the application. It can then be integrated on a single chip, which can be completely simulated by the VLSI CAD tools in less time than is required to build a prototype with discrete TTL chips.

With the increase in use of pattern recognition for image-processing applications, there is a growing need for special-purpose, high-performance processors that can execute these algorithms in real time. Moment invariance is a computational intensive algorithm that requires data compression and unique representations of digitized images. Moment invariant equations are analyzed for use in pattern recognition for flaw detection in digitized images as presented by Hu and Maitra. The moment invariant algorithm compresses the image to six constants, while providing a unique representation of the image that is invariant to translation, scaling, rotation, illumination, and contrast. The digitized images have 256 grey scales and a resolution of 512 x 512 pixels. Moment invariant processing at frame rates requires the use of special-purpose processors capable of a high throughput of 7.68 Megapixels/s. Methods for implementing the moment invariant algorithm into VLSI are examined, and architectures for implementing the equations are considered. After determining the most crucial time element, of the moment algorithm, the calculation is implemented in a 40-pin VLSI ASIC. Implementation of the moment calculation will allow the use of multiple VLSI ASICs in parallel. Thirty calculations are necessary to determine the moments necessary for input into the Hu and Maitra equations. This results in a total parallel-processing capability of 235.8 Million operations per second (MOPS). The use of pipelining in the design allows the multiplication function and the summation functions to be separated so that each can perform in parallel. VLSI design considerations for implementing the moment calculation are emphasized.

TABLE OF CONTENTS

CHAPTER		PAGE
1. INTRODUCTION		1
1.1 Motivation		1
1.2 Chapter Contents		2
1.3 Appendix Contents		2
1.4 Definition of Terms		2
2. BACKGROUND		7
2.1 The Need for Performance Improvement		7
2.2 Justification for the Use of VLSI		8
3. MOMENT INVARIANTS		9
3.1 Introduction		9
3.2 Moment Properties of Images		10
3.3 Moment Invariant Algorithms		10
3.3.1 Hu Algorithm		13
3.3.2 Maitra Algorithm		14
3.4 Equation Statistics		16
3.5 Summary		18
4. MOSFET MECHANICS		19
4.1 Comparison of CMOS to nMOS		19
4.2 MOSFET Device Description		21
4.3 CMOS Inverter		24
4.4 CMOS Device Model Parameters		29
4.5 CMOS Transmission Gate		33
4.6 Conclusions		33

5. ARCHITECTURE.....	35
5.1 Conventional VLSI Architectures.....	35
5.2 Multiplication Architectures	39
5.3 Physical Design Limitations.....	42
5.4 Conventional Moment Generator Architecture.....	43
5.5 Conclusions.....	46
6. DESIGN AND IMPLEMENTATION.....	48
6.1 Design Goals.....	48
6.1.1 Performance Goals.....	48
6.1.2 Area Goals	50
6.1.3 I/O Design Goals.....	50
6.1.4 Power Design Goals	50
6.2 Design for Testability	53
6.4 Adder Cell Design	58
6.4.1 Conventional Adder Cells	58
6.4.2 Manchester Carry Conditions	60
6.4.3 Manchester Carry Conditions Adder	63
6.5 Multiplier Design.....	65
6.6 Register Cell design	67
6.6.1 Static Register Cell Amplification Factor.....	69
6.7 Clocking Scheme.....	70
7. SIMULATION AND TESTING.....	73
7.1 Simulation.....	74
7.1.1 Design Rule Checkers (DRC)	74
7.1.2 ESIM the Event Simulator	75
7.1.3 Compiled Logic and Fault Simulator for	76
7.1.4 COSMOS Logical Simulator Test of	77
7.1.5 Pspice the Analog Simulator	78
7.2 Testing.....	80
7.2.1 Using Scan Paths to test the Functions.....	81

CHAPTER	PAGE
8. CONCLUSIONS	82
8.1 Summary of the Prototype Design.....	83
8.2 Integration of Tools.....	84
8.3 Future Work.....	85
LIST OF REFERENCES	87
APPENDIXES	90
APPENDIX C COSMOS Logic Simulator Files.....	93
APPENDIX D COSMOS Fault Simulator Files.....	95
APPENDIX E Pspice Simulator Control Files.....	97
APPENDIX F Pspice Simulator Waveforms	103
VITA	113

LIST OF TABLES

TABLE	PAGE
Table 3.1 Moment Calculations	17
Table 3.2 Hu Algorithm.....	17
Table 3.3 Maitra Algorithm	18
Table 4.1 Comparison of CMOS to nMOS.....	20
Table 6.1 I/O Pin Assignments.....	51
Table 6.2 Pspice Simulator Power and Loads	52
Table 6.3 Adder Cell Buffer Addition Effects.....	65
Table 6.4 Amplification Factor Test.....	69
Table 7.1 Manchester Carry Conditions Adder Cell Propagation Time Chart.....	79
Table 7.2 Static Register Cell Propagation Time Chart	79

LIST OF FIGURES

FIGURE	PAGE
Figure 4.1 CMOS Inverter Schematic.....	24
Figure 4.2 P-Channel MOSFET I_{ds} Curves.....	25
Figure 4.3 N-Channel MOSFET I_{ds} Curves.....	26
Figure 4.4 Graphical Solution for Inverter Current with Equal Size Devices	27
Figure 4.5 Graphical Solution for Inverter Current with Ideal Normalized Device	28
Figure 4.6 CMOS Inverter Output Current and Voltage Relationships.....	30
Figure 4.7 CMOS Transmission Gate.....	34
Figure 5.1 One-Dimensional Linear Array.....	36
Figure 5.2 Two-Dimensional Orthogonal Array	36
Figure 5.3 Two-Dimensional Hexagonal Array	37
Figure 5.4 Binary Tree Architecture.....	38
Figure 5.5 Simple Multiply Array Based on The Carry Propagate Adder.....	40
Figure 5.6 Simple Multiply Array with Pipelining	41
Figure 5.7 Conventional Moment Generator Architecture	44
Figure 6.1 Pipelined Moment Generator Layout.....	49
Figure 6.2 Linear Feedback Register Used for Signature Analysis.....	54
Figure 6.3 LSSD Built-In Test Block Diagram.....	54
Figure 6.4 Multiplexing Path Built-In Logic	55
Figure 6.5 Prototype Moment Generator Functional Layout with Scan Paths.....	57
Figure 6.6 Half Adder	58
Figure 6.7 Full Adder.....	59
Figure 6.8 Precharged Carry Line.....	61

FIGURE	PAGE
Figure 6.9 Static Carry Chain.....	62
Figure 6.10 Manchester Carry Conditions Adder Cell.....	64
Figure 6.11 Adder Cell Functional Diagram	66
Figure 6.12 Static Register Cell Designed with Scan Path Capability.....	68
Figure 6.13 Single-Phase and Two-Phase Clocking Schemes.....	71
Figure 7.1 COSMOS Test Method Block Diagram.....	76

CHAPTER 1

INTRODUCTION

By combining the performance of custom very large scale integrated (VLSI) special-purpose processors with pattern recognition algorithms, such as moment invariants used in image processing, high-performance computational engines that can process digitized images in real time become possible. Custom architectures designed using VLSI are becoming more prevalent in making pattern recognition algorithms that execute in real time.

1.1 MOTIVATION

VLSI circuit technology provides the capability to put 10 to 100,000 transistors on a single silicon circuit referred to as an application-specific integrated circuit (ASIC). This allows many algorithms once limited to designs implemented primarily by special-purpose computers or dedicated processor boards to be incorporated into a single chip. With internal switching speeds of the CMOS transistor reaching 100 to 120 MHz, the algorithms incorporated into VLSI ASICs also achieve a substantial performance improvement over conventional von Neumann architectural implementations and even custom implementations based on discrete devices. In addition to the performance improvements offered by VLSI, the design can be customized to fit the specific needs of one application and integrated on a single chip that can be completely simulated by the VLSI CAD tools in less time than required to build a prototype with discrete TTL chips.

With the increased use of pattern recognition for image processing applications, there is a growing need for special-purpose, high-performance processors that can process these algorithms in real time. Moment invariants is a computational intensive algorithm that offers the capability for data compression and unique representations of digitized images. This thesis presents the design and implementation of a VLSI coprocessor that is used to compute moments used in the moment invariant algorithms developed by Hu and Maitra.

1.2 CHAPTER CONTENTS

Chapter 2 discusses the background of VLSI use in pattern recognition. Chapter 3 introduces the moment invariant algorithms used for pattern recognition that are implemented in VLSI in this design. Chapter 4 discusses the mechanics of CMOS transistors and circuits and presents a comparison of CMOS to nMOS technology. Chapter 5 discusses the architectural design of the ASIC and its internal functions. Chapter 6 presents the design of the special-purpose ASIC to be used in the moment invariant algorithm. Chapter 7 discusses the tools used to do the simulation and testing of the VLSI ASIC. Chapter 8 is the conclusion, which discusses the results of the design in terms of lessons learned and improvements that could be implemented in future designs.

1.3 APPENDIX CONTENTS

Appendix A contains the layout of the Manchester carry chain adder cell. Appendix B contains the layout of the static memory cell used as a scan path and pipe between the multiplier and adder functions and as the accumulator register. Appendix C contains a COSMOS logic simulation control file. Appendix D contains a COSMOS fault simulation control file. Appendix E contains a Pspice simulation control file. Appendix F contains a Pspice timing plots.

1.4 DEFINITION OF TERMS

The following is a list of the terms and abbreviations used in the thesis to describe the design and implementation of the pattern recognition system:

ANAMOS

An executable program used by COSMOS which generates boolean function descriptions from the network description provided by "sim2ntk".

ASIC

Acronym for Application Specific Integrated Circuit.

BILBO

Scan path test method which combines IBM's LSSD method with signature analysis to produce a built-in test function that can be reset to a known state, scan in and scan out serial test vectors, and generate pseudo-random test patterns (signature analysis).

CIF

Acronym for Caltect Intermediate Format which is used as a protocol for file storage format.

CMOS

Acronym for Complimentary Metal Oxide Semiconductor.

COSMOS

Acronym for Compiled Simulator for Metal Oxide Semiconductor.

DARPA

Acronym for Defense Advanced Research Projects Agency.

DRC

Acronym for Design Rule Checker which is used by the VLSI color graphics editor (magic) to interactively display geometric and semiconductor layout errors to the VLSI designer.

ESIM

Event Simulator, an executable program used to perform logical simulations on VLSI designs.

Ext2sim

Executable program which converts the circuit description extracted by the VLSI graphical editor to a simulation file that can be used to simulate the design.

EXOR

Acronym for the boolean logical operator for exclusively-or.

Fanout

A quantitative measurement of the number of loads on a driver circuit.

Farad

The unit of capacitance of a capacitor with a voltage potential of one volt and a stored charge of one coulomb.

FET

Acronym for Field Effect Transistor

gm

Transconductance, measures the ratio of the change in drain current to the change in gate to source voltage.

Hz

Hertz.

Ids

Drain -to-source current.

Kp

Process dependent part of the gain factor.

Latch-up

Description of the phenomenon caused by the formation of parasitic transistors that form between the MOSFETs in a CMOS design.

MOPS

Million operations per second.

pFET

P-channel field effect transistor.

pmcc

Acronym for the register cells used to make pipe functions.

LGCC

Executable program used by the COSMOS simulator to construct a "C" source code program from the boolean function description generated by ANAMOS.

LSSD

Level Sensitive Scan Design approach which allows a test vector to be written into a serial shift register and then applied in parallel to a combinational logic circuit.

mA

Milliampere.

mcc

Acronym for manchester carry conditions adder cell.

Metal2

Second level metal used for routing interconnections on a VLSI design.

Metastable

Undefined region of operation that can occur in improperly designed storage cells.

MOS

Metal Oxide Semiconductor.

MOSFET

Metal Oxide Semiconductor Field Effect Transistor .

MOSIS

An acronym for MOS Implementation System. MOSIS provides low cost fabrication services to government contractors, agencies and universities. MOSIS is sponsored by DARPA and NSF.

NAND

Boolean logical operator for complimented AND.

nFET

N-channel FET.

nMOS

N-channel MOSFET.

NSF

National Science Foundation.

nwell

N-type well used to provide a n-type substrate for pFETs on a p-type wafer.

Pixel

Picture element, used describe intensity at a point in a digital image. A pixel is typically stored in one byte for 256 grey scale images.

pFET

P-channel FET.

Pipelined

Architectural design method used to operate multiple functions in parallel to derive a speed improvement in the system.

PLA

Programmable Logic Array.

pMOS

P-channel MOSFET.

Pspice

Executable program that is used to do timing and waveform analysis of a VLSI design.

pwell

P-type substrate used for the formation of n-channel transistors on a n-type wafer.

rds

Drain-to-source resistance.

Sim

The extension of the simulation file generated by the 'ext2sim' program.

Sim2ntk

Executable program used by COSMOS to generate a network list from the simulation file.

Vdd

Power supply voltage, typically 5 volts.

Vds

Drain-to-source voltage.

Vgs

Gate-to-source voltage.

VLSI

Very Large Scale Integration.

CHAPTER 2

BACKGROUND

One difficulty in making pattern recognition techniques practical for use in real-time systems is the high amount of computational processing necessary to extract the elements that uniquely define the images under study. It becomes very important to evaluate special-purpose, high-performance computers for applicability to pattern recognition algorithms. VLSI technology provides the digital design engineer the ability to design custom special-purpose, high-performance computation engines. To fully utilize the high-switching speeds attainable by VLSI technology, the architecture of the functions implemented in VLSI need to be pipelined to allow a high degree of parallelism in the design.

2.1 THE NEED FOR PERFORMANCE IMPROVEMENT

To test the accuracy of the algorithm's results and performance, the moment invariant algorithm was tested on a Sun 386i Workstation^[15]. The Sun was programmed to examine 512X512 images with each picture element (pixel) having 256 possible grey scales. As equations (1) and (2) show, the computer had to process 256 Kbytes of data per image to determine the resulting moment invariant constants that would represent the image. To satisfy the requirements for the pattern recognition algorithms, seven second- and third-order moments would have to be evaluated. In addition, the centroids of the image would have to be found to determine the centralized moments. The following equations show that the amount of raw data per image is 256 Kbytes.

$$256 \text{ grey scales/pixel } (8 \text{ bits/pixel} = 1 \text{ byte/pixel}) \cdot \quad (1)$$

$$512 \text{ pixels} \times 512 \text{ pixels} \times 1 \text{ byte/pixel} = 256 \text{ Kbytes} \cdot \quad (2)$$

The Sun was able to calculate the invariant moment for one image in 20 seconds. The desired design goal for implementing the moment invariant algorithm is to process 256 Kbytes of data at frame rates. To process the images at frame rates, the processor is required to process 30 frames per second. Equation (3) shows that to process 30 frames that contain 256 Kbytes each in 1 second requires that the raw data be processed at 7.68 Mbytes per second.

$$30 \text{ frames/second} \times 256 \text{ Kbytes/frame} = 7.68 \text{ Mbytes/second} . \quad (3)$$

To speed up the algorithm, it was determined that the moments calculation needed to be performed at a much higher rate. VLSI ASIC designs contain CMOS transistors that can switch at rates as high as 100 to 120 MHz using current technology and fabrication techniques. Input of the data into and out of the chip, however, is still limited to TTL rates of 10 to 20MHz. Because of the high switching speeds of CMOS inverters and the use of special-purpose architectures that incorporate pipelined and parallel processing, CMOS VLSI designs can be used to implement the moment calculation at speeds necessary to meet the design goals.

2.2 JUSTIFICATION FOR THE USE OF VLSI

Two primary requirements must be met before a design can be considered for VLSI ASIC implementation. These requirements are modularity and simplicity. On a systems level, the implementation of all the moments required to satisfy the six moment invariants equations represents a great deal of repetition. The moment calculation requires the use of multipliers, adders, and storage registers. Each of these can be broken down into simple cells that then can be replicated quickly into arrays to create the necessary functions. These elements can be configured into a pipe to allow parallel processing of the data.

CHAPTER 3

MOMENT INVARIANTS

This chapter presents the concept of moments and describes how moments are related to digital imaging systems. The equations that are used to make the image invariant to changes in orientation, scaling, contrast and illumination are presented. The basis for the selection of the part of the algorithm to be implemented in the VLSI design is presented.

3.1 INTRODUCTION

Moments are used to describe the distribution of a physical system with finite mass relative to a reference point in the system. The relationship is described by summing the products of each element in the system with its euclidean distance from the reference point. In physical systems, moments are used to describe the moment of inertia of a physical mass relative to one of the axes in the system. Inertia is the resistance of the rotational system to movement in the same way that the mass is the resistance of an object in a linear system to movement. Moments also can be used to describe or represent digitized images. The mass of the elements in the physical system is replaced by the light intensity of a point in the image captured by the camera and digitization equipment. In the digitized image, the distance to the reference point must now be measured in two- rather than three-dimensional space. The use of moments to represent images allows a representation of the image to be stored in several bytes rather than the 256 Kbytes required to store a 256 grey-scale 512X512 pixel image. In addition to the computer resources saved by data compression, the moment of the image allows the representation to be compared quickly with other images for pattern recognition applications[6].

Obvious problems develop when the moments are used to describe images uniquely. The physical system exists in three dimensions while the image is contained in two dimensions. The distance to the center of gravity of the physical system remains the same even under changes in orientation. This is not true for images. The image of the physical system changes because of changes in the orientation of the physical system with respect to the camera. The resultant image has distance measurements reduced to one plane, which

reduces the ability to accurately include depth in the distance measurements. Changes in orientation and proximity to the camera system hide the true moment information about the physical system. In addition, the mass of each element in the physical system has been replaced by the intensity of a pixel. The intensity of the pixel is not an inherent property of the image, as is the mass of the physical system. The intensity varies with the background lighting and the location of light sources used to illuminate the object. As a result, changes in orientation, scaling, and lighting have direct effects on the digitized image and can prevent moments from being used to generate unique representations for use in pattern recognition.

3.2 MOMENT PROPERTIES OF IMAGES

This section introduces the equations used to develop the moment invariant algorithms. First a definition about moments must be established and given about moments must be stated. The definition of a second-order ordinary moment is given in equation (4).

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy , \quad (4)$$

where $f(x,y)$ is the density distribution (or intensity) that must be piecewise continuous and has nonzero values in a finite part of the xy plane. The uniqueness theorem^[6] states that if $f(x,y)$ is piecewise continuous and has nonzero values in a finite part of the xy plane, then moments of all orders exist and the moment sequence (m_{pq}) is uniquely determined by $f(x,y)$.

Uniqueness theorem: The double moment sequence (m_{pq}) is uniquely determined by $f(x,y)$; conversely, $f(x,y)$ is uniquely determined by (m_{pq}) .

The digitized image that defines $f(x,y)$ is composed of a finite number of elements or pixels that would be zero only if the image was captured in total darkness. As a result, orders of all the moments of the digital images exist. Once the moments have been determined to exist, they must be made invariant to changes.

3.3 MOMENT INVARIANT ALGORITHMS

Moment Invariants is a technique that attempts to make moments invariant to changes in translation, rotation, scaling, illumination, and contrast. Two moment invariant algorithms are

described in this chapter. The two algorithms were developed by Hu^[7] and Maitra^[10]. Hu developed an algorithm that takes all second- and third-order moments of the image and produces seven constants that are invariant to translation, rotation, and scaling. Maitra's algorithm offers an improvement to Hu's algorithm by taking the seven constants and in putting them in to six more equations, which generates six constants that are also invariant to changes in contrast and illumination.

Moments can be made invariant to translation by considering the moment with respect to the centroid of the image. Central moments take the centroid into consideration and are invariant to translation changes by definition. The central moment of order $(p+q)$ for a piecewise continuous function $f(x,y)$ is defined in equation (5).

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q f(x,y) dx dy . \quad (5)$$

The centroids $\bar{x}$ and $\bar{y}$ are defined as

$$\bar{x} = m_{10}/m_{00} \text{ and } \bar{y} = m_{01}/m_{00} . \quad (6)$$

For a digital image, the double integration is expressed as double summation since the pixels are discrete points instead of piecewise continuous surfaces. The discrete central moment of order $(p+q)$ is defined as

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y) dx dy . \quad (7)$$

All 7 moments of orders 2 and 3 are needed for Hu's algorithm. All combinations of the discrete moments of orders 1, 2, and 3 are shown in equations (8) through (16).

$$\mu_{10} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^0 f(x,y) dx dy , \quad (8)$$

$$\mu_{01} = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^1 f(x,y) dx dy , \quad (9)$$

$$\mu_{11} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^1 f(x,y) dx dy , \quad (10)$$

$$\mu_{20} = \sum_x \sum_y (x-\bar{x})^2 (y-\bar{y})^0 f(x,y) dx dy , \quad (11)$$

$$\mu_{02} = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^2 f(x,y) dx dy , \quad (12)$$

$$\mu_{30} = \sum_x \sum_y (x-\bar{x})^3 (y-\bar{y})^0 f(x,y) dx dy , \quad (13)$$

$$\mu_{12} = \sum_x \sum_y (x-\bar{x})^1 (y-\bar{y})^2 f(x,y) dx dy , \quad (14)$$

$$\mu_{21} = \sum_x \sum_y (x-\bar{x})^2 (y-\bar{y})^1 f(x,y) dx dy , \quad (15)$$

$$\mu_{03} = \sum_x \sum_y (x-\bar{x})^0 (y-\bar{y})^3 f(x,y) dx dy . \quad (16)$$

Even though Hu's algorithm does not require directly the central moments of orders 0 and 1, the determination of the centroid requires the ordinary moments of orders 0 and 1. The 0 order ordinary moment also is used to normalize the centralized moments. As seen in equations (17) through (26) below the central moments of order 1 are zero and of little use to the algorithm. The next 7 moments and the moment of order 0 are used by Hu's algorithm. Hu takes advantage of the properties of moments to produce his results. The first property is that moments are linear operators; therefore, by the superposition theorem the moment of the image is the sum of the image's pixel moments. Secondly, a moment in one coordinate system can be mapped to another coordinate system by other linear operators. These are the two properties on which moment invariants is built.

Ordinary moments are determined with respect to the origin of the coordinate system. Ordinary moments can be mapped by linear operators into a new coordinate system where the moments in the new coordinate system describe centralized moments. The central moments can be expressed in terms of ordinary moments as shown in equations (17) through (26).

$$\mu_{00} = m_{00} , \quad (17)$$

$$\mu_{10} = 0 , \quad (18)$$

$$\mu_{01} = 0 , \quad (19)$$

$$\mu_{20} = m_{20} - \bar{x}m_{10} , \quad (20)$$

$$\mu_{02} = m_{02} - \bar{y}m_{01} , \quad (21)$$

$$\mu_{11} = m_{11} - \bar{y}m_{10} , \quad (22)$$

$$\mu_{30} = m_{30} - 3\bar{x} m_{20} + 2\bar{x}^2 m_{10} , \quad (23)$$

$$\mu_{03} = m_{03} - 3\bar{y} m_{02} + 2\bar{y}^2 m_{01} , \quad (24)$$

$$\mu_{12} = m_{12} - 2\bar{y} m_{11} - \bar{x}m_{02} + 2\bar{y}^2 m_{10} , \quad (25)$$

$$\mu_{21} = m_{21} - 2\bar{x} m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01} . \quad (26)$$

The central moments, as previously mentioned, are invariant to changes in translation. This demonstrates how a set of moments can be mapped into a new coordinate system by linear operators causing the resulting moments to be invariant to changes in translation and orientation in the plane of the image. Another linear operator can be used to normalize the moments thereby making the resultant moments also invariant to changes in parallel projection or scaling. The normalized central moments, denoted by η_{pq} , are defined as:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} ; \quad (27)$$

$$\gamma = \frac{p+q}{2} + 1 ; \quad (28)$$

for

$$p + q = 2, 3, \dots . \quad (29)$$

As a result, the moment invariants are made invariant to scale changes by the normalization of the central moments.

3.3.1 HU ALGORITHM

Hu's moment invariant algorithm uses the normalized central moments of orders 2 and 3 as input to seven algebraic equations [(30) through (36)] as defined:

$$\phi_1 = \eta_{20} + \eta_{02}; \quad (30)$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2; \quad (31)$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2; \quad (32)$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2; \quad (33)$$

$$\begin{aligned} \phi_5 = & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]; \end{aligned} \quad (34)$$

$$\begin{aligned} \phi_6 = & (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ & + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}); \end{aligned} \quad (35)$$

and

$$\begin{aligned} \phi_7 = & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. \end{aligned} \quad (36)$$

This method produces seven constants for use in pattern recognition that are invariant to changes in orientation, rotation, position, size, and parallel projection (focus).

3.3.2 MAITRA ALGORITHM

Equations (37) and (38) represent the changes produced when the image is translated, rotated, scaled, and subjected to a change in contrast. With the moment invariants described thus far, the results for the two images $f_1(x,y)$ and $f_2(x,y)$ should be the same under the following conditions:

$$f_1(x, y) = kf_2(x', y'), \quad (37)$$

and

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = c \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a \\ b \end{bmatrix}. \quad (38)$$

The rotation is described by the angle theta, (a,b) describes the translation, c produces a change in scale, and k produces the change in illumination and contrast. The second equation is used to map points in the first coordinate system into a new coordinate

system in which the origin has been translated by (a,b), rotated by theta, and scaled such that the points are farther apart for $c > 1$ and closer together for $c < 1$.

Without the normalization procedure shown previously, the relationship between the invariant moments for the two images can be shown by equations (39) through (45).

$$\phi_1(1) = \frac{k}{c^4} \phi_2(1); \quad (39)$$

$$\phi_1(2) = \frac{k^2}{c^8} \phi_2(2); \quad (40)$$

$$\phi_1(3) = \frac{k^2}{c^{10}} \phi_2(3); \quad (41)$$

$$\phi_1(4) = \frac{k^2}{c^{10}} \phi_2(4); \quad (42)$$

$$\phi_1(5) = \frac{k^4}{c^{20}} \phi_2(5); \quad (43)$$

$$\phi_1(6) = \frac{k^3}{c^{14}} \phi_2(6); \quad (44)$$

$$\phi_1(7) = \frac{k^4}{c^{20}} \phi_2(7). \quad (45)$$

If the moment invariants were figured using the normalized central moments the variations in the results caused by the scaling term c would be eliminated. This leaves the variation due to the contrast and illumination changes made by the factor k . Maitra proposes using the following six equations [(46) through (51)] that have eliminated the k term and therefore make the resulting six equations invariant to changes in illumination and contrast:

$$\beta(1) = \frac{\sqrt{\phi(2)}}{\phi(1)}; \quad (46)$$

$$\beta(2) = \frac{\phi(3)\mu_{00}}{\phi(2)\phi(1)}; \quad (47)$$

$$\beta(3) = \frac{\phi(4)}{\phi(3)}; \quad (48)$$

$$\beta(4) = \frac{\sqrt{\phi(5)}}{\phi(4)}; \quad (49)$$

$$\beta(5) = \frac{\phi(6)}{\phi(4)\phi(1)}; \quad (50)$$

and

$$\beta(6) = \frac{\phi(7)}{\phi(5)}. \quad (51)$$

3.4 EQUATION STATISTICS

To develop an understanding for where the maximum amount of computational effort is required in the moment invariants algorithm, the equations have been broken down into the number of basic algebraic operations required for the computation. Tables 3.1, 3.2, and 3.3 give the statistics for each equation. Table 3.1 presents the statistics for the ordinary moment equations. Table 3.2 presents the statistics for the algebraic operations required by the Hu algorithm. Table 3.3 presents the statistics for the equations used by the Maitra algorithm. As is evident by the statistics given in Tables 3.1 through 3.3, the moments take up a great deal of the computations necessary to implement the moment invariants algorithm. Each of the nine moments required, either for the centroid determination or for the second- or third-order moments used in the Hu and Maitra algorithms, requires one addition and one to three multiplications per pixel in the image. With each image having 256kB or 262,144 pixels, it is obvious that the moments will take a high percentage of the computation required to develop moment invariant descriptors. The algorithms developed by Hu and Maitra are composed of algebraic equations that are can typically be solved in less than ten steps each. To allow the moment invariants algorithm to be computed in real time, the pixels must be processed at a rate of 7.68 Mpixels per second. The high number of multiplications and additions required to calculate the moments makes the moment generator the choice for implementation into VLSI. The implementation of the moment equation into a VLSI ASIC allows for the use of multiple ASICs for deriving one set of moment invariant coefficients for a given image. If it became necessary to operate at speeds higher than frame rates, more systems of moment chips could be used in parallel to accomplish the higher rates.

Table 3.1. Moment Calculations.

Equation	Multiplication	Division	Addition	Subtraction	Powers
m00	0	0	262,144	0	0
m01	262,144	0	262,144	0	0
m10	262,144	0	262,144	0	0
m11	524,288	0	262,144	0	0
m02	524,288	0	262,144	0	0
m20	524,288	0	262,144	0	0
m12	786,432	0	262,144	0	0
m21	786,432	0	262,144	0	0
m03	786,432	0	262,144	0	0
m30	786,432	0	262,144	0	0

Table 3.2. Hu Algorithm

Equation	Multiplication	Division	Addition	Subtraction	Powers
phi1	0	0	1	0	0
phi2	3	0	1	1	0
phi3	4	0	1	2	0
phi4	2	0	3	0	0
phi5	12	0	7	4	0
phi6	6	0	5	2	0
phi7	12	0	6	5	0

Table 3.3. Maltra Algorithm

Equation	Multiplication	Division	Addition	Subtraction	Powers
beta1	0	1	0	0	1
beta2	2	1	0	0	0
beta3	0	1	0	0	0
beta4	0	1	0	0	1
beta5	1	1	0	0	1
beta6	0	1	0	0	0

3.5 SUMMARY

Moment invariant algorithms can be used in pattern recognition algorithms, used in digital-image processing applications, to compress and represent images. Moments, which are normally used to uniquely describe a physical three-dimensional system based on inherent properties such as the mass of the elements in the system, can be used to describe digital images that represent the physical system captured in a two-dimensional image. Moment invariant algorithms developed by Hu and Maitra can make the resultant-image descriptors invariant to changes in orientation, scaling, contrast, and illumination. The high amount of computations required to produce the descriptors is due to the requirement of nine moments calculations, ranging from order zero to order three. The time required to determine the invariant coefficients can be reduced by off loading the computation of the moments from the computer to custom VLSI ASICs. With the use of the VLSI design the moment invariant algorithm can potentially be determined in real time.

CHAPTER 4

MOSFET MECHANICS

The primary building blocks used to implement the functions required to determine the necessary moments in the pattern recognition algorithm are composed of MOSFET switches. The MOSFET switches control the flow of information in silicon by charging and discharging nodes in each circuit. This chapter discusses the first-order approximations used to describe the operations of these switches. This chapter also discusses the inverter and the transmission gate that are the basic building blocks with which all other logical functions are built. The device mechanics that govern the operation of these two primary elements are reviewed. The curves that describe the operation of the CMOS inverter are graphically solved from the families of curves that characterize each device. Parameters used to model the devices for simulations also are presented, as well as comparison between the nMOS and CMOS technologies, and used for the basis for choosing CMOS as the device technology for the VLSI design.

4.1 COMPARISON OF CMOS TO NMOS

The CMOS circuits are ratioless, meaning that the output voltage levels are not a function of the geometric ratio of the pullup to pull-down devices. The nMOS inverter is very dependent upon device size and uses an inverter with a pullup-to-pulldown ratio of 8 to restore weak signals (possibly degraded by a series of pass transistors) and a ratio of 4 to maintain feasible levels of noise margins, logic threshold, and device gain in the standard inverter. This is another advantage of CMOS over nMOS; CMOS logic is fully restored and does not require the addition of special amplifiers to restore signal strengths. Because the CMOS inverter fully restores the signal level, the 1's and 0's are all strong signals. This results in a larger voltage swing and, therefore, a higher noise immunity for CMOS than nMOS.

Table 4.1 provides a summarizes the differences between the CMOS and nMOS technologies.

Table 4.1. Comparison of CMOS to nMOS.

CMOS	nMOS
Logic levels are fully restored .	Logic levels are not fully restored, resulting in lower noise margins.
Transition times can be made symmetric.	nMOS is faster than CMOS.
Transmission gates can drive the inputs of other transmission gates.	Pass transistors cannot drive the gate of other pass transistors.
Static power usage is almost zero .	nMOS requires high amounts of static power.
Power supply voltage can vary from 1.5 to 15 volts.	A fixed supply voltage is required.
2N devices are required for N inputs.	N = 1 devices are required for N inputs.
Typical load-to-driver device ratio is 1:1 or 2:1.	Load-to-enhancement-driver ratio is typically 4:1 to optimize logic "0" output while minimizing current (8:1 required to restore nFET degraded output signals).
CMOS encourages regular layout styles.	Depletion load and different driver transistor sizes inhibit layout regularity.
CMOS scales well.	nMOS design rules require modifications to scale the devices significantly (ratio rules used to design nMOS are size dependent).

The CMOS fabrication process has a tendency to produce parasitic circuits that provide a low impedance path from V_{dd} to ground when triggered. This phenomenon is known as "latch-up." These parasitic circuits are composed of resistors and transistors that form in and between the actual transistors, the wells, and the substrate. The latch-up of the parasitic circuit typically is triggered by a high-transient current flow in the chip. Design rules

exist that aid VLSI designers in producing designs that are latch-up free. These design rules include placing substrate contacts (which connects the well to the appropriate level) as close as possible to the source of the device.

The CMOS technology was chosen for the design of the moment generator due to its low power consumption and ease of use in design. The use of the ratioless CMOS devices gives strong logical 1s and 0s, making this technology much less complicated to design custom cells than the nMOS technology requiring constant numerical analysis of transistor sizes each time a circuit modification is performed.

4.2 MOSFET DEVICE DESCRIPTION

The transfer of information through a MOSFET network is performed by the transfer of charge from one node to another. This transfer is controlled by varying the input voltage to the gate of a MOSFET device, which in turn controls the flow of current through the device. Information flow is performed when one device routes a stored charge from one node to another. The highest available source of capacitance for storing the charge in a node is the gate of the MOSFET transistor. Equations (52) through (54) describes the gate capacitance along with typical values for its variables.[18]

$$C_g = \frac{\epsilon A}{t_{ox}} \quad (\text{gate capacitance}) ; \quad (52)$$

$$A = WL = 3\mu\text{m} \times 2\mu\text{m} = 6\mu\text{m}^2 \quad (\text{typical gate area}) ; \quad (53)$$

and

$$t_{ox} \approx 500 \text{ angstroms} \quad (\text{typical silicon dioxide thickness}) . \quad (54)$$

The speed with which a charge is moved from one node to the next is the transit time of the device. To move the charge from one node to the next, the charge must pass through the diffusion channel of the controlling device. The transit time is a function of the diffusion channel length divided by the average drift velocity of the majority carriers through the channel. The average drift velocity is directly proportional to the mobility of the majority carriers and the electric field intensity across the gate region from drain to source as shown in equation

(56). The electric field intensity is directly proportional to the drain-to-source voltage (V_{ds}) divided by the length. The transit time, t , is defined in the equations (55) through (57).

$$t = \frac{L}{\text{average drift velocity } (v)} \quad (\text{transit time}) ; \quad (55)$$

$$v = \mu E \quad (\text{average drift velocity}) ; \quad (56)$$

and

$$E = \frac{V_{ds}}{L} \quad (\text{electric field intensity}) . \quad (57)$$

Given that the change in charge at the gate node is directly proportional to the gate capacitance times the change in voltage at the gate, equations (58) through (61) describes the charge which can flow into or out of the gate node. The current, I_{ds} , through the device which is sourcing or sinking the charge to the gate node, is approximated as change in the charge divided by the transit time as shown in equations (59) through (61). Drain current derivation:

$$\Delta Q = C_g(V_{gs} - V_{th}) = \frac{\epsilon W L}{t_{ox}}(V_{gs} - V_{th}) \quad [\text{charge in transit } (\partial Q = C_g \partial V_{ds})] ; \quad (58)$$

$$I_{ds} = \frac{\text{charge in transit}}{\text{transit time}} = \frac{\Delta Q}{t} \quad [\text{Drain Current } (I_{ds} = \partial Q / \partial t)] ; \quad (59)$$

$$I_{ds} = \frac{\epsilon W L}{t_{ox}}(V_{gs} - V_{th}) / (L^2 / \mu V_{ds}) = \frac{\mu \epsilon}{t_{ox}} \left(\frac{W}{L} \right) (V_{gs} - V_{th}) V_{ds} ; \quad (60)$$

and

$$I_{ds} = \beta (V_{gs} - V_{th}) V_{ds} . \quad (61)$$

The drain-to-source current typically flows through a MOSFET device to charge or discharge a node that is serving as the gate of another MOSFET device. This current flow is proportional to the transistor gain factor, beta. Equation (62) defines the gain factor. The parameters for the gain factor equation have been grouped to separate the process dependent variables from the geometry dependent variables.

$$\beta = \frac{\mu \epsilon}{t_{ox}} \left(\frac{W}{L} \right) . \quad (62)$$

The process dependent variables are partially determined at the fabrication facility by the type of processes chosen to implement the silicon chip, such as pwell technology or nwell technology, and by the selection of vapor deposition or ion implantation for creating the MOSFET devices. These parameters are dependent upon doping levels, the material chosen for the gate oxide, and how thickly the gate oxide is deposited on the substrate. The geometry dependent variables are determined by the designer. Typically, the smallest allowable size for a CMOS device is 3 lambda wide (W) by 2 lambda long (L). (Lambda is a scalable unit used to represent relative device sizes. Current fabrication processes are making VLSI ASICs with lambda equal to 1 micrometer.) The geometry variables are also process dependent, due to less than 1 lambda or 1 micrometer accuracy of the layout mechanisms.

Two types of devices are used in CMOS, p-channel MOSFETs or pFETs and N-channel MOSFETs or nFETs. Both devices can be characterized by three regions of operation: cutoff, linear, and saturation regions described in equations (63) through (65). The device is in the cut off region where the voltage at the gate is not large enough to produce the inversion layer, which provides the channel for majority carriers to flow through from the source to the drain. Once the gate voltage exceeds the threshold voltage, the channel is established between the drain and source, an inversion layer is created, and current flow is linear with respect to the gate voltage whenever the V_{ds} is less than the difference between the V_{gs} and the threshold voltage. When the V_{ds} exceeds this level, the inversion layer becomes pinched off near the drain. The pinched-off condition is the result of a reverse bias condition that develops between the drain and channel when the drain voltage exceeds the gate voltage. All current flow at this point is due to majority carriers falling out of the inversion layer and entering the depletion layer that separates the channel from the substrate. The high V_{ds} that caused the pinch-off condition in the first place now sweeps the majority carriers in the depletion region into the drain. The device is said to be in saturation at this point as further increases in the V_{ds} causes only a slight increase in the drain-to-source current flow.

$$I_{ds} = 0 \quad , \quad V_{gs} - V_t \leq 0 \quad (\text{cut-off region}) \quad , \quad (63)$$

$$I_{ds} = \beta \left((V_{gs} - V_t)V_{ds} - \frac{V_{ds}^2}{2} \right) \quad , \quad 0 < V_{ds} < V_{gs} - V_t \quad (\text{linear region}) \quad , \quad (64)$$

$$I_{ds} = \frac{\beta}{2} (V_{gs} - V_t)^2 \quad , \quad 0 < V_{gs} - V_t < V_{ds} \quad (\text{saturation region}) \quad . \quad (65)$$

4.3 CMOS INVERTER

A CMOS inverter is shown in Figure 4.1. The inverter is composed of two enhancement-mode CMOS transistors. The p-type device, or pFET, is used to pull the output node to V_{dd} (5 volts), or a logical one. The n-type device, or nFET, is used to pull the output node down to V_{ss} (0 volts), or logical low. In each case, the source of the device is tied to its source of majority carriers. For the pFET, the majority carriers are the holes, and the source is V_{dd} . The nFET's majority carriers are electrons, and its source is V_{ss} or ground. The operation of both types of devices may be described by the voltage magnitude at the gate and drain with respect to its source.

By definition, an enhancement-mode device enhances the concentration of the majority carrier into the channel to produce an inversion layer that permits conduction through the channel. For the nFET, the gate voltage should be positive with respect to the source (and greater than the threshold voltage) for conduction to begin. Likewise, the pFET gate

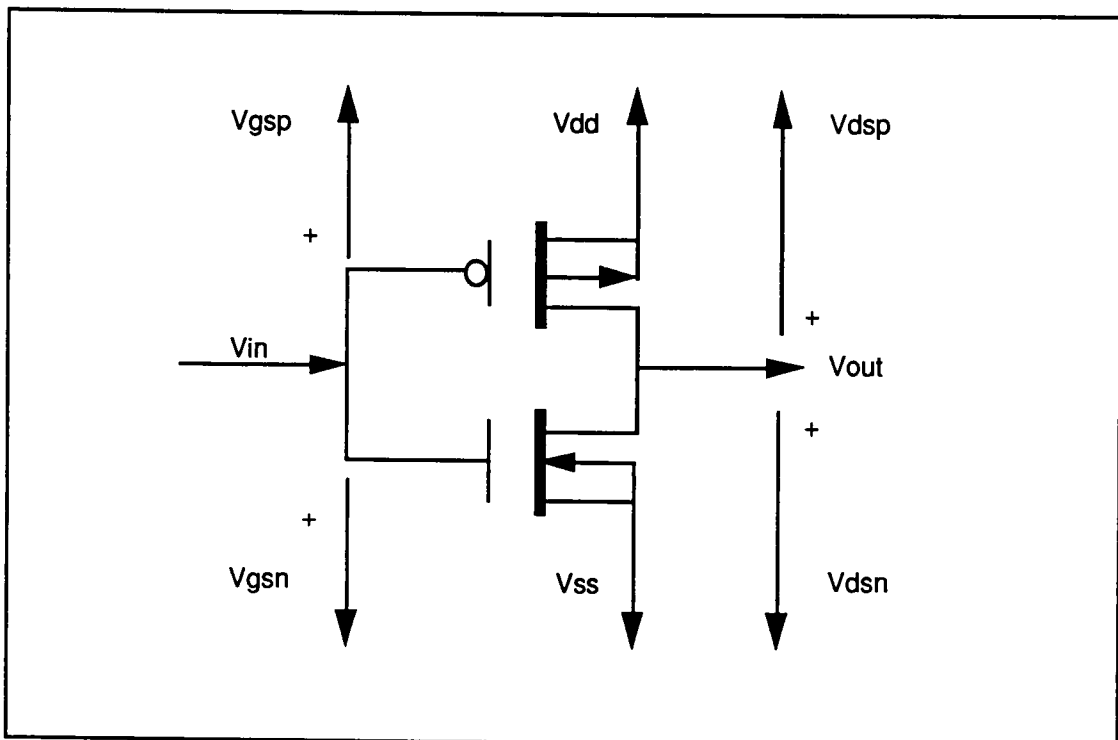


Figure 4.1. CMOS Inverter Schematic.

voltage must be negative with respect to the source to enhance the occurrence of holes in the inversion layer beneath the gate.

The drain current curves for the pFET and nFET are shown in Figures 4.2 and 4.3. It is possible to use the family of curves graphically that describes the operation of each type of CMOS-enhancement device to generate one curve, describing the operation of both devices when they are connected in series to form an inverter. The resultant curve describes the series operation of both devices together through the inverter's full range of operation.

Figure 4.2 illustrates typical values for a pFET. The graph shows negative values of drain current for negative values of drain and gate voltages. This may seem difficult to envision and totally incorrect when one realizes that the only source of voltage to the chip is positive 5 volts. However, the V_{DS} measurements are made with respect to a source that remains constant at 5 volts. The gate voltage and drain voltage vary from 0 to 5 volts. As a result, the only positive voltage that can be obtained from either of the two measurements is zero; all other measurements are negative.

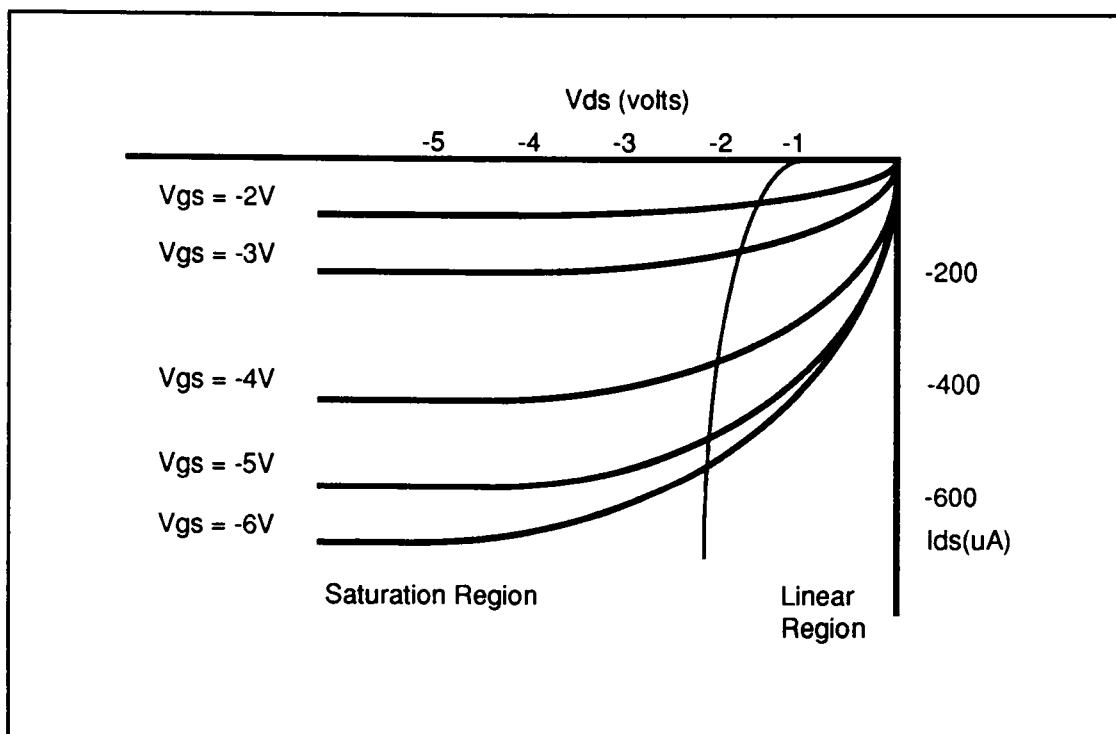


Figure 4.2. P-Channel MOSFET I_{ds} Curves.

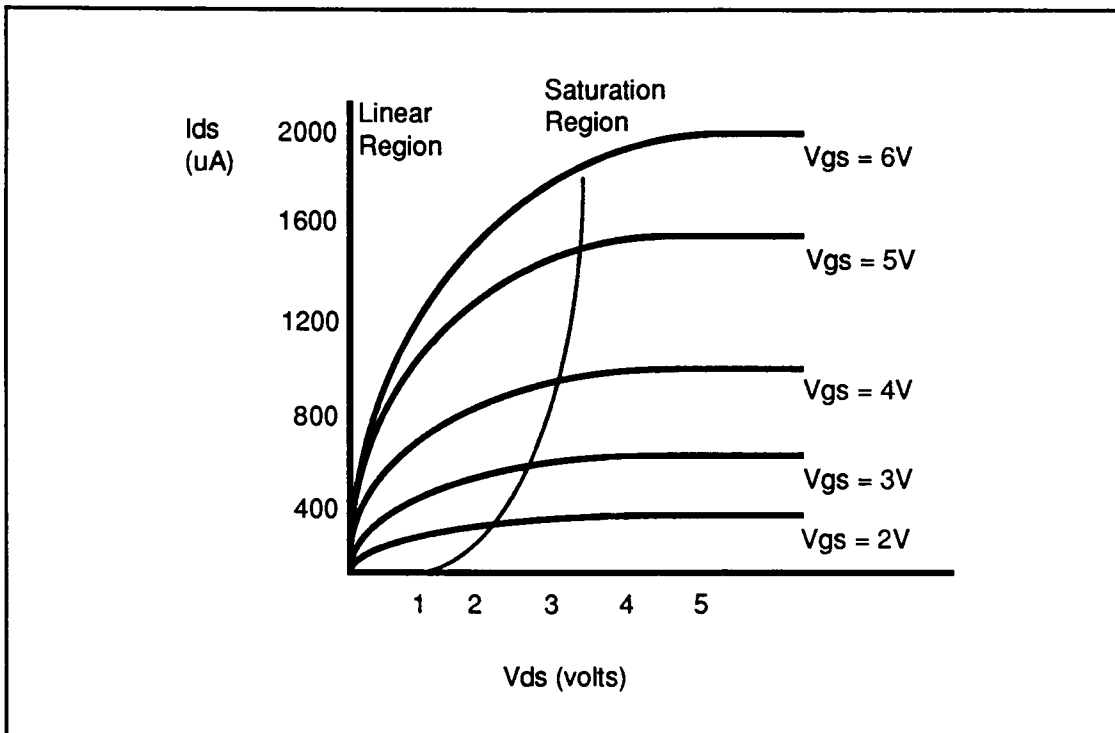


Figure 4.3. N-Channel MOSFET I_{ds} Curves.

Figure 4.3 shows the drain-to-source currents for the nFET. Because the source of the nFET is tied to ground, the resultant V_{ds} and the V_{gs} are always positive. Note that for similar size devices, the current levels for the same magnitude of voltage inputs are more than double in the nFET chart than those in the pFET chart. This is due to the majority carriers mobility in the nFET channel having approximately 2.5 times the mobility of the majority carriers than in the pFET channel.

The two graphs for the nFET and pFET drain-current curves can be placed on top of one another and graphically solved for a solution by noting that the current, through a series circuit, is always equal at any point as described by Kirchhoff's current law and that the gates of each device are tied together to form the inverter. Figure 4.4 illustrates an attempt to reach a graphical solution that describes the CMOS inverter operation, based on the known operation of the two enhancement FET's. Note that the FET's are of similar geometric size and therefore have asymmetric outputs.

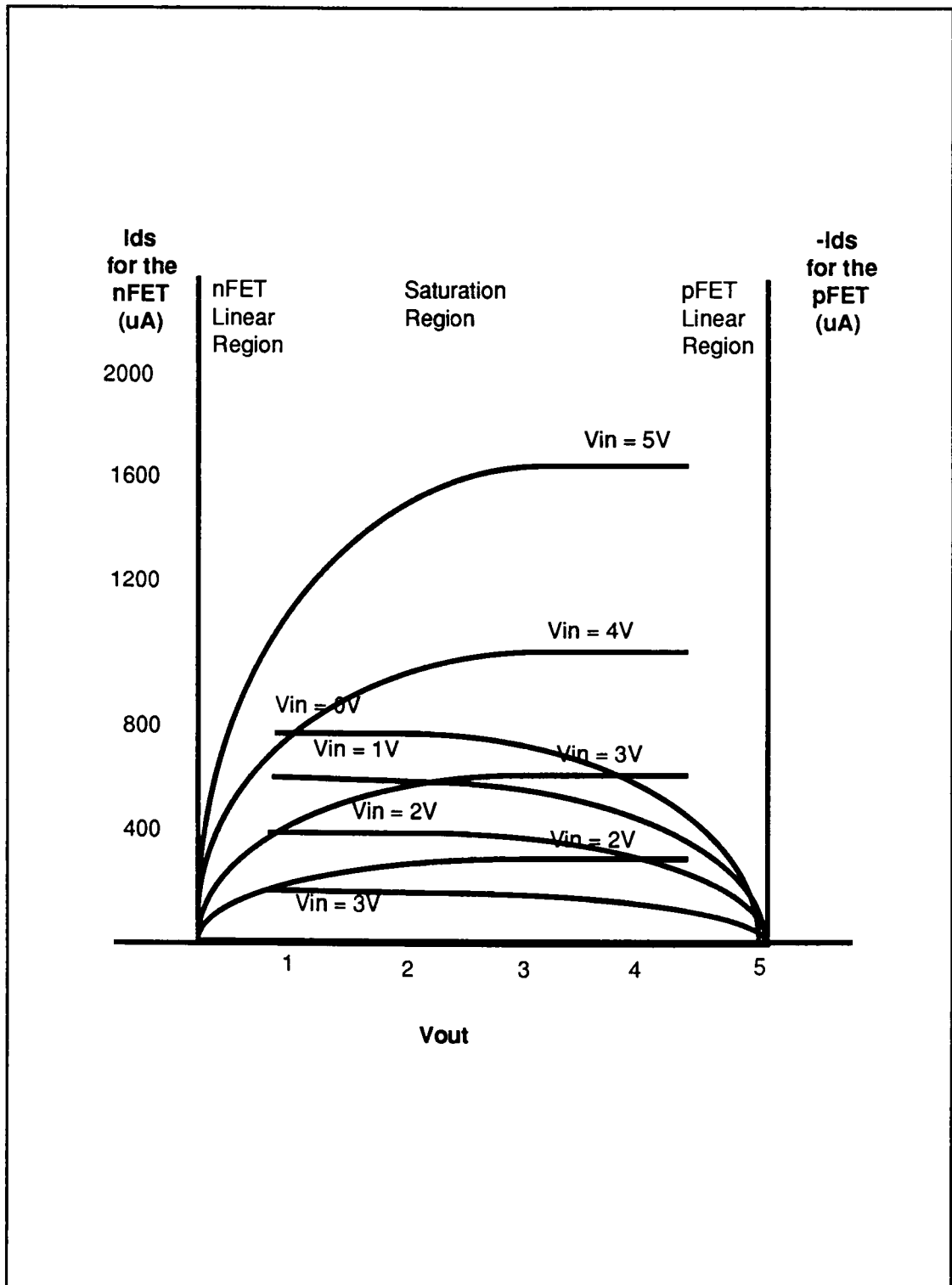


Figure 4.4. Graphical Solution for Inverter Current with Equal Size Devices.

By increasing the width of the pFET to two to three times the width of the nFET, the inverter theoretically can be designed to produce symmetric outputs for the same respective input voltages. Figure 4.5 depicts the graphs for ideal normalized pFET and nFET devices with equal values of drain-to-source current magnitude for respective values of V_{gs} .

The difference in the mobility of the pFET and nFET devices produces asymmetric voltage swings at the output of the CMOS inverter. In general, the pFET mobility is 2.5 times slower than the nFET, and this results in the rise time being slower than the fall time. The excepted procedure to reduce the asymmetry of the CMOS inverter transition times makes the pullup device three times wider than the pull-down device (given that the device lengths are equal). Unfortunately, there are other factors having input on the operations speed of the pFETs and nFETs and, therefore, resulting in the asymmetry of the output waveforms. Most of these variables are extremely process dependent (depending on the doping of p versus n, effective channel length differences, effective threshold differences, and the type of well technology chosen to implement the ASIC).

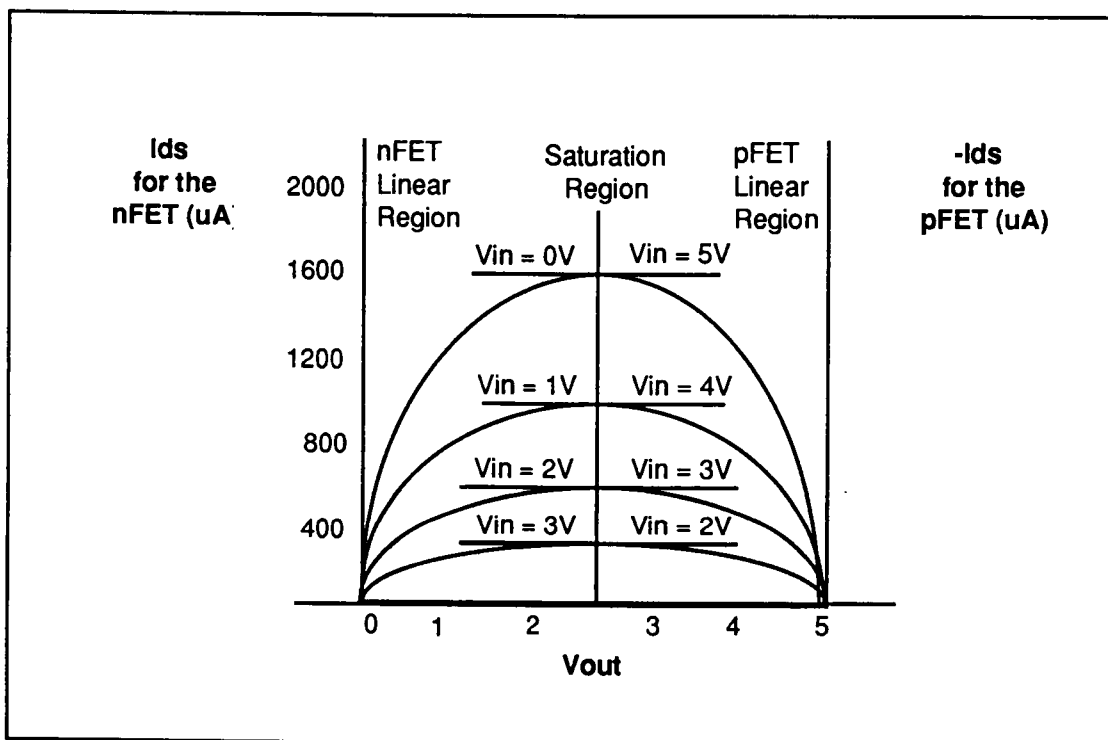


Figure 4.5. Graphical Solution for Inverter Current with Ideal Normalized Devices.

Figure 4.6 summarizes the joint operation of the two devices as a CMOS inverter without a load capacitance. Note that the CMOS inverter has five regions of operation. All five regions are shown with combinations of the three modes of operation used earlier to describe the nFET and pFET. In the first region, the input voltage is zero, and the nFET is in its cutoff region. The pFET is in the saturation region, but there is no current flow as the V_{DS} of the pFET is zero. As the input voltage increases, the devices move into region II, and the nFET moves out of its cutoff region and into its linear region while the pFET remains in saturation. Once the output voltage equals the input voltage, both devices are in the linear region (region III). In region IV, the nFET begins to move into saturation, which begins to quickly pull the output node down to ground potential. As the input node is driven to 5 volts, the pFET V_{GS} drops below its threshold and cuts off. The nFET is in full saturation at this point, but no current flows since the V_{DS} differential of the nFET is zero.

Since current flows only when the devices are in transition, the CMOS circuits have dynamic power consumption only. There is a negligible amount of leakage current that can flow in the device when the device is in the cutoff state. Power consumption for the CMOS circuits is a function of their switching frequency. This is in contrast to the nMOS inverter that has the pullup device on constantly, resulting in a substantial amount of static current flow.

4.4 CMOS DEVICE MODEL PARAMETERS

Changing the size of the devices does affect other parameters in the CMOS inverter. The logic threshold voltage in equations (66), (67), and (68) is a function of the device threshold voltages and the geometric ratio of the pullup-to-pulldown sizes of the devices. An increase in the logic ratio, k , increases the noise margin of the CMOS inverter. This also slows down the device. Logic threshold voltage:[13]

$$\frac{1}{2}\beta_n(V_{inv} - V_{thn})^2 = \frac{1}{2}\beta_p(V_{dd} - V_{inv} - V_{thp})^2; \quad \begin{array}{l} I_{ds} \text{ with both} \\ \text{devices in saturation,} \end{array} \quad (66)$$

$$V_{inv} = \frac{\sqrt{\beta_p}(V_{dd} - V_{thp}) + \sqrt{\beta_n}V_{thn}}{\sqrt{\beta_p} + \sqrt{\beta_n}} \quad \begin{array}{l} \text{(threshold voltage) ;} \end{array} \quad (67)$$

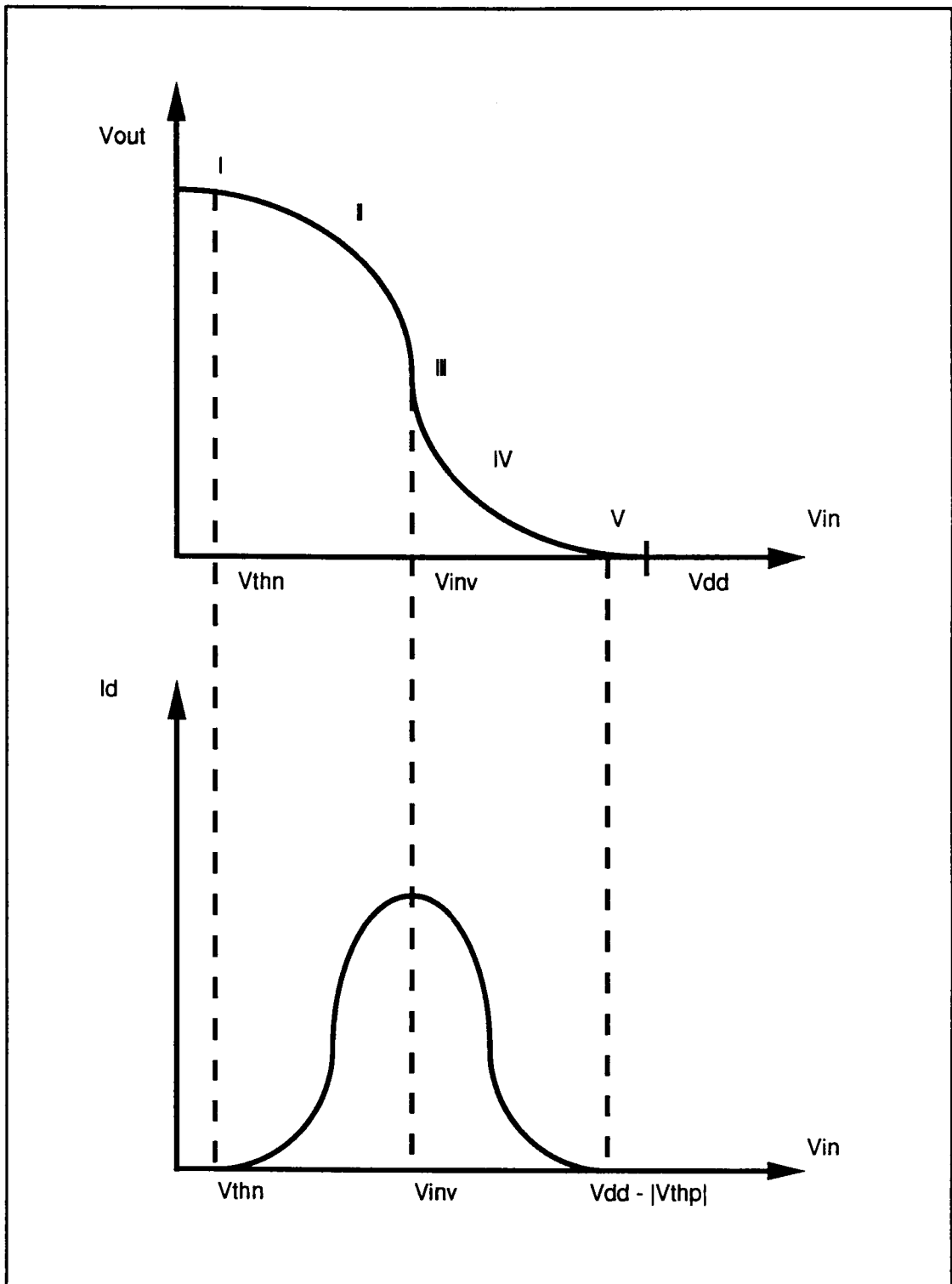


Figure 4.6. CMOS Inverter Output Current and Voltage Relationships.

and

$$V_{inv} = \frac{\sqrt{k}(V_{dd} - V_{thp}) + V_{thn}}{1 + \sqrt{k}}, \quad k = \frac{\beta_p}{\beta_n}. \quad (68)$$

Equations (69) through (74) give parameters used to module the MOSFET devices in timing simulators such as Pspice. The transistor gain parameter is shown in equations (69) through (72). Typical values for the process dependent part of the gain factor, K_p , range from 10 to 30 $\mu A/V^2$, and these values can vary from 10 to 20% primarily because of variations in the starting material and the growth of oxides on the substrate. Equations (73) and (74) describe the process gain factor and its typical value.

$$\beta = \frac{\mu e}{t_{ox}} \left(\frac{W}{L} \right); \quad (69)$$

$$\mu_n = 500 \text{ cm}^2/V\text{-sec};$$

$$\epsilon = 4\epsilon_0 = 4 \times 8.85 \times 10^{-14} \text{ F/cm};$$

$$t_{ox} = 500 \text{ Angstroms}; \quad (70)$$

$$\beta = \frac{500 \text{ cm}^2/V\text{-sec} \times 4 \times 8.85 \times 10^{-14} \text{ F/cm} \left(\frac{W}{L} \right)}{500 \text{ Angstroms}}; \quad (71)$$

and

$$\beta = 35 \frac{W}{L} \mu A/V^2. \quad (72)$$

Process Gain Factor K_p :

$$K_p = \frac{\mu \epsilon}{t_{ox}}, \quad (73)$$

and

$$K_p = 35 \mu A/V^2. \quad (74)$$

The channel length modulation factor, λ , takes into consideration changes in the drain current caused by changes in the saturation current (caused by changes in V_{ds}).

$$I_{ds} = \frac{\beta}{2}(V_{gs} - V_t)^2(1 + \gamma V_{ds}) \quad 0 < V_{gs} - V_t < V_{ds} \text{ (saturation region) ,} \quad (75)$$

$$0.02/V < \lambda < 0.04/V . \quad (76)$$

Equations (77) and (78) define mobility and give a typical value of mobility for a nFET device. The mobility for the pFET device is approximately 2.5 times smaller than the nFET value.

$$\mu = \frac{\text{average carrier drift current (v)}}{\text{Electric Field (E)}} , \quad (77)$$

and

$$\mu_n = 500 \text{ cm}^2 / \text{V} \cdot \text{sec typical value for electrons} . \quad (78)$$

The threshold voltage of a MOSFET device is shown in equation (79) in terms of the body effects contribution:

$$V_t = V_{t(0)} + \gamma \left[\sqrt{V_{sb} + 2\phi_F} - \sqrt{2\phi_F} \right] , \quad (79)$$

and

$$\gamma = \left(\frac{t_{ox}}{\epsilon_{ox}} \right) \sqrt{2q\epsilon_{Si}N} . \quad (80)$$

As Pspice analyzes each node in the network, and it builds models for the circuits it is simulating. Drain-to-source resistance can change drastically as the devices change regions of operation, and Pspice must determine new values for use in modeling the circuit as the regions change. For the linear region of operation, the resistance is approximated as a linear function of the V_{gs} . In the saturation region, the resistance is approximated as infinitely high.

$$\lim_{V_{ds} \rightarrow 0} \frac{dI_{ds}}{dV_{ds}} \approx \beta(V_{gs} - V_t) , \quad (\text{linear region}) \quad (81)$$

$$R_{ds} \text{ LINEAR REGION}^a = \frac{1}{\beta(V_{gs} - V_t)} , \quad (\text{linear region}) \quad (82)$$

$$\frac{dI_{ds}}{dV_{ds}} = \frac{d\left(\frac{\beta}{2}(V_{gs} - V_t)^2\right)}{dV_{ds}} = 0, \quad (\text{saturation region}) \quad (83)$$

The transconductance parameter in equation (84) is used to define the relationship between changes in the V_{gs} and changes in the drain-to-source current. Note that the transconductance is a linear function of V_{ds} and the process gain factor in the linear region and is approximated as being directly proportional to V_{gs} in the saturation region with no effects caused by changes in V_{ds} .

$$g_m = \frac{\partial I_{ds}}{\partial V_{gs}} \quad V_{ds} = \text{constant}, \quad (84)$$

$$g_m = \beta V_{ds}. \quad (\text{linear region}) \quad (85)$$

$$g_m = \beta(V_{gs} - V_t). \quad (\text{saturation region}) \quad (86)$$

4.5 CMOS TRANSMISSION GATE

The CMOS transmission gate takes advantage of the strengths of both of the enhancement devices by putting the devices in parallel. The nFETs used in nMOS designs pass a strong "0" but a weak "1." This is caused when the device cuts off as the output voltage reaches the threshold voltage of the input voltage. At this point, the V_{gs} is less than the threshold voltages, and the nFET cuts off. By putting a pFET in parallel with the nFET, the pFET's ability to pass a strong "1" keeps driving the output to V_{dd} after the nFET shuts off. Likewise, the nFET will drive the output node to a strong "0" after the pFET cuts off. The CMOS transmission gate is shown in Figure 4.7.

4.6 CONCLUSIONS

The implementation and design of the moment invariants algorithm was performed with scalable CMOS devices. This low-power-consumption, ratioless, fully restored family of devices made designing the individual circuits less complex than designing with nMOS. The disadvantage in the selection is the reduction in possible speed and the higher possibility of the occurrence of latch-up.

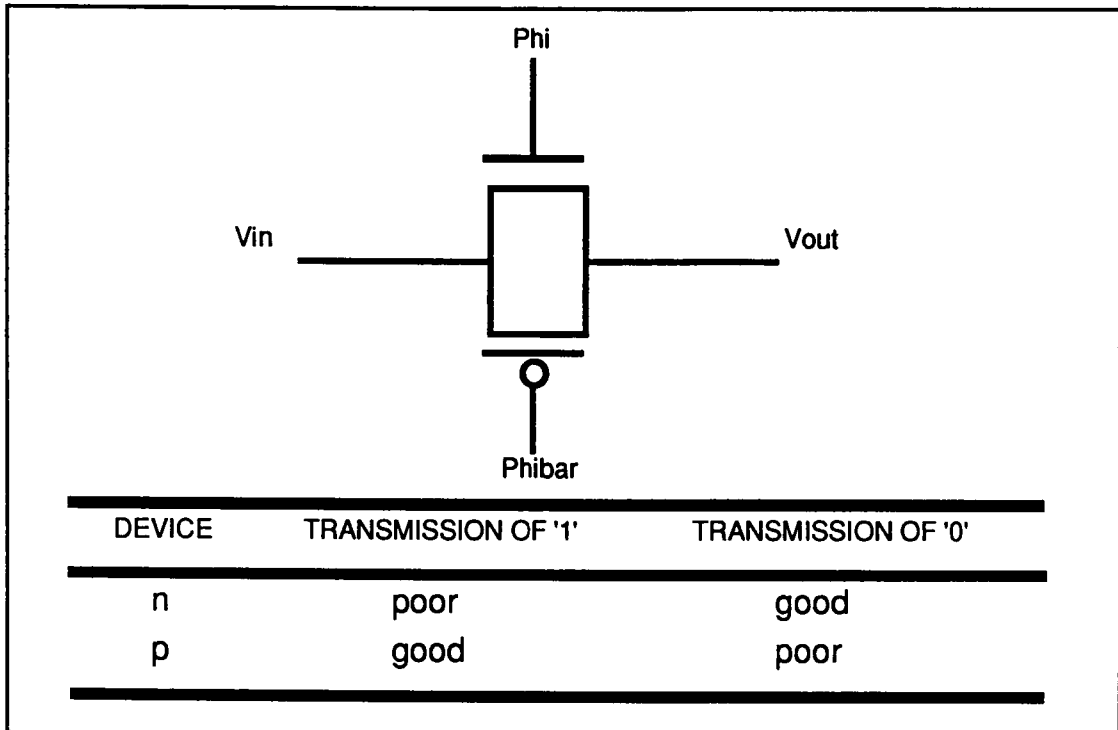


Figure 4.7. CMOS Transmission Gate.

CHAPTER 5

ARCHITECTURE

This chapter presents architectures that are considered for implementing the moment invariant algorithm. A high priority will be given to architectures that allow for a high degree of replication of internal cells while maintaining a simplistic approach to deriving the moment. Array based architectures that allow for a maximum amount of parallelism, while maintaining simple interconnection paths between array cells will be sought for implementation of the moment generator. The design goals for the moment generator and the limitations of the physical chip are presented as part of the criteria used to select a suitable architecture. A conventional architecture is presented for the implementation of a moment generator. A few of the architectures used in pattern recognition and image processing applications are presented in this chapter. The conclusion presents an architecture that will allow optimal performance of a moment generator in the pattern recognition algorithm.

5.1 CONVENTIONAL VLSI ARCHITECTURES

VLSI designs are incorporated into a system when the solution by conventional methods is very time consuming and when the algorithm used to solve the problem can be broken down into simple modules that can be implemented with a high amount of repetition. As a result, most VLSI architectures use parallelism through arrays and pipelining to optimize the performance. Five popular architectures for implementing VLSI designs to solve pattern recognition problems are presented.^[4] The simplest architecture is the one-dimensional linear array that is shown in Figure 5.1. This also is referred to as a data path, where each processor in the path can perform the same function on the data traversing the path or each processor can have a unique purpose.

The two-dimensional array shown in Figure 5.2 has one extra dimension for data flow and is capable of solving more complex problems than the linear one-dimensional array. While the linear array offers the capability to pipeline the processing of the data and therefore offers parallelism, the mesh array offers a higher degree of parallelism and connectivity along with the capability to process two parallel input vectors.

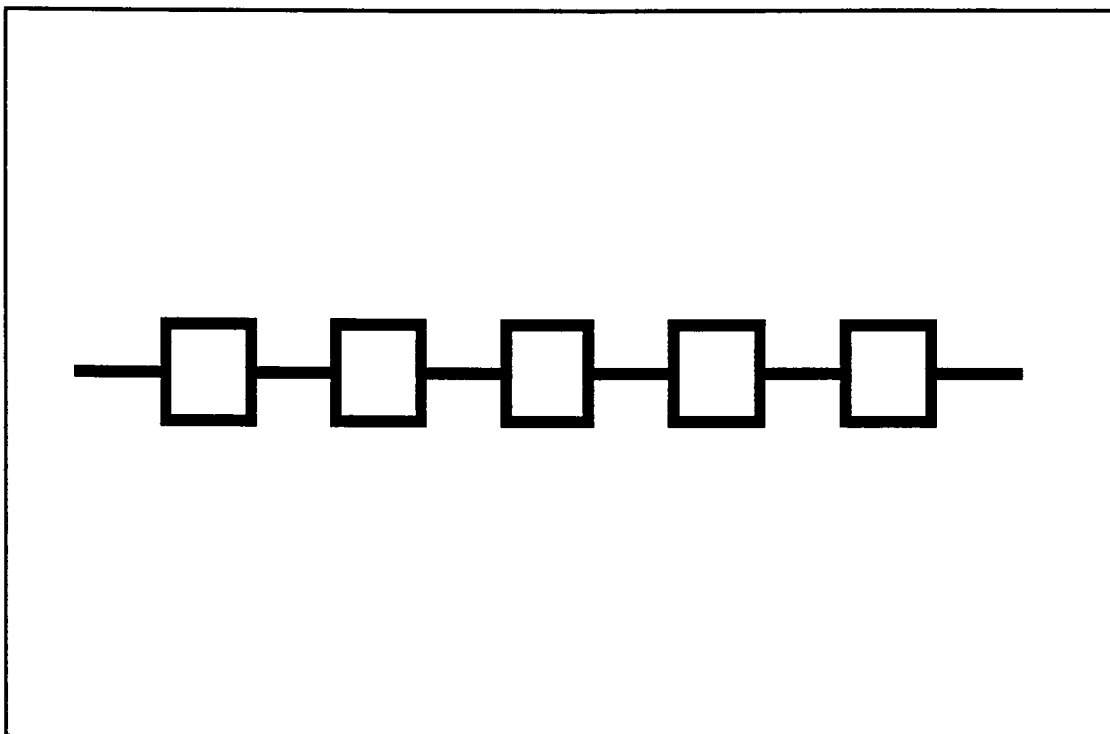


Figure 5.1. One-Dimensional Linear Array.

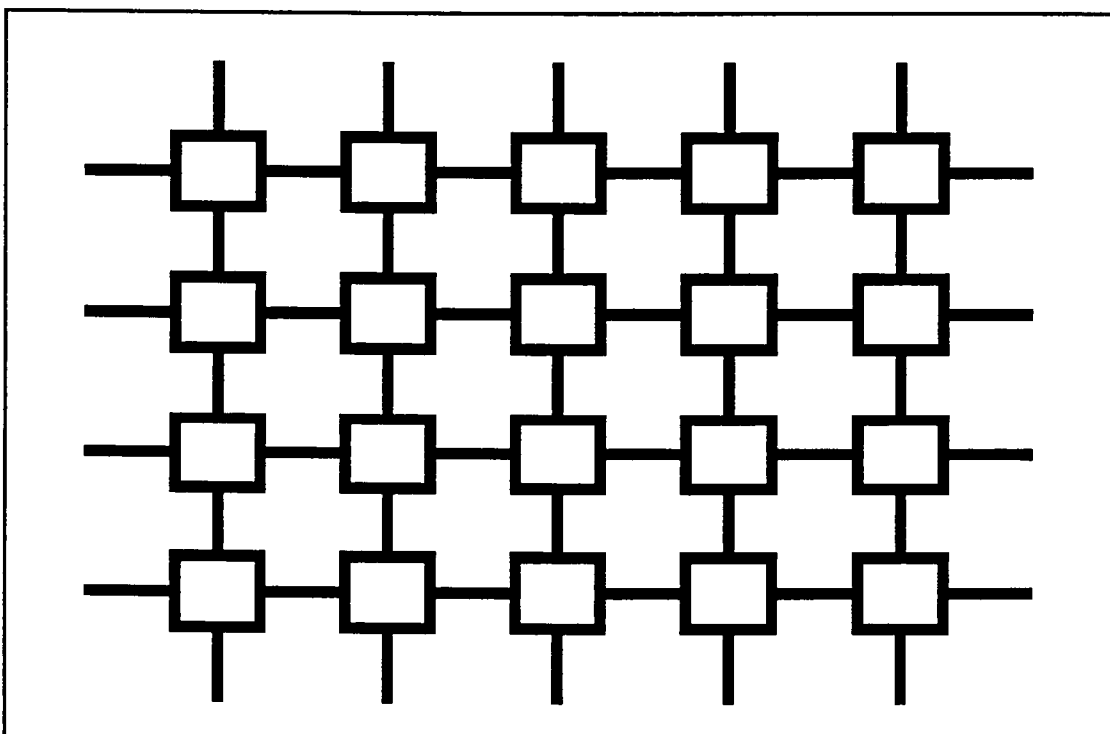


Figure 5.2. Two-Dimensional Orthogonal Array.

Figure 5.3 is a two-dimensional hexagonal array. This array has one more data path than the orthogonally connected array. The connectivity between the processors helps to qualify the arrays for performing certain type of problems more effectively than others. The hexagonal array typically is used for matrix multiplication or LU decomposition or other functions that require both orthogonal and diagonal data flow.

The one-dimensional linear array and the two-dimensional square and hexagonal arrays are mesh-connected processor arrays. The information paths between these arrays are simple and regular. The tree architecture (Figure 5.4) is more complex. The tree architecture derived its name from the resemblance to an upside-down tree. The first node in the tree is the root node, and the remaining nodes branch off of the root or off of one another as branches do in a tree. For a tree array to be needed for a given application, the problem to be implemented must have a growth pattern that increases exponentially. The node processors in figure 5.4 must be able to break the problem further into two more parts. The communications bandwidth of this type of architecture is concentrated at the lowest levels. If a given application requires more communication at the top level than is provided by the one

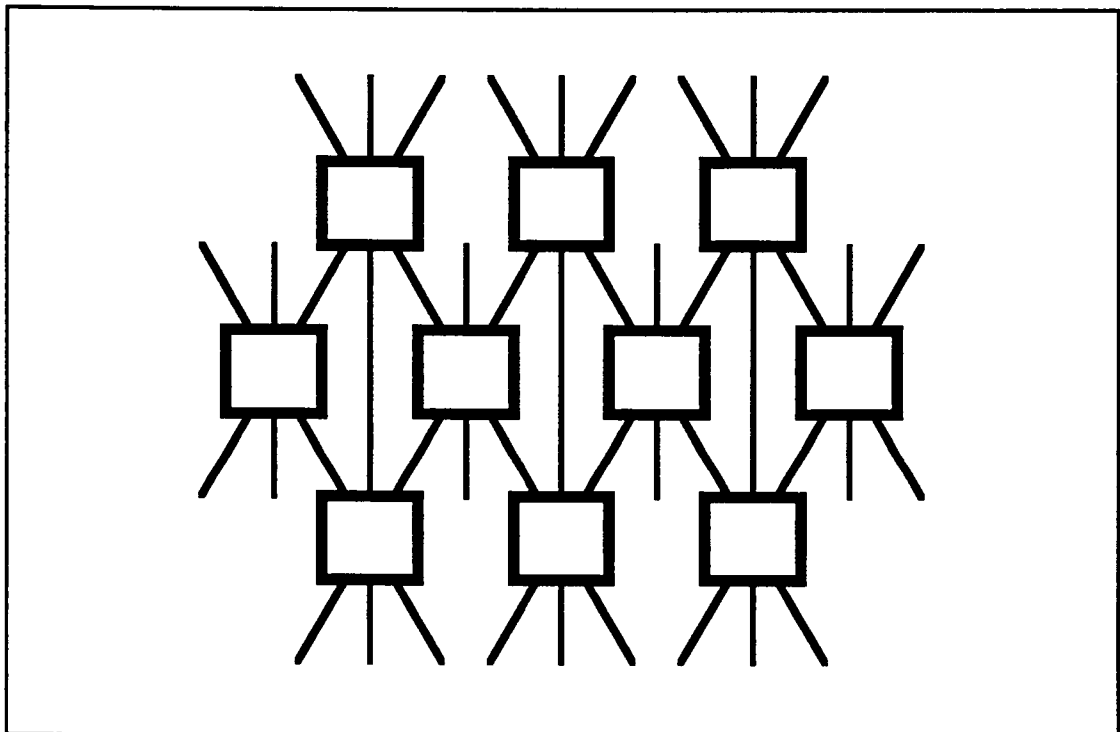


Figure 5.3. Two-Dimensional Hexagonal Array.

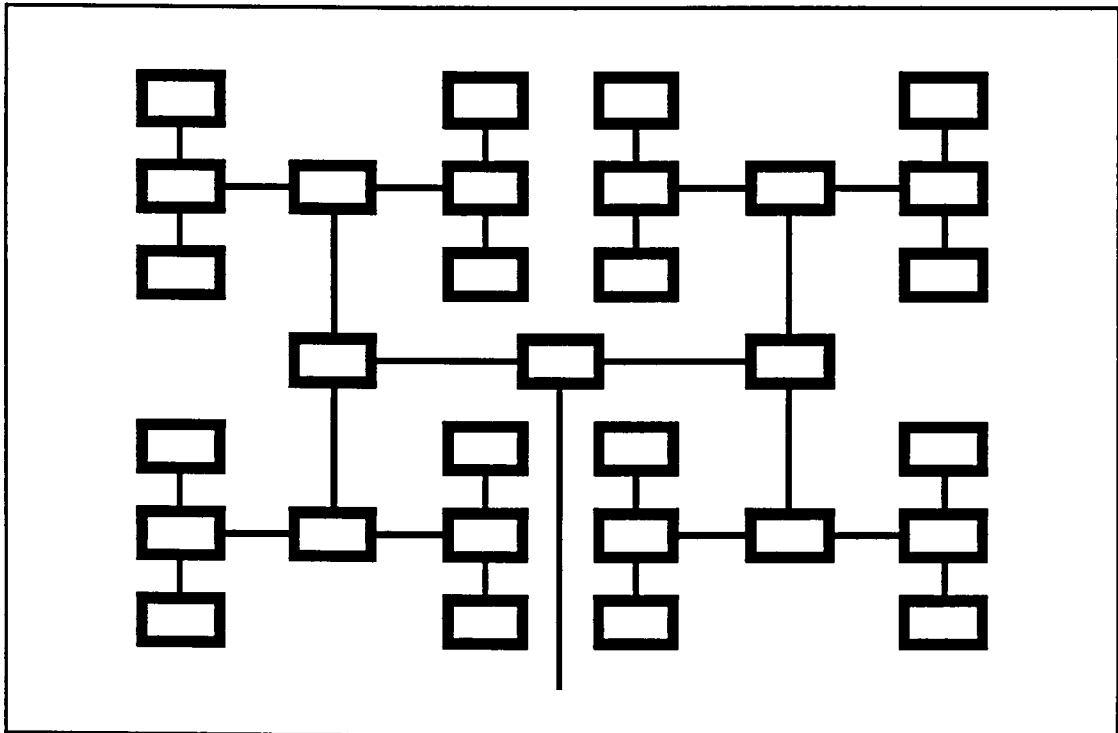


Figure 5.4. Binary Tree Architecture.

communication path to the root node of the tree, the process will not be able to take advantage of the capabilities of this architecture. The binary tree array in Figure 5.4 is particularly useful in applications requiring a search process. Sort algorithms are examples of problems that would use the resources of a tree architecture. The binary trees strength is being able to search through its members in less time than a one- or two-dimensional array.

Systolic arrays attempt to use data in such a manner to maximize computational efficiency while minimizing throughput. This is achieved by ordering the data so that it is used effectively by each cell that it passes through. In a typical application, a systolic array will be able to produce one piece of output data for every piece of input data clocked into the array. This also stipulates that a systolic based design must be synchronous rather than transparent or asynchronous. For the multiply function this would mean that for every pair of operands entered into the systolic based multiplier array, one product would be produced. This would not be the product for the current inputs but the product for a earlier pair of inputs. The series of partial products for a series of given multiplications would ripple through the systolic array in the same manner that a series of waves would ripple through a pond. The method uses the

principle of pipelining to allow several multiplications to occur in parallel in the array. As a result, a high level of computational efficiency can be attained with a low I/O bandwidth. To allow the cells to be utilized in this way the data has to be preprocessed before it can be input into the systolic array and post-processed on the way out. The use of pipelining will require a pipe to be inserted between each row of processor cells in the array. In addition, the preprocessing and the post-processing will require a large amount of register cells.

The architecture selected for implementing the moment generator function should allow regular data flow paths between internal modules and should allow the functions to be implemented with as much repetition of the basic building blocks as possible. While the data path or linear array provides for one parallel input vector, two parallel input vectors are needed for the multiplication function in the moment generator. The hexagonal and orthogonal arrays provide for two parallel input vectors and allow a high degree of connectivity between cells that can be modified to allow for the multiplication scheme selected. The high amount of data into the moment generator eliminates the possibility of considering schemes that may be incorporated using tree architectures due to their requirement for limited throughput at the root node. The systolic array could be considered for more advanced multiplication schemes, but its requirement to reorder the data so each cell could be used at every clock cycle would require a large number of register cells for preprocessing the input data vectors and output results as well as pipe registers between cells in the array.

5.2 MULTIPLICATION ARCHITECTURES

Figures 5.5 and 5.6 illustrate a simple arrays used to implement a multiplication operation. Each element in the array is composed of an adder cell, the exact composition of which does not have to be defined at this point. This array has two input data streams and attempts to make full use of each data item as it passes through the cells. The connections implement a carry propagate adder approach. Each carry must be allowed to propagate through each row before the results from that row can be properly processed by the next row. For a multiplier array with operands M- and N-bits wide, the carry would have a maximum propagate path that would go through M+N cells. The effect of the long propagate time for the carry can be reduced by the addition of look-ahead carry logic that can be added to each row of adders.

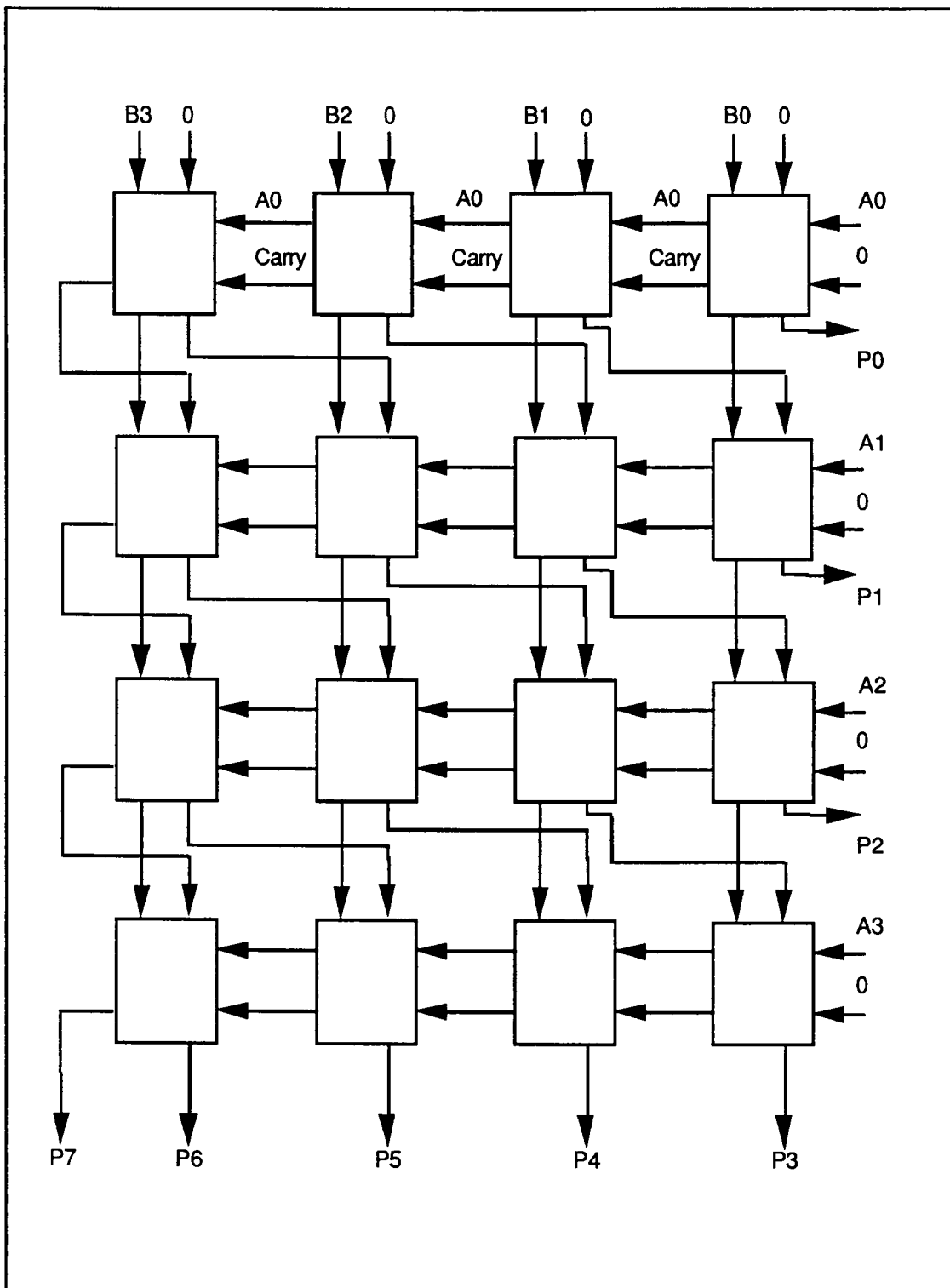


Figure 5.5. Simple Multiply Array Based on the Carry Propagate Adder.

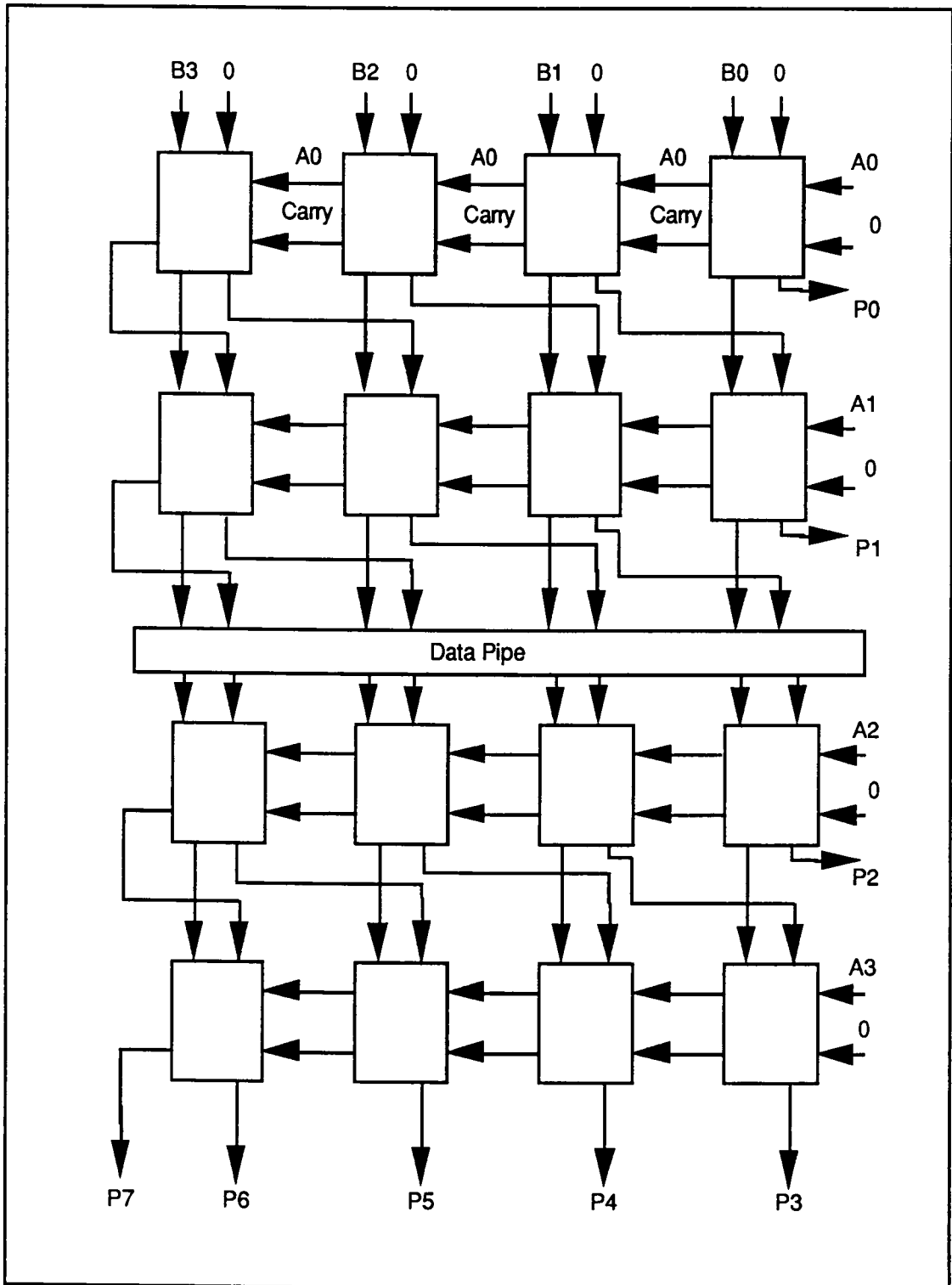


Figure 5.6. Simple Multiply Array with Pipelining.

The propagation time through the worst case timing path, referred to as the critical path, becomes a limitation of this design. The maximum size of the multiplier becomes limited to the maximum time allotted for the multiplication and the number of cells that the carry signal can propagate through in this time. The critical-path time can be reduced by the addition of pipes in the multiplier array as shown in Figure 5.6. This breaks the multiplier array into smaller segments, which allows larger arrays to perform in a shorter period of time by allowing parts of the multiplication to occur in parallel. In addition, the data pipe offers the extra advantage of providing a scan path for implementing built-in testing.

Carry save adders also could be used to further decrease the dependence of the multiplier array on the size of the multiplication operands. In the carry save adder approach, each adder cell takes the same three inputs and presents two results to the next row of adders. (The carry propagate technique takes three inputs and presents one result.) The propagation time for the carry is reduced to the time for the adder to add the three inputs to produce the two results. The last stage of the multiplier would have to be a carry propagate adder, but the time delay caused by this technique could be reduced further by the addition of a look-ahead carry circuit to the final stage.

Other optimizations can be performed if the designer is willing to do more customizations to the design. At this point, all of the adder cells in the array are the same, which this keeps the data paths simple. The first row of adders can be replaced by logical AND gates as there is no addition to be performed in the first row and no carry to propagate. A modified Booth recoding implementation^[16] also can be used to cut the number of rows in half at the expense of an increase in the layout area.

5.3 PHYSICAL DESIGN LIMITATIONS

Before deciding on the architecture best suited to implement the multiplication operation, the requirements for all functions on the chip should be examined. The design goals dictated the speed and size of the desired operations, and the physical constraints of the VLSI platform for implementing the chip put limitations on available pins and the area of the silicon available for the design. To perform the ordinary moment calculation of each image at frame rates (30 frames per second), each pixel will be in a data stream being fed into each

moment generator processor at a rate of 7.68 Mpixels per second. This allows 127 nanoseconds per operation before the chip has to input the next pixel.

On the NSF-supplied platform, 40 pins are available for getting data in and out of the chip along with support functions such as clocks and power. Six I/O pins are used for power and ground connections. Four pins are dedicated to power and ground of the I/O buffers, and two extra pins must be used to provide power and ground to the core of the ASIC or the ordinary moment generator. The four I/O pins used for power and ground of the I/O buffers are an extra precaution against latch-up and prevent noise from the switching of large these capacitance nodes from reaching the center of the chip where the smaller switches are more susceptible to noise problems.

The NSF-supplied chip platform provides a silicon real estate area of 1800X1800 micrometers. Approximately 100 micrometers should be cut off of this area on all four sides to leave a buffer area for getting metal runs from the moment generator to the I/O pins. This leaves an area of 1600 x 1600 micrometers for the moment function.

5.4 CONVENTIONAL MOMENT GENERATOR ARCHITECTURE

Figure 5.7 illustrates an architecture that typically is used to implement the moment determination. This design makes maximum use of the data once it is in the chip by repetitively performing multiplications and additions necessary to derive the desired order of the moment. The third-order ordinary moment is the highest moment necessary for the moment invariants algorithm. This design would be required to implement three multiplications and one addition per pixel of input data. The process must be done in less time than required to read in the next pixel. Feeding the pixel data into the chip at 7.68 MHz allows 127 nanoseconds for the worst case third-order moment determination, or 31.8 nanoseconds for each of the three multiplications and the final summation. This assumes that the data input function and the data output function are separated from the moment determination through pipelining. A major advantage to this type of approach is the potential usefulness of this design in many applications. This is the basic architecture used to implement many special purpose image processing engines. If the multiplier function could be expanded to do division and access to the subtraction and addition functions could be included as an option in the data path, the chip could function as a complete arithmetic processing unit.

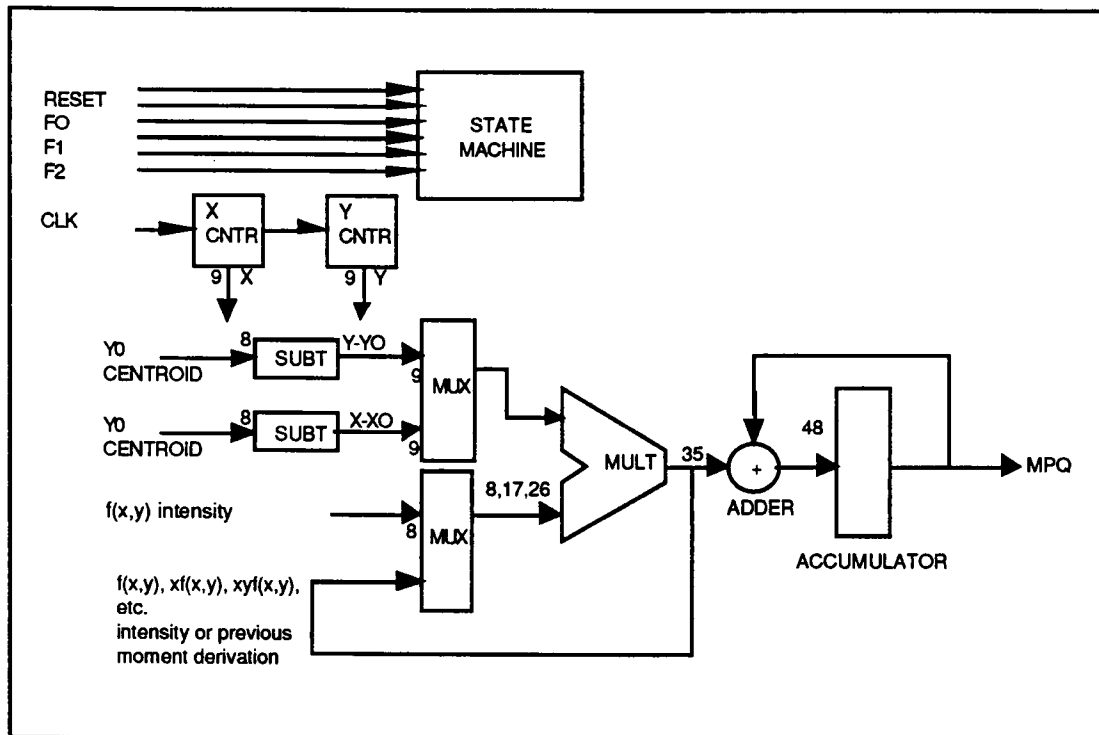


Figure 5.7. Conventional Moment Generator Architecture.

The counter "CNTR" and subtraction "SUBT" functions of the conventional design allow the positional determination of the pixel in the image to be performed completely on the chip relative to the centroid. This is done to reduce the flow of information into the chip. This design requires the x and y centroid to be fed into the chip for each frame and the intensity to be fed into the chip one pixel at a time at 7.68 MHz. A multiplexer then can be used to determine which positional difference term to use in the present multiplication, saving a considerable amount of I/O pins. Reducing the I/O activity also reduces power consumption since the ASIC I/O pins often require more power than the application.

The state machine for this design could be implemented as a counter that counts the steps necessary to do the desired operation and logic and signals which functional module to execute at any given step or as a PLA. The state machine would be acting similarly to the write control store (or microcode) of a microprocessor. The state machine would have to stipulate which values of delta x and delta y to use during the sequence, and the inputs to the state machine would have to indicate which order of the moment that the ASIC was to implement. The state machine should be designed with the capability to switch control of the chip internals from the internal state machine to a device such as a ROM or PLA external to the ASIC.

Although this would be slower than using the internal state machine, it would allow a method of testing the functionality of the design and a fallback method for implementing a state controller if the one on the chip failed. This design would maximize the use of the silicon in each ASIC because each of the 7 moments could be derived on one chip. It also would allow maximum use of the internal bandwidth of the ASIC by performing 11 or 12 steps up to three times for the third-order moments in the time required for the I/O circuits to read in one pixel or to output 1 moment result. If the moments could not be calculated in a sequential manner and meet the performance requirement of 7.68 Mpixels per second, pipelining could be added to the moment generator to allow functions to operate parallel.

The complexity of the conventional design approach is another issue. The possibility for error caused by the lack of a regular simple design would be high. If any of the timing signals became skewed significantly, the chip would be nonfunctional. Signal skew is directly proportional to the square of the wire length and varies with temperature. The temperature across the silicon can vary as much as 20% with some designs. It would be important to note which segments of the design run hotter because of more activity and to make the controls signals to these areas as tolerant as possible to signal skew. Timing analysis and careful simulation of the final design would be very necessary for this type of approach, which would be limited by the inability of the simulators to simulate the effects of temperature or temperature gradients.

The state machine architecture may be a good design approach for the final chip that is constructed on a 100-, or more, pin chip with a large amount of real estate. MOSIS can generate up to a 132-pin chip with real estate on the order of 8218X9779 micrometers². The chip that comes with the NSF program is 1800X1800 micrometers² with 40 I/O pins. The NSF platform prevents using the conventional design because of the lack of area to implement all the functions and the lack of I/O pins to get data in and out of the chip. In addition, designing the custom cells required to implement this design would take an excessively long. If this design was implemented it would have to be done with standard cells, and extensive simulations would be necessary to decrease the probability of failure caused by timing errors.

5.5 CONCLUSIONS

Because the internal switching speeds of CMOS transistors reach 100 to 120 MHz, the CMOS transistors combined with pipelined parallel processors should be capable of calculating the 3 multiplications and 1 addition on each pixel in less time than it takes to read the next pixel from a TTL interface at 7.68 MHz. The limited space and I/O pins available on the 1800X1800 micrometer chip will prevent the implementation of a moment generator large enough to perform moment calculations on the 256 grey-scale 512X512 pixel images. The space is available for implementing a prototype for proof of principle.

The basic architecture for the multiply and the summation functions was selected to be a data path at the highest level of the moment generator system. The multiply would be the first element in the cell and the summation would be the second. The architecture of the multiply was selected to be based on the hexagonal and orthogonal two dimensional arrays. The architecture selected for the prototype was based on the implementation of dual functionality required at each chip. The data path architecture would simplify the design task by allowing the chip to work without the use of the state machine, which would be necessary to implement multiple moment derivations on one chip.

The chips should be designed such that a first-order moment calculation or a multiplication with no summation can be performed on each chip. This would allow the chips to be cascaded together to perform the third-order moment calculations necessary for the moment invariants algorithms. The dual function capability of the ASIC would allow the ASICs to be arrayed into a hierarchy at the chip level on the printed circuit board. This allows for multiple use of the same chip.

This design allows the two multiplication operands to be fed into the left and bottom of the array and the resulting product to be taken from the top and right of the orthogonal two-dimensional array. The multiplicand and the multiplier will pass orthogonally through the array while the resulting partial products will propagate diagonally through the array until reaching the periphery of the array, at which time it would be final product. The regular nature of the array would make the design easy to implement and allow use of the same adder cell for all processors in the array. This allows for one adder cell to be designed and thoroughly simulated and then replicated into an array to form the multiply function. The array also can be

broken up easily into sections for the insertion of pipes to speed up the calculations if the carry propagate operation becomes excessively long. This implementation has the advantage of also allowing scan paths to be incorporated into the pipes for quick self-testing and observability of internal nodes.

The carry propagate adder implementation, based on the two-dimensional orthogonal and hexagonal array, was selected. The carry propagate adder was chosen over the carry save adder because of its capability to read the actual intermediate result in each pipe. This will greatly simplify simulation and testing of the chip because the data going into the scan path between two rows of adders will not have to be encoded and the data read out of the scan path will not have to be decoded. For each adder cell, the carry save adder passes forward two bits, the sum and the carry bits, to the next row or bank of adder cells. This requires that the pipe between each row of adders be twice as long as the pipe for the carry propagate adders. The carry look-ahead function is used to reduce the effects of the propagation of the carry through a long bank of adders.

The advantage of the carry save adder approach is that it eliminates the carry propagate path as the critical timing path in the adder bank. The carry must still be allowed to propagate through the last bank of the adder for the formation of the final product. The propagation time is normally reduced in the last stage through the use of a carry look-ahead function. This represents a large amount of overhead in the form of real estate necessary to implement the carry look-ahead function. The design of a carry look-ahead function is based on the same ideas used for the Manchester carry conditions but for more than two input bits (generally four). To reduce the effect of the carry propagate in this design, Manchester carry conditions were used as the basis for the design of each adder cell. This allowed the carry propagate adder scheme to be used for each bank while minimizing the time for the carry propagate path in each bank of adders without the use of a carry look-ahead function.

CHAPTER 6

DESIGN AND IMPLEMENTATION

The design goals for VLSI ASICs are highly dependent on one another. Design techniques that reduce area usually reduce speed. As speed increases so does the switching power, which increases the size of the metal power and ground runs that sink and source current to the internal functions. Design techniques that increase speed also reduce the noise margins of the devices making them more susceptible to noise-related failures. More regular and modular designs that make the circuit easier to design and fix usually require more area. Including special test circuitry on the ASIC reduces the area available for the main functions; however it, reduces the time spent testing the chip and allows testing to more thoroughly determine the success of the design. All of these factors must be considered for a VLSI ASIC design.

6.1 DESIGN GOALS

6.1.1 PERFORMANCE GOALS

The main performance goal chosen for moment generator design is speed. To process 256 grey-scale 512X512 pixel images at a real time rate requires processing data at 30 frames per second, or 7.68 M pixels per second. Performing the worst case third-order moment calculation requires 3 multiplications and 1 addition per pixel. Hu's equations require 7 moments, 3 second-order moments, and 4 third-order moments. The second-order moments require 2 multiplications and 1 addition. The third-order moment requires 3 multiplications and 1 addition. Moreover, the centroids require that the zeroth-order ordinary moment be calculated and divided into the 2 first-order ordinary moments. The normalization of the moments requires the use of the zeroth-order ordinary moment. (The division has to be performed by the computer or special-function math processor.) Once the centroid is determined, two subtractions per pixel must be performed to derive the positional relation of the pixel to the centroid for use in calculating the centralized moment. (See Chapter 3 for a discussion of the moment invariant algorithms and supporting equations.) Hence, a total of 30

math operations per pixel processed must be performed, or 235.8 MOPS for the ordinary and centralized moment calculations part of the moment invariant algorithm. This assumes that an off-the-shelf chip that can subtract will be used to determine the delta x and delta y values for the centralized moment calculations. The results of the centralized moment calculations by the ASICs must next be fed to Hu's and Maitra's moment invariant equations. To keep up with the data stream from the image digitizer system, each operation has to be done in less than 127 nanoseconds. Each pipe in the process must be able to read data, process data, or output data in this amount of time.

By using parallel interfaces capable of TTL speeds of 10 to 20 MHz, the 7.68 MHz bandwidth on the I/O to the chip is attainable for a parallel transfer. The multiplier, adder, accumulator, input, and output functions can be pipelined to give each function the full 127 nanoseconds to perform its operation. Each pipeline requires more space on the chip, so the pipelines will have to be integrated in as room allows. Figure 6.1 depicts the desired layout of the functions used to implement the moment generator used for an initial design goal, which was based on the use of full adder cells arranged in a two-dimensional orthogonal matrix for the multiply function.

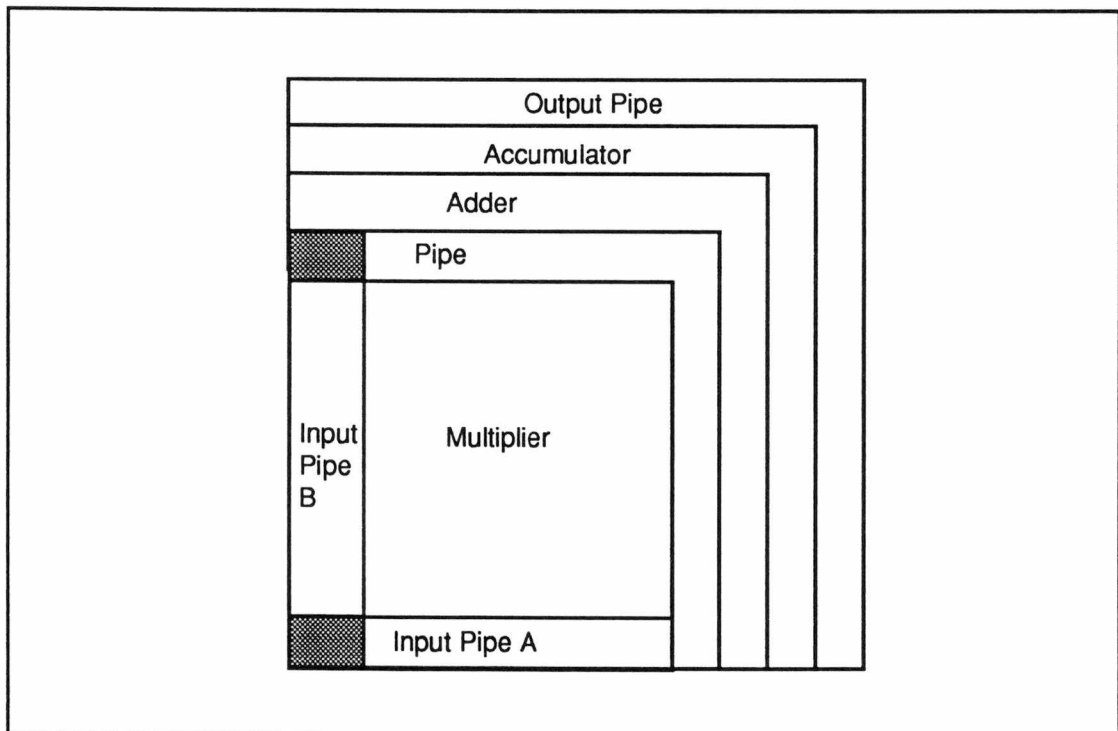


Figure 6.1. Pipelined Moment Generator Layout.

6.1.2 AREA GOALS

The real estate on the NSF platform is 1800X1800 micrometers. Because the I/O pins and timing considerations limit the input scheme to two parallel input operands of 8 and 4 bits in width and an output scheme with one parallel word 12 bits in width counting overflow, the design goal for the layout of the multiplier was set to 1300X1300 micrometers. The internal cell size of the adders that make up the multiplier array have to be 160 micrometers high by 300 micrometers wide to allow for a 8X4 multiplier. This leaves two sizes that must be designed for the rest of the adder cells and register cells used in the adder, accumulator, and registers used as pipes and scan paths. The horizontal rows of the pipes, adders, and accumulators can be 100 micrometers high by 300 micrometers wide. The horizontal rows can be 160 high by 100 long. The minimum size is the target design size for the cells.

6.1.3 I/O DESIGN GOALS

The 1800X1800-micrometer chip has 40 I/O pins. Of these 40 pins, four are allocated to power and ground for the pads and I/O buffers. Two more pins are allocated for power and ground for the chip internals. This leaves 34 pins available for input and output of data, control signals, and test functions. One reset function is supplied to set all storage nodes to a known state. The clock will require one input pin to implement a single-phase clock (the complement of the clock is generated internally). The test pins include one serial input pin and one output pin for each scan path and a control function pin to switch the registers used in the scan path between their normal parallel functions and their serial test functions. The implementation of the 8X4 multiplier and the 11 bit accumulation function requires 12 input and 11 output pins. The pins for the 8X4 multiplier and 11 bit summation functions are shown in Table 6.1. Test In and Test Out are two I/O pads that are connected only to each other. This will be used as a test to ensure that the fabrication process is completed successfully.

6.1.4 POWER DESIGN GOALS

The maximum current rating for a metal line is about 1 mA per micrometer of width of the metal run sourcing or sinking the current. The heat from high levels of current density can cause metal migration, in which the metal molecules are removed from the hot point, and deposited in cooler areas on the metal run. This process continues until a gap is developed in

Table 6.1. I/O Pin Assignments.

Function	Pins
Power	
I/O Pad Power	4
Internal Power	2
Moment Generator	
Product Input Pins	12
Moment Output Pins	11
Multiply Overflow	1
Add Overflow	1
Control	
Clock	1
Reset	1
Add Control	1
Scan Path Control	1
Testing	
First Pipe Scan Path	2
Accumulator Scan Path	1
Pad Test	2
Total	40

the metal run and no more current flows. Metal migration occurs when the current density reaches 2 milliamperes per micrometer of width. The metal runs that supply and sink current to the CMOS devices must be large enough to handle the maximum loads encountered for all operations. For a CMOS circuit there is no static power dissipation (a negligible amount of power only is lost because of leakage currents at dynamic storage nodes), and all power consumed is used for switching. Pspice simulations show the drain currents to be at or lower

than 2 to 3 nanoamperes for the cells and less than one-tenth of a microampere for the entire moment generator. The power and ground mesh was run horizontally through the chip in parallel with the data flow. The power and ground runs had a maximum of five adder cells and two register cells per run. Consequently, current levels are estimated to be much less than the 1 mA per micrometer limit. The minimum width of a metal run is 4 micrometers, and this was more than tripled for the power and ground runs that are approximate 15 lambda wide each.

The power estimates for the adder cell, register cell, and the moment generator are given in Table 6.2. The equation used to estimate the power consumption of the inverters in the moment generator chip is shown in equation (87). The frequency of operation, f , is 7.68 MHz. A conservative estimate for the load capacitance, C_L , of a typical inverter is 0.2 pF. The power supply voltage, V_{dd} , is 5.0 volts.

$$P = V_{dd}^2 f C_L. \quad (87)$$

The adder cell has a total of 47 transistors which is estimated to be 25 inverter dynamic power loads. Using equation (87), the power estimate for the adder cell is 0.983 milliwatts. The register cell has 34 transistors which is estimated to be 17 inverter dynamic power loads. The estimated dynamic power consumption for the register cell is 0.668 milliwatts. One slice of the moment generator function (not including any considerations for the I/O pads) contains five adder cells and two register cells. The power estimate for the slice is 1.65 milliwatts which translates to 0.33 milliamperes. The moment generator function has 43 adder cells and 22 register cells. The power estimate for the moment generator (with no consideration for the I/O pads) is 67 milliwatts and the current is estimated to be 11.4 milliamperes. A knowledge of the I/O pad capacitance will give a better estimate for the whole ASIC.

Table 6.2. Moment ASIC Internals Power Estimate.

	Adder Cell (43 each)	Register Cell (22 each)	Moment Generator
Power (watts)	0.983E-03	0.668E-03	67.0E-03
Current (amperes)	0.197E-03	0.133E-03	11.4E-03

With a current estimate of the slice of the moment generator being 0.33 milliamperes, the 16 lambda power rail and the 14 lambda ground rail for the slice is more than adequate to meet the dynamic current demands. The power estimate indicates that the power and ground rails are roughly 45 times larger than necessary. These metal run sizes could be reduced to allow more room for an additional slice of the moment generator to be added to the moment generator function but were conservatively designed to handle any unanticipated peak current demands that the adder and register cells may require.

6.2 DESIGN FOR TESTABILITY

There are two main concepts that deal with design for testability, controllability and observability. Controllability is the capability to set and reset every node in the circuit. Observability is the capability to observe either directly or indirectly the state of any node in the circuit. There are many methods that can be used to implement testability and controllability for the VLSI ASIC by either completely building the test function into the chip or designing interfaces that allow the chip to be tested in conjunction with testing information formulated externally to the chip.

A popular method for implementing built-in testing is signature analysis. This method incorporates a linear feedback register as shown in Figure 6.2. The patterns generated are always the same once the feedback circuit is tied into the shift register. The pseudo-random nature of the patterns is determined by the place on the register from where the inputs to the exclusive-OR (EXOR) gate are taken where the outputs are deposited in the same register.

Another popular approach to built-in testing is the Level Sensitive Scan Design (LSSD) approach (Figure 6.3) developed by IBM. This method allows a test vector to be written into a serial shift register and then applied in parallel to a combinational logic circuit. The results of the combinational circuit to the test vector then are stored in the same shift register. One of the most thorough designs for built-in testing is the Built-in Logic Block Observation (BILBO) method for testing the blocks in the chip. The BILBO method combines IBM's LSSD method with signature analysis to produce a built-in test function that can be reset to a known state and generate pseudo-random test patterns.

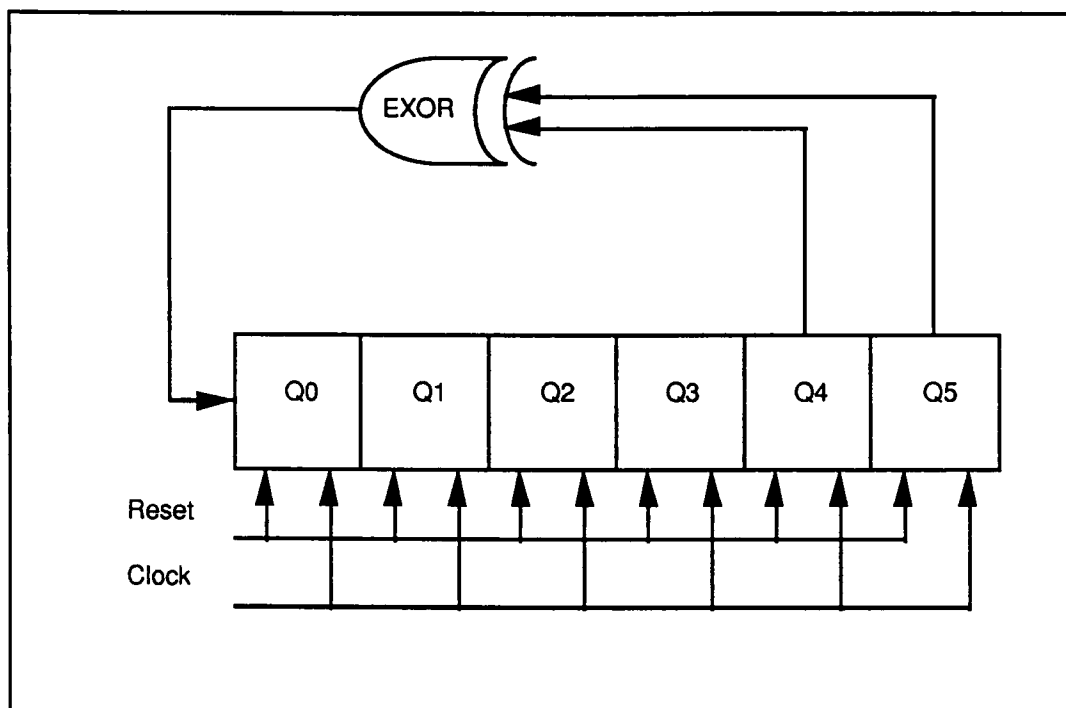


Figure 6.2. Linear Feedback Register used for Signature Analysis.

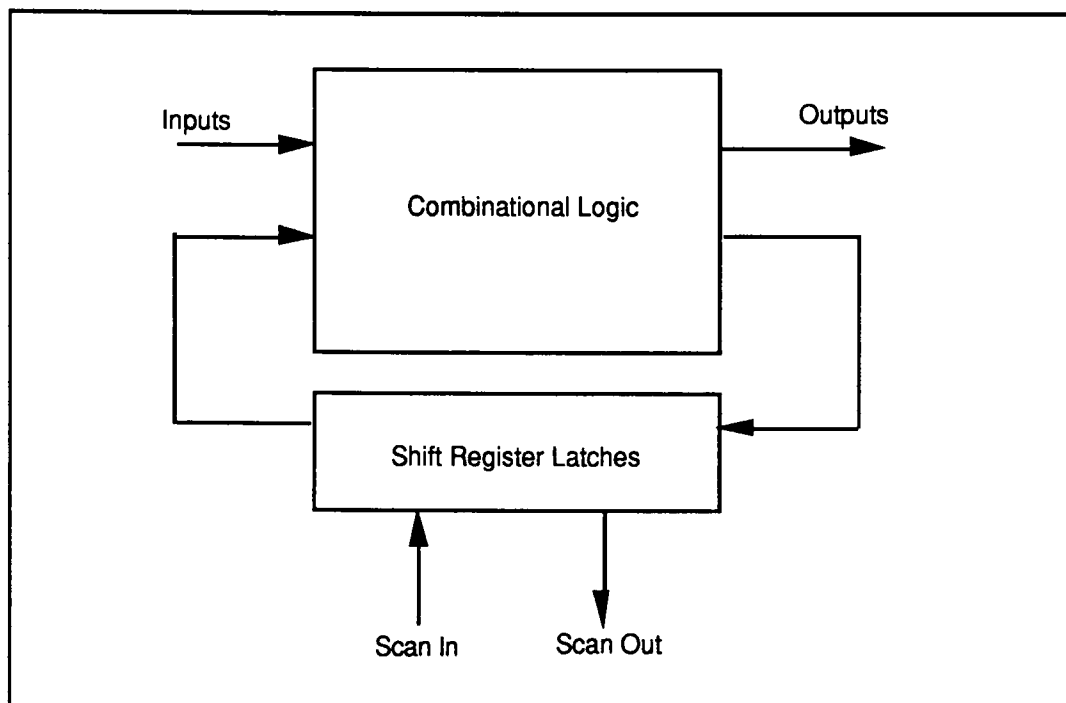


Figure 6.3. LSSD Built-In Test Block Diagram.

used for input, and the data is output serially for examination. Automatic test programs are available for combinational logic functions, further reducing the work involved in testing combinational logic designed with this approach. The disadvantages of this approach involve a loss of performance in the combinational circuit caused by the added circuitry in the feedback path and a loss of chip real estate and I/O pins.

Multiplexing path built-in test functions can be used to select test vectors or the normal data stream for input into a combinational logic function. Figure 6.4 illustrates the multiplexing path method of testing with pattern generators and analyzers incorporated on the chip. The circuit can then be quickly and completely tested in place with no external interaction. This is a very useful design for ASICs that are used in a system requiring a high confidence degree in the correct operation of the internal components. The ASIC can be programmed through the use of an internal state machine to run through its diagnostic periodically or to detect periods of inactivity and continuously run its self test when there is no data to process.

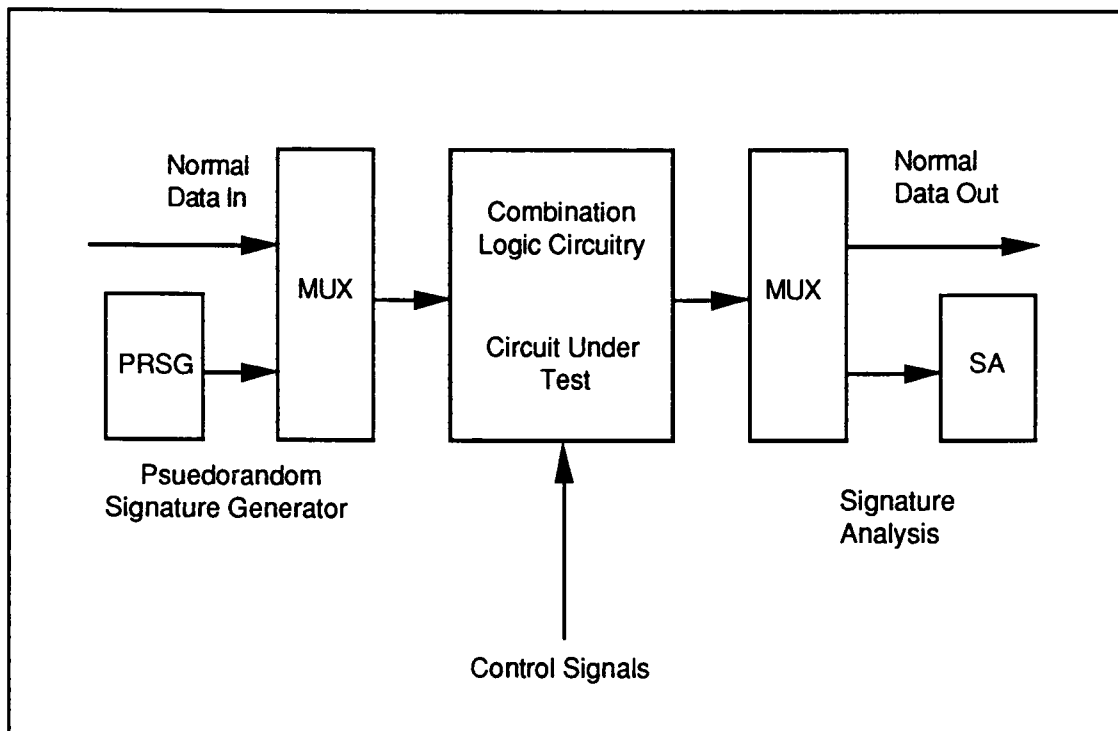


Figure 6.4. Multiplexing Path Built-In Logic.

6.3 SCAN PATH DESIGN

Scan testing uses all the storage nodes in a design by linking them together into one or more shift registers called a scan path. Any intermediate points in the circuit where controllability and observability are needed require that a storage node be added to the design at that point. At any point in the operation of the chip, the information in the scan path can be scanned out of the chip for examination by the designer. In addition, test vectors can be scanned into the scan path. Although it provides a high fault coverage, a scan path requires a synchronous design with no asynchronous circuitry such as static feedback circuits. Scan testing also requires that additional real estate be invested in storage nodes and connection paths.

Scan paths were chosen as the built-in testing scheme for the moment generator because of their low impact on the size of the chip. The registers that are already in the chip architecture for pipelining and accumulation purposes were designed to work as serial shift registers also. This required a low cost in real estate in the cell. Only two transmission gates per register cell were added to operate the parallel register cells as shift registers. These two transmission gates are used to multiplex the parallel and serial input data. Each pipe required the use of an additional control line to switch between the normal parallel operation and serial operation. The location of the two pipes used in the moment generator are shown in the block diagram layout of the chip in Figure 6.5, (white blocks indicate register cells used as scan paths).

The static design of the adder cells as well as the register cells will allow the test vectors used to test the summation function and the multiplier functions to be scan in at a reduced clock rate with no fear of losing data. This is the advantage of using static designs over dynamic designs. A dynamic design would stipulate a minimum clock frequency that would need to be running at all times to keep internal storage nodes in the registers from losing data. The use of the static design simplifies the interface needed to test the chip. The external clock driver for the ASIC can be stopped and singled stepped by a operator activated pushbutton.

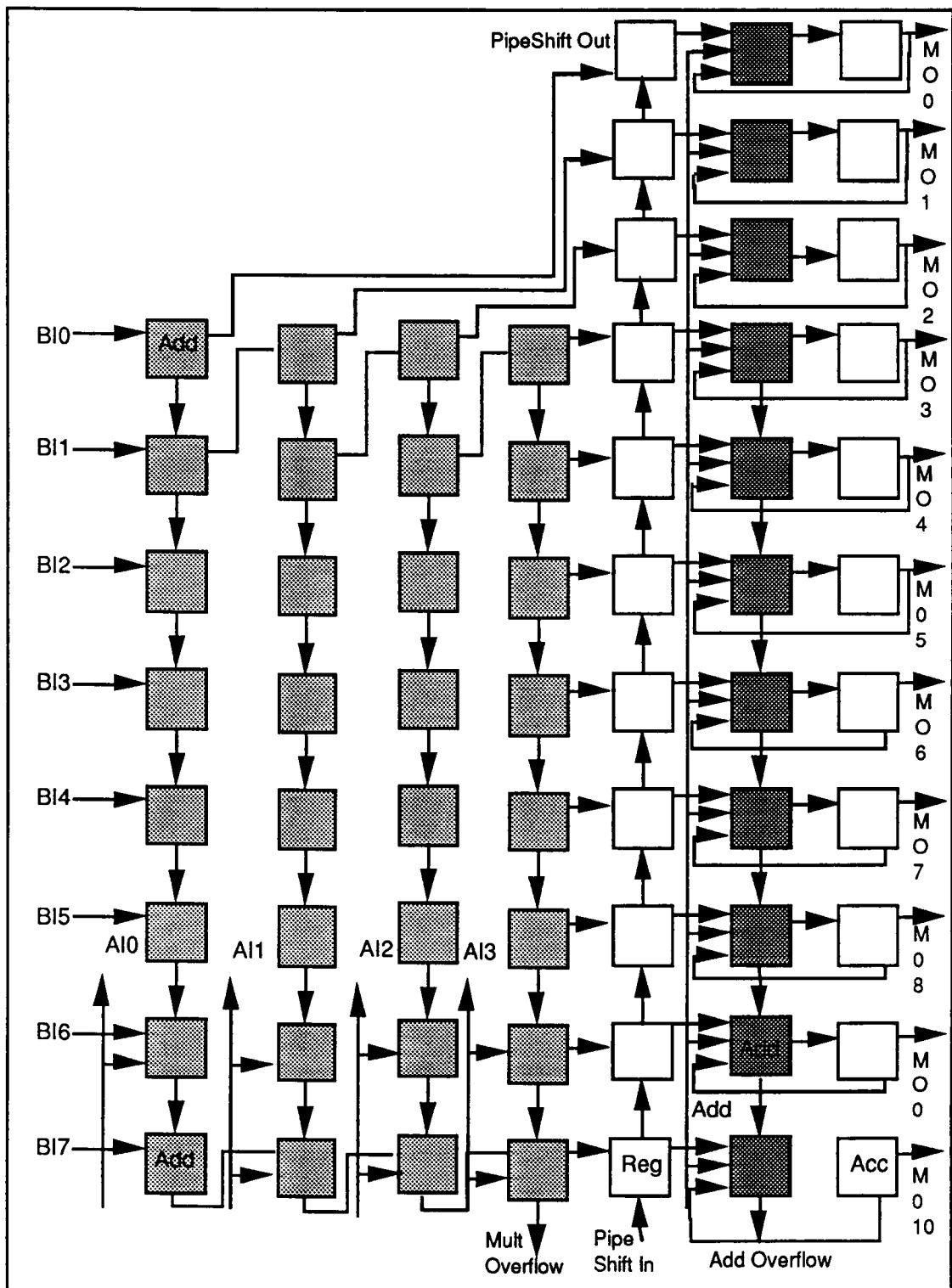


Figure 6.5. Prototype Moment Generator Functional Layout with Scan Paths.

6.4 ADDER CELL DESIGN

6.4.1 CONVENTIONAL ADDER CELLS

The most widely known method of implementing an adder is using two half adders make one full adder. A half adder, as shown in Figure 6.6, is composed of a logical AND gate and a logical EXOR gate. The truth table for the half adder also is shown in Figure 6.6. When used in a multiple bit word, the half adder is not capable of allowing for the effect of adding in a carry from the addition of the two bits just below it

A full adder is composed of two half adders and one logical OR gate and can be used as a bit slice of a larger adder capable of adding multiple bit-wide data words where the carry is allowed to propagate up through the adder slices or cells from the least significant bit to the most significant bit. A illustration of a full adder is shown in Figure 6.7.

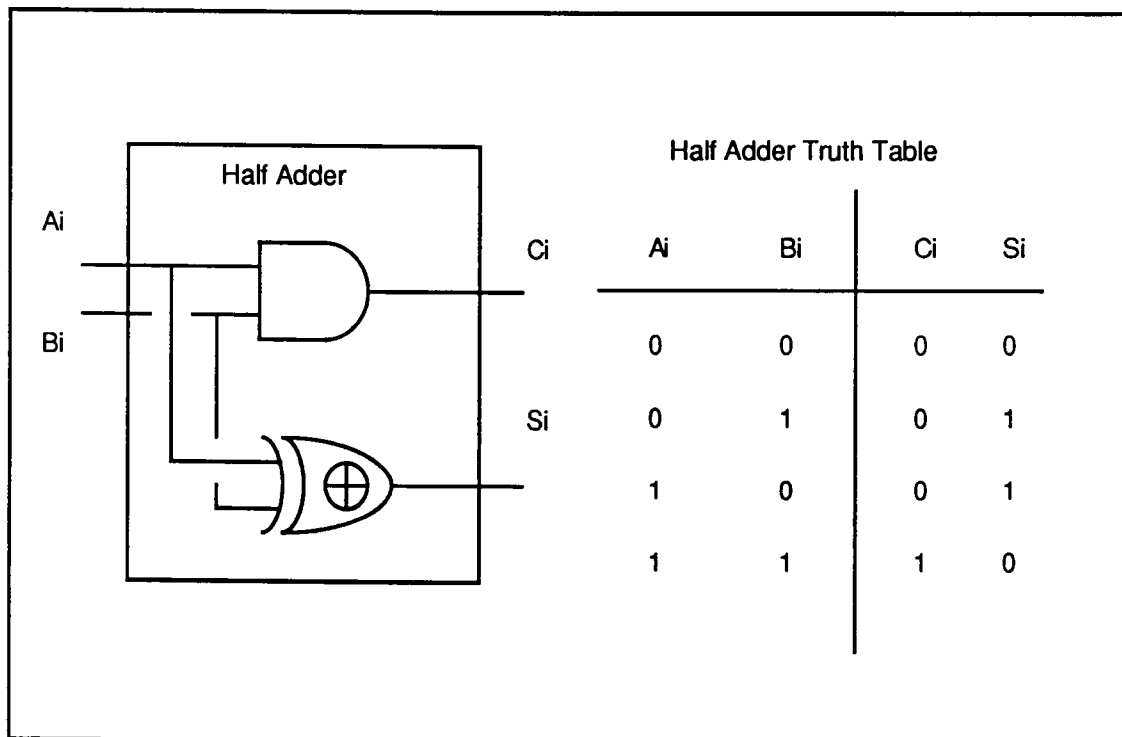


Figure 6.6. Half Adder.

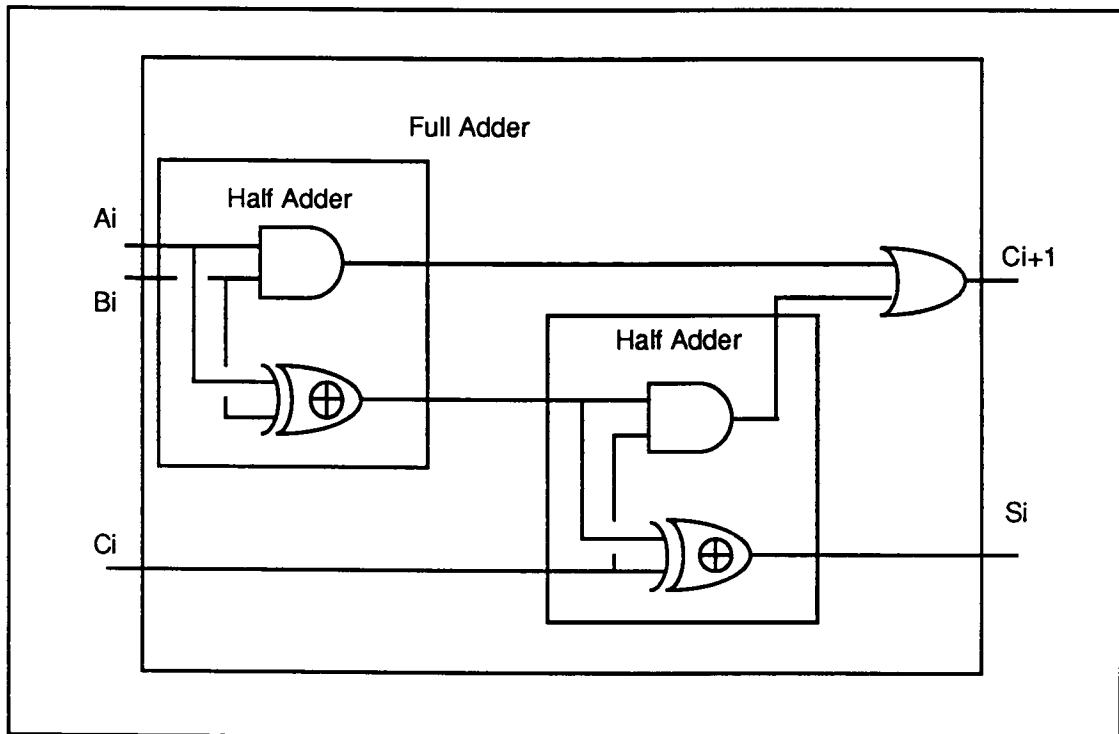


Figure 6.7. Full Adder.

The advantages of this design method are the simplicity of the approach and the need to design three logic gates: an AND gate, an OR gate and an EXOR gate. All of these gates are readily available as standard cells which eliminates the need to do custom design if space and speed are not an issue. If desired, the half and full adders can be implemented completely with NAND gates, further reducing the number of unique cells that need to be designed to implement the adder function. If one cell is used to construct all other functions and building blocks more time can be spent designing and simulating that cell. The disadvantage of this approach is the carry path when used in a carry propagate scheme as discussed in Section 5.2. A carry condition in the least significant bit must be allowed to propagate from the least significant bit to the most significant bit. For the full adder shown in Figure 6.7, the carry has to propagate through two logical gates in each bit of the adder.

Carry look-ahead adders can be used to decrease the time required for the carry signal to propagate through the adder. This scheme involves computing carries for several sets of bits rather than just one pair at a time. The size of the grouping can be optimized for the

application but usually is considered optimum for groups of four, especially if buffers are not included in the carry path. This approach has the disadvantage of adding a significant amount of circuitry to the design. The adder cells can be designed with a method similar to that used to implement the look-ahead adder but on a two bit basis rather than four or more. This is the basis for the Manchester carry chain conditions.[5]

6.4.2 MANCHESTER CARRY CONDITIONS

Manchester carry conditions concentrates on minimizing the propagation time through the adder cells by studying the adder function solely from the carry signal viewpoint. Given two words, A and B, that must be added together to produce a sum, any two-bit adder cell within the adder bank can have a carry-in signal or a carry-out signal. The conditions of the input operand bits A_i and B_i can have effects on the carry out signal that are completely independent of the carry-in signal. Equations (88) through (90) summarize these conditions.

$$K_i = \overline{A_i B_i} \text{ (Kill Condition) ;} \quad (88)$$

$$G_i = A_i B_i \text{ (Generate Condition) ;} \quad (89)$$

and

$$P_i = A_i \oplus B_i \text{ (Propagate Condition) .} \quad (90)$$

Given that the two input data bits to the adder, A_i and B_i , are both low, there will be no carry signal out of the cell independent of whether the carry in signal was high or low. If both the input data bits, A_i and B_i , are high the carry out signal will be generated by the adder cell independent of the present value of the carry in signal. For the two remaining input vectors the carry out signal will depend only on the state of the carry in signal. In these two states, either A_i or B_i is active high but not both. These two states are detected with a EXOR function as shown in equation (90). This creates the propagate condition were the carry in signal is propagated to the carry out port of the adder cell. The summation is expressed as the result of the exclusive-or of the two operand bits A_i and B_i along with the carry-in signal in an analogous manner to the way summation is performed in the full adder.

$$S_i = P_i \oplus C_{i-1} \text{ (Summation) .} \quad (91)$$

The typical implementation of the Manchester carry chain is shown in Figure 6.8. The carry chain is the critical path and is precharged. The chain shown in the figure is charged high to a logical 1 (5 volts) on the high phase of the clock, and the carry condition is allowed to propagate through the path during the low phase of the clock.

Precharged nodes or buses are very sensitive to any form of noise in the chip. Noise can be generated by perturbation in the power-line voltage because of intermittent demands for an excessive amount of current that may be larger than the metal runs can sink or source or larger than the power supply can handle. Noise also can occur in the form of crosstalk between adjacent lines, making it important not to have two long runs in parallel. After a voltage transition, noise also can occur in the form of oscillations or ringing caused by improperly matched impedances between the driver and the load. Signal skew at the input also can destroy a precharged value by preventing the correct signal from reaching the correct level in the desired setup time thereby causing the outputs of a combinational circuit to bounce from one level to another. A large amount of oscillation noise or crosstalk can remove the charge from the line causing an intermittent error condition. Any bouncing caused by

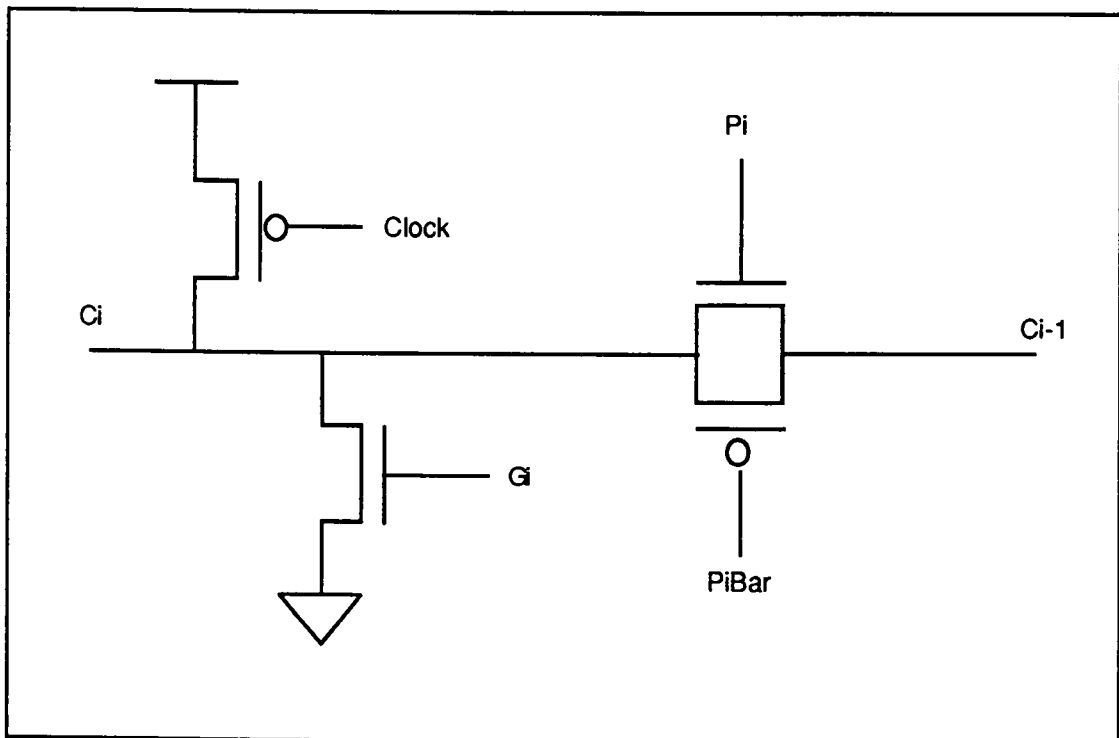


Figure 6.8. Precharged Carry Line.

signal skew must be allowed to stabilize before the results are applied to the precharged node. The high phase of the clock must be longer than the maximum time required for the carry conditions in each cell to stabilize. If the conditions are still bouncing or ringing when the clock goes low, the charge could be removed from the line causing an intermittent error condition.

The circuitry used to implement the carry path in the adder cell was based on a static implementation of the carry chain and is illustrated in Figure 6.9. Each carry-in signal was buffered from the proceeding cell by two CMOS inverters. This was to maintain the goal of keeping all logic external to the cells positive. One inverter would have reduced the speed and still provided buffering of the carry signal in each adder but this would require the design and simulation of an odd and even adder pair. This prevents any ringing in the present cell from interfering with the effects of the carry conditions from the previous cell. One transmission gate was inserted into the path to implement the propagate condition. The output of the kill and generate function is isolated from the carry chain by another transmission gate. These two transmission gates were included to obey the consistency rule described in Section 6.4.3. Both transmission gates are controlled by the propagate signal and its complement.

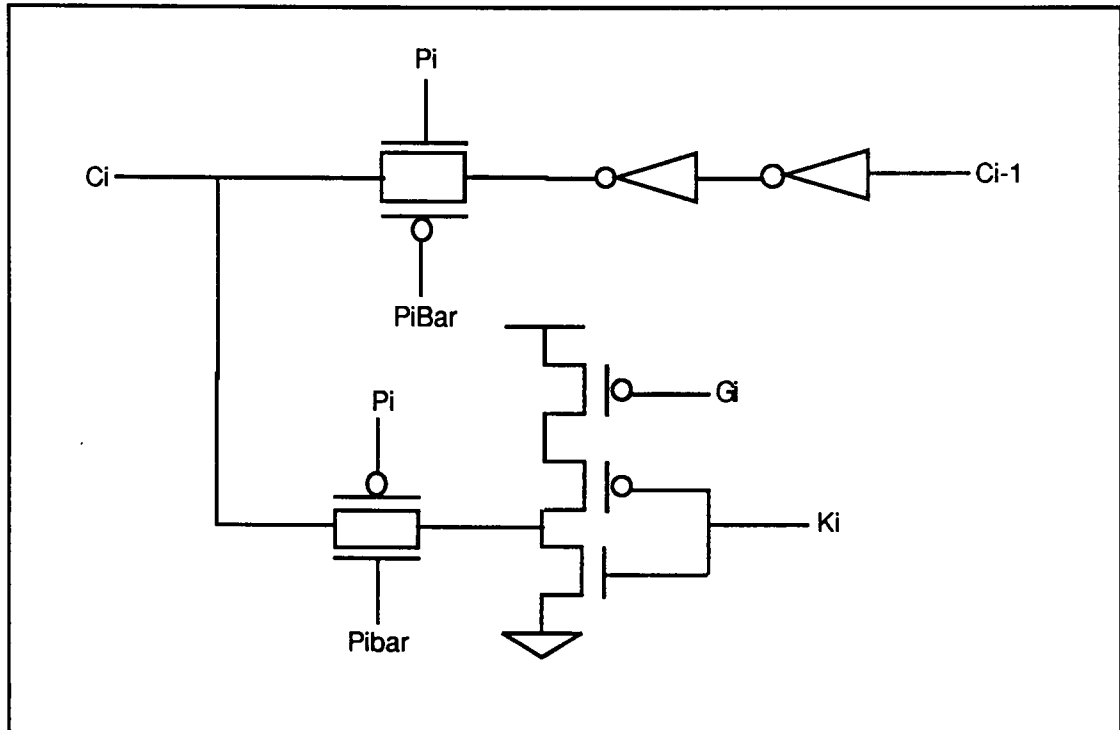


Figure 6.9. Static Carry Chain.

6.4.3 MANCHESTER CARRY CONDITIONS ADDER

DESIGN

The adder cell design was performed by providing one logical function for each of the three Manchester carry conditions, one logical function for the summation, and circuitry to support the carry chain. As the individual logic functions were designed, ESIM was used to verify logical functionality. The two EXOR gates used to implement the propagate condition and the summation for each cell could not be simulated by ESIM, so Pspice was used. Once all logical functions were designed and simulated, they were integrated into one adder cell. The initial design used the minimum number of gates to implement each separate function. Integration of the functions into one cell introduced violations of the consistency rule and possible loading problems for the "SumIn" and the "SumOut" nodes.

SumIn had seven loads to drive that included the EXOR function. "SumIn" was required to drive the source pFET in the output stage of the EXOR. As a result, two buffers were added, one to drive loads in the EXOR function and one to drive the other five loads dependent upon SumIn. Through simulations with Pspice, the "SumOut" signal also was determined to have an excessive transitions time (7 to 8 nanoseconds). CMOS circuits have a tendency to bounce when passing through the transition region, and as a result transition times should be kept as short as possible. To increase the transition time of the "SumOut" signal and to prevent the possibility that the signal could be loaded by external sources, two drivers were added to the "SumOut" signal; the resultant signal is called "SumOutBuf." The resultant adder cell design is illustrated in Figure 6.10.

Adding the buffers to "SumIn" and "SumOut" made the adder circuit highly resilient to changes in loads external to the cell. The buffers also increased the time required for "SumIn" to propagate through the cell to "SumOut" but reduced the time required to produce the carry propagate condition, "Pi," allowing the carry to propagate faster through the cell. The addition of the buffers also increased the size of the adder cell. Table 6.3 summarizes the effects of adding the buffers to the circuit.

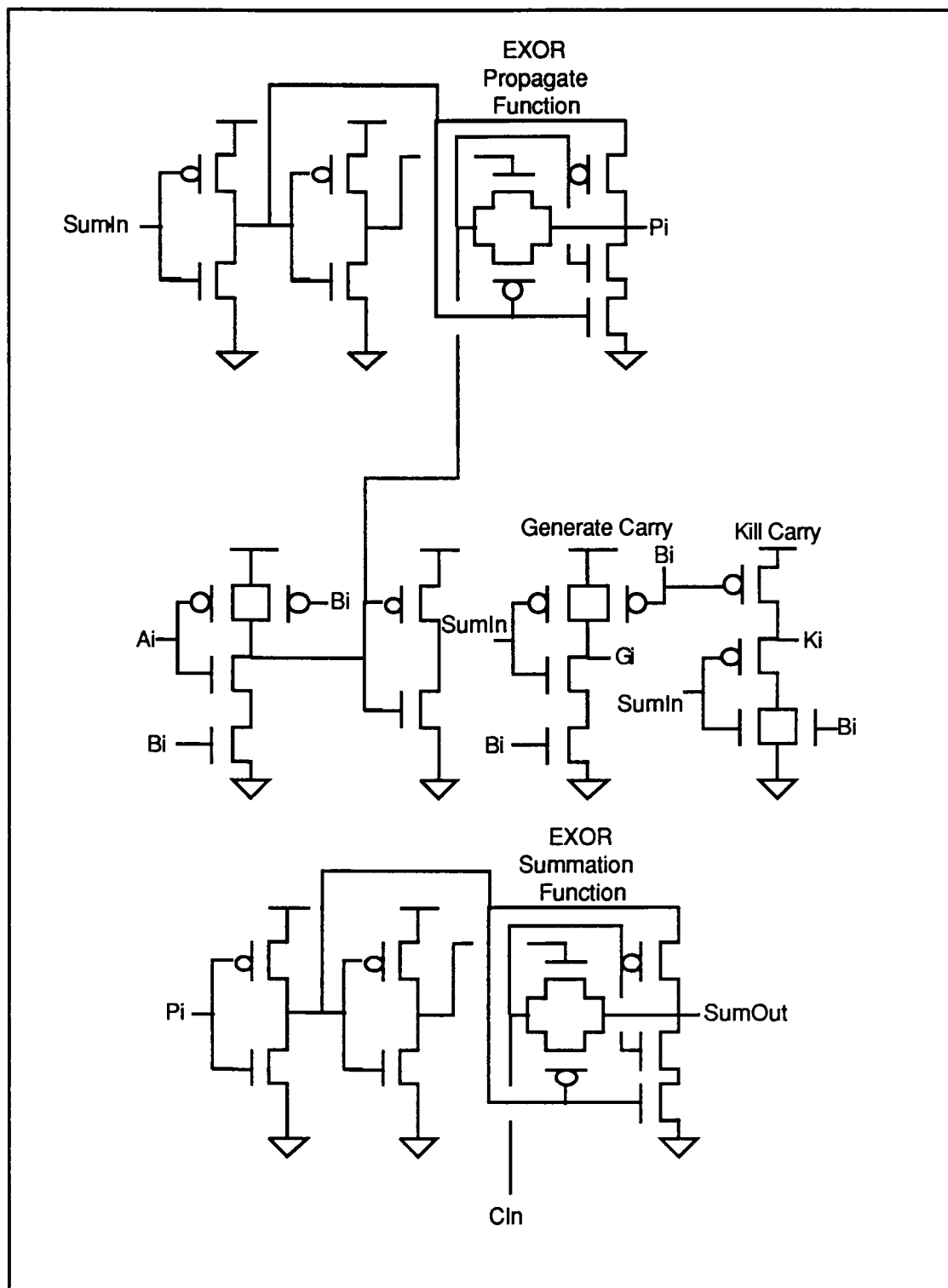


Figure 6.10. Manchester Carry Conditions Adder Cell.

Table 6.3. Adder Cell Buffer Addition Effects.

Adder Cell	Size	SumOut Propagate Time	PI Propagate Time
Without Buffers	132X150	6	5
With Buffers	132X212	8	4.5

After examining the loads on each signal used in the adder cell, the cell was examined for violations in the consistency rule. The consistency rule states that no two sources (i.e., inverters) are allowed to drive the same node at the same time. Ignoring this rule can produce points in the circuit that momentarily attempt to connect VDD and ground together, destroying the chip. Violations of the consistency rule were detected by examining the drain current levels for the transistors generated as output data from the Pspice simulation. The problem was indicated by excessive levels of drain current, which jumped from its usual 3 to 4 nanoamperes to 1 to 2 milliamperes.

6.5 MULTIPLIER DESIGN

Arrays are well suited to VLSI design. A design that can incorporate an array-based architecture satisfies two of the main criteria for using VLSI. First, the cells should be modular and require a great deal of replication; second, the interconnection between the cells should be regular. The architecture used to implement the multiplier is based on the two-dimensional orthogonal array described in Section 5.2. The multiplier array is shown in Figure 6.5. This section describes how the two dimensional orthogonal array is used to construct a multiplier function.

Given two M-bit numbers A and B, the product of these numbers can be represented as the partial product shown in equation (92).

$$X_{i+1} = X_i + A_i B^i ,$$

$$P = X_M .$$
(92)

A_i is bit i in the M -bit binary operand A . This implementation allows the multiply operation to be performed in a series of adders banks. Each bank of adders is composed of adder cells. In the $i+1$ row of the array, B is shifted by i bits and added to the partial result from the previous row (X_i) if $A_i = 1$.

Note that at this point in the design, this implementation allows the designer to implement a variety of adder cells and adder banks. The designer can choose standard cells, full adders, or Manchester carry-chain-based adders and incorporate them as carry propagate cells or (with some modification) carry save cells. The block diagram of a multiplier array is shown in Figure 5.5. The array incorporates a carry propagate adder approach in each row.

M carries can occur in each row. The time critical path in the multiplier array is through the carry propagate path in the first row and the N adders in the last column. The length of the critical path is two $M+N$ cells. The functional description of each block is given in Figure 6.11. Each block is composed of a Manchester carry condition adder cell and a logical AND gate, which determines where that particular row in the array will add its inputs based on the value of A_i .

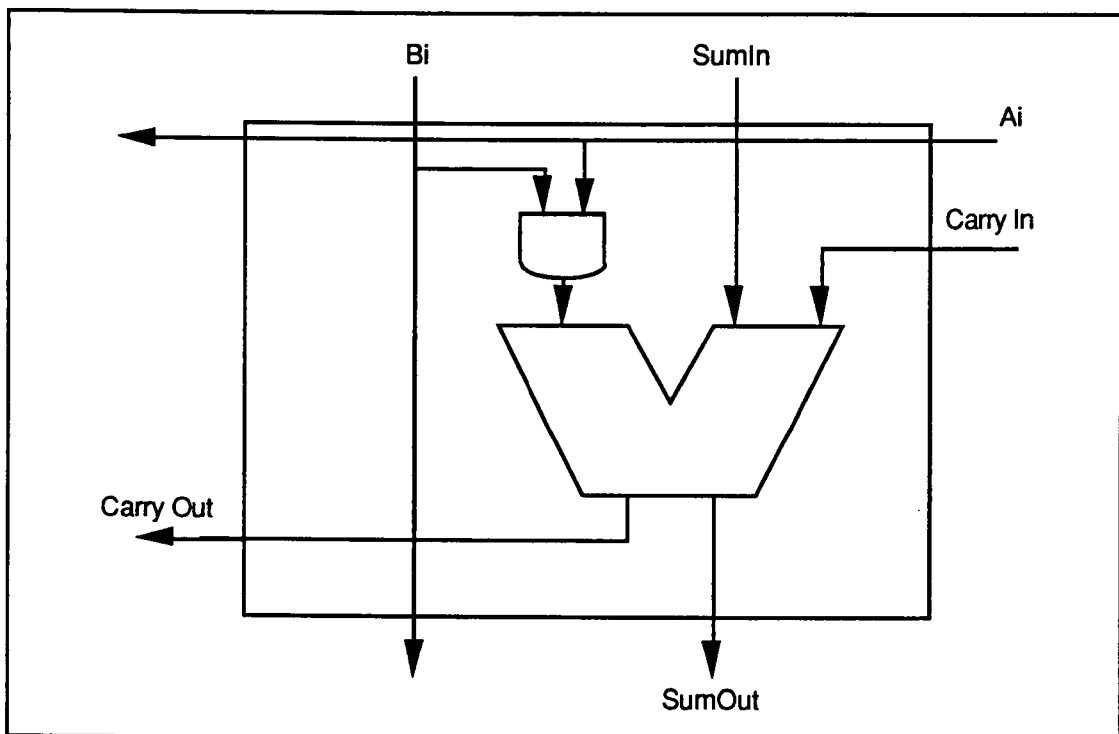


Figure 6.11. Adder Cell Functional Diagram.

6.6 REGISTER CELL DESIGN

To pipeline the multiplier function from the summation function a register was inserted between them. A register also was needed to implement the accumulator used to store the summation results. The registers used to implement the pipe and the accumulator were also designed for use as scan paths for testing the chip. To allow the registers to load, store, and output parallel data and be used as a scan path, both parallel and serial capability were designed into each cell. Several standard methods for designing register cells are discussed in the literature and this section discusses the design technique selection for implementing the dual port register cell used in the moment generator chip and the design of the cell.

The two main categories of register cells are static cells and dynamic cells. In static cells, stored information is due to the conduction or lack of conduction of a transistor. Static cells consume more power than dynamic cells but can hold information for long periods of time. Dynamic cells store information in a charge at an internal node by use of the node capacitance. The charge must be refreshed frequently, or it will leak off the node; and because CMOS as is scaled to smaller and smaller sizes, the amount of time required for a charge to leak off a node has decreased to a few 10 MHz clock periods.^[12] Dynamic cells take considerably less power to operate and smaller space to implement than static cells but are also very susceptible noise. Because dynamic devices require a periodic refresh, they must be operated at a minimum frequency. This can make testing the chip during debugging or checkout very difficult when single stepping the clock is necessary.

Static cells are implemented through the use of asynchronous cross-couple circuits (such as an RS latch), clocked cross-coupled latches, and cross-coupled inverters. Semistatic register cells are dynamic memory cells in which the node is refreshed every clock cycle. Semistatic cells require a large amount of circuitry to implement and still have the disadvantage of dynamic nodes in that they are susceptible to noise. Because of the presence of a synchronous feedback path, the circuit requires the use of a two-phase non-overlapping clock circuit. Pure static memory cells violate the consistency rule, which is often alleviated by the strategic placement of transmission gates but requires no clocking circuit to load or store data.

Asynchronous static cells composed of cross-coupled inverters were selected to implement the register cells. The cross-coupled inverter design is the same design used to implement a RS latch which is the basic building block for many static storage nodes such as D flip-flops, T flip-flops, and master/slave pulse triggered JK flip-flops. These cells can be used without dynamic nodes or synchronous feedback paths which eliminated the need for a two phase nonoverlapping clocking scheme. A single-phase complementary clock was used to integrate the two-half bit register cells into one register cell so that data could be latched into each half bit cell synchronously. As illustrated in Figure 6.12, two transmission gates are used to select between the serial and the parallel modes of operation. Two more transmission gates are used in each half-bit cell to clock data into the cross-coupled inverter. These transmission gates insured that the storage nodes were isolated from the multiplier outputs until the outputs had sufficient time to stabilize. The full-bitcell is composed of two half-bit cells. Each half register cell uses a cross-coupled inverter to statically store the data. All other circuitries are support functions for synchronously loading the data and selecting between parallel and serial operation.

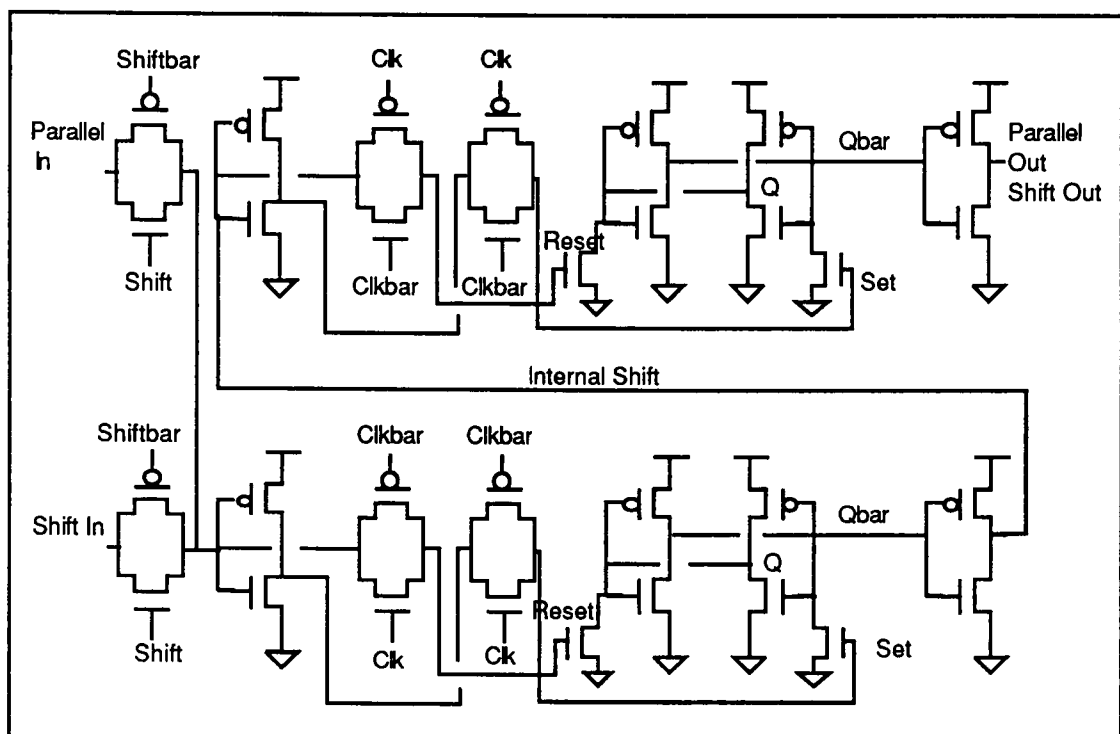


Figure 6.12. Static Register Cell Designed with Scan Path Capability.

6.6.1 STATIC REGISTER CELL AMPLIFICATION FACTOR

TEST

The static cells violate the consistency rule by the very nature of their function. As shown in Figure 6.12, the cross-coupled inverters have to be forced from one storage state to the next by forcing one of the inverter outputs to a low value. If the inverter output is high when this occurs, the pull-down transistor and the pullup transistor fight for control of the node. The larger of the two devices will be the winner. To allow the pull-down transistor to overcome the pullup transistor, the length of the pullup transistor was set to 4 lambda, and the width was reduced to 2 lambda. The gain of the closed-loop portion of the static cell used to implement a static storage register must be greater than unity to prevent the cell from getting stuck in the metastable state and to prevent the transition from one state to the next from happening too slowly. The amplification factor can be determined from the transistor specifications or experimentally. Using the Pspice dc sweep analysis, the amplification factor can be determined by measuring the slope of the Vout versus the Vin graph. This is the method discussed and documented here. The first half-bit storage cell was tested first by sweeping the node "FFQ1" with an input linear dc voltage from 0 to 5 volts in 0.1 volt increments. The amplification factor was approximately 1.5 through the metastable region. Several other test runs gave similar results, as shown in Table 6.4.

Table 6.4 Amplification Factor Test

Test	Filename	Gain in Transition Region	Sweeping Direction
1	pmccAmpTest1.plt	u= 1.5	FFQ1 from 0 to 5 volts
2	pmccAmpTest2.plt	u= 1.5	FFQ1 from 5 to 0 volts
3	pmccAmpTest3.plt	u= 1.5	FFQ2 from 0 to 5 volts
4	pmccAmpTest4.plt	u= 1.5	FFQ2 from 5 to 0 volts
5	pmccAmpTest5.plt	u= 1.5	FFQ2bar from 0 to 5 volts

Note that sweeping from 0 to 5 volts yielded the same results as sweeping from 5 to 0 volts. This probably should have been obvious from the geometric symmetry of the devices. Insufficient levels of amplification factors were detected when sweeping FFQ2bar from 0 to 5 volts and monitoring the voltage at FFQ2. The amplification factor was 1.5 for V_{in} in the range of 0 to 0.7 and then went to less than 0.5 for V_{in} greater than 0.7 volts. Increasing the gate region length on the pullup transistors by a factor of 2 made the amplification factor less than 1 for the entire range of the sweep and the output voltage never got above 0.5 volts. This agrees with the equations (93) through (97), which describe the amplification factor.

$$\mu = \frac{\Delta V_{ds}}{\Delta V_{gs}} . \quad (93)$$

$$\mu = g_m r_{ds} . \quad (94)$$

$$g_m = \frac{\Delta I_{ds}}{\Delta V_{gs}} , \quad V_{ds}=C . \quad (95)$$

$$I_{ds} = \beta(V_{gs}-V_t)V_{ds} . \quad (96)$$

$$\beta = \frac{\mu_n \epsilon}{t_{ox}} \left(\frac{W}{L} \right) . \quad (97)$$

Increasing the length of the diffusion channel decreases the process gain as described in equation (96). This in turn decreases the drain current in equation (95), the transconductance in equation (94), and the amplification factor in equation (93). Decreasing the gate region of the pFETs to $2X2 \lambda^2$ increase the amplification factor to 1.5 for dc sweeps from either direction on both static storage cells.

6.7 CLOCKING SCHEME

The selection of a clocking scheme is one of the most important initial decisions in the design of a VLSI ASIC. Complimentary single-phase clocking and two-phase non-overlapping clocking schemes are illustrated in Figure 6.13. The clocking scheme depends on the type of circuitry used in the ASIC. First time designs often use a two-phase clocking scheme, which gives the designer the ability to control the gaps between clock phases. This can be used to counteract erroneous operation caused by clock skew.

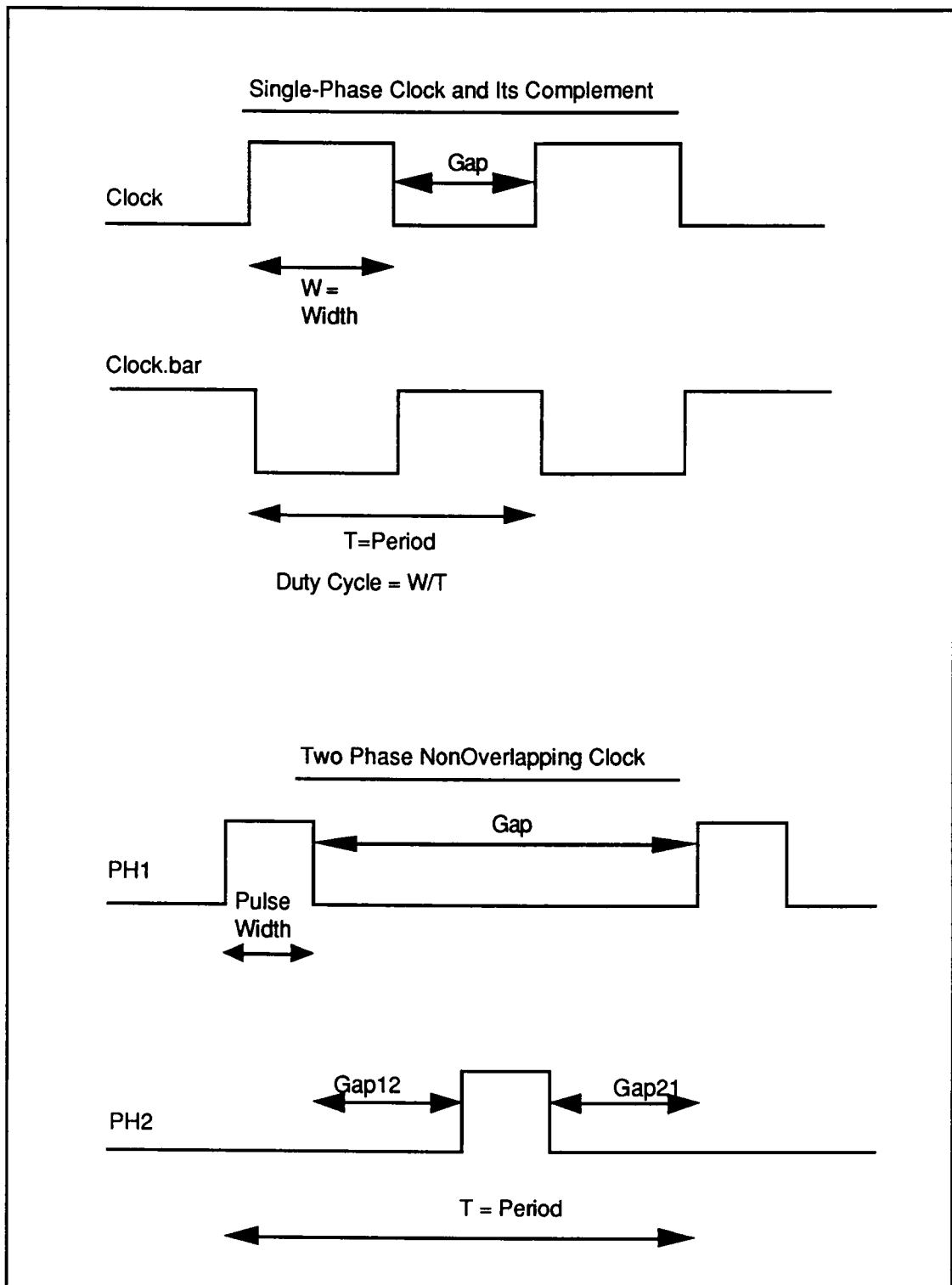


Figure 6.13. Single-Phase and Two-Phase Clocking Schemes.

Single-phase, complementary single-phase, and non-overlapping two-phase clocks are commonly used in CMOS designs. Single-phase clocking has tight timing constraints that must be met when used in circuits with sequential state machines. If the timing constraints are not met, multistepping, or a race condition can occur. Theoretically the race condition can be eliminated through the use of a single-phase clock and its complement. This divides the closed loop of the storage node composed of the feedforward stage and the feedback path into two stages that will never be on at the same time. This is not always the case in the actual circuit because the clocking signals can be skewed. There are two sources for signal skewing. The first source is random propagation delay through combinational logic in the signal path or simply varying lengths of signal paths. Signal delay is directly proportional to the square of the path length. The second form of skew results from long transit times that allow the clock and its complement to be active simultaneously. In both cases, clock skew can permit the two stages to be on at the same time causing a race condition.

Because of these problems CMOS designers usually choose two-phase clocking schemes. Two-phase clocking takes up more area on the circuit but can be used to prevent race conditions in any circuit with a clocked feedback path such as dynamic storage nodes or sequential state machines. The two-phase clock can be used in conjunction with two half-stage registers to ensure complete isolation on the feedback path.

The adder cell used in the moment generator ASIC is completely static and uses no clocks. The register cell is composed of two static storage nodes that are clocked. The clock, however, is not used to control the transmission gate of a clocked feedback path. The feedback path, as well as the feedforward path, is completely static. As a result, the circuit is race condition immune. The single-phase clocking scheme was chosen as the clocking scheme of the moment generator ASIC.

The time period of the single phase clocking scheme is determined by the function in the pipelined data path with the worst case propagation delay. The delays of the multiplier function and the summation function are determined by simulation and are discussed in Chapter 7.

CHAPTER 7

SIMULATION AND TESTING

The final step in the design of a VLSI ASIC, before sending the design to be fabricated, is simulation. Simulation allows the design engineer to develop a level of confidence that the design will perform as intended. After the ASIC is returned from the manufacturer, the physical chip is tested .

Simulation is performed with logic simulators, fault simulators, timing simulators and analog simulators. With the possibility of 10,000 to 100,000 transistors on a single ASIC, there is no time to test every possible combination of states in which the circuits can exist. The designers of the chip must a set of test methods and vectors that will allow the greatest coverage of the chip in a feasible amount of time.

The logic simulators and fault simulators are the easiest and fastest to use. These simulators test the circuits on a logic or functional block level. The timing and analog simulators often model the individual circuits in the ASIC and approximate or predict changes in the waveforms at each node in the circuit. These simulators take a great deal of time and effort to set up and run because of the large amount of mathematical analysis required. The timing and analog simulations usually are performed only after the logic and fault simulators have tested the design thoroughly and proven that it is working and has a satisfactory degree of fault coverage.

After computer simulations have verified repeatedly that the design is working properly, the drawing is mailed electronically to a silicon foundry participating in the MOSIS program to be fabricated. The fabrication process takes about two weeks. The chip then is tested with special purpose ASIC test equipment (ATE) or simply by plugging it in to the electronic interface board where it is designed to operate. If built-in testing techniques were incorporated during the design of the chip, these circuits can be used to help test subsections of the chip.

7.1 SIMULATION

7.1.1 DESIGN RULE CHECKERS (DRC)

AND CIRCUIT EXTRACTORS

The first forms of simulation done on VLSI designs usually are done through the DRC in the graphical layout editor by extracting a network list from the raster image of the California Intermediate Format (CIF) files. Magic^[11] is the graphical layout editor used to design the custom cells for the moment generator ASIC. Magic has a built-in DRC with a rule base that describes geometric relationships between layers in the design. Because Magic uses abstract layers to design the circuits, extraction is simple. This allows the Magic editor to draw quickly the circuit on the screen and store the images of the circuit in a compressed form, saving disk space on the computer and reducing the time needed to store and retrieve files. The Magic testing method uses an incremental hierarchical DRC. This allows the examination of circuits and reporting of errors while the circuit is being designed. In addition, errors found in hierarchical cells are reported only once. This prevents the designer from being overwhelmed with error messages. This type of examination is more advantageous than using the extractor for error checking because the user gets instantaneous feedback to bad design practices and then can determine intuitively the nature of the problem without an exhaustive knowledge of the design rules. The disadvantage of this approach is that errors detected by other types of extractors are not detected until a detailed continuous-timing simulation (usually Pspice) is employed to test the circuit. Circuit extractors can provide information about nodes that are floating (less than two connections), nodes that are tied to a fixed voltage such as power or ground, or nodes that short power and ground together. Circuit extractors also can determine voltage thresholds approximations by examining the length-to-width ratios of the transistors and comparing the approximations to acceptable thresholds. These checks provide essential debugging information early in the design before the circuits grow to the point that determining the source of these errors becomes prohibitively difficult.

7.1.2 ESIM THE EVENT SIMULATOR

Most logic function inputs are based on asynchronous events rather than the synchronous inputs. As a result, most logic simulators such as ESIM are event driven. Logic simulators are good for well-structured circuits that have regular logic counterparts. Most logic simulators have trouble with nonstandard circuits such as the static memory cells in the register cells of the moment generator as well as the logical EXOR gates used to implement the adder cells.

ESIM can use the names assigned to nodes in the circuit with the label utility in the Magic graphical editor. This saves time since the setup files for running ESIM require no modification before each test run. ESIM also has three states for the switches that it simulates and uses gate-size and capacitance information from the circuit extractor output to do simulations. Moreover, ESIM allows the designer to select all subnets that correspond to simple logic gates and the conversion of them into a form allowing faster simulation (25 to 50% in some cases). ESIM also provides diagnostic capability that allows the designer to find design errors in the circuitry such as oscillations.

Event-based logic simulators such as ESIM react to events or state changes at nodes in the circuit that are usually caused by transitions of the input data or control signals. The simulators use simplified models of the circuits, usually consisting of resistance and capacitance to determine the transition to the next stable voltage level (and the time required to transition). The simulator produces a list of the nodes affected by each transition and works its way through the circuits until all nodes reach stable values. This methodology causes real problems with circuits that have bidirectional information flow such as transmission gates. In these cases, ESIM can go into an infinite loop, and circuits that actually are correct are determined to be oscillating or nonfunctional.

While ESIM was used extensively in early cells designs evaluated for use in the moment chip, ESIM never was able to converge to a result on either of the two cells used in the moment generator design. Both cells used irregular designs to implement internal functions. The adder cell design was based on the Manchester carry chain conditions. As a result, the adder cell contains two EXOR gates. Each half-bit register in the register cell contained one static storage function. The half-bit static register cell was composed of two

inverters in a closed-loop to from the memory component. ESIM was not able to simulate the EXOR gates or the closed-loop inverter designs.

7.1.3 COMPILED LOGIC AND FAULT SIMULATOR FOR MOS CIRCUITS (COSMOS)

COSMOS^[2] is a set of programs that transforms a VLSI circuit into a simulation program written in the C programming language. The programs that make up COSMOS are illustrated in Figure 7.1. The sim2ntk program generates a switch level network based on the transistor network list extracted from the Magic file. Note that the switches presevre no information as to the direction of current flow in the transistors from which they were derived. As a result, information can flow in either direction. This requires that special labels be added to circuits such as transmission gates and exclusive-or gates to prevent the resultant simulation file from getting stuck in loop as if the circuit were oscillating. The network list of switches is passed to ANAMOS which analyzes the switch network and generates a boolean

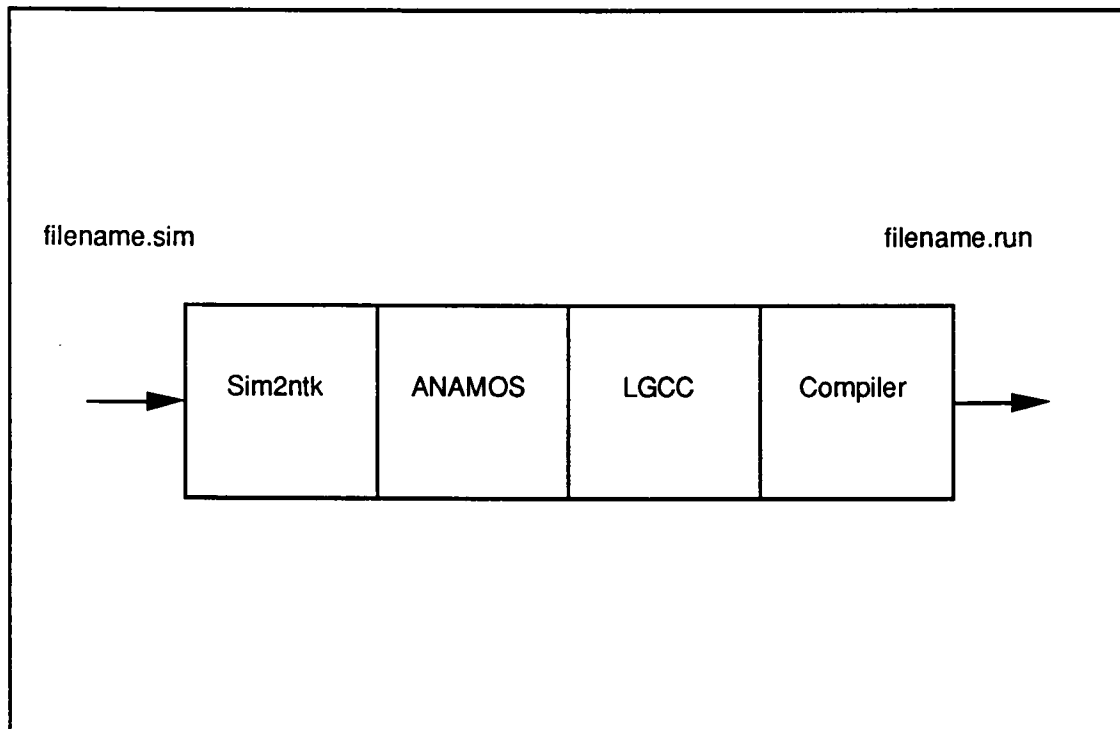


Figure 7.1. COSMOS Test Method Block Diagram.

description of the circuit. The program LGCC, converts the logical boolean function into C source code. The computer operating system's C compiler and linker are used to compile the code into executable modules and to link these modules with other object modules from either the logic or fault simulator libraries to produce a program for logic simulation or fault simulation .

As previously mentioned, COSMOS does not account for the diode action of the transistors when building switching models and therefore tends to have problems with bidirectional information flow; COSMOS does provide; however, a three-state model of the transistor composed of 0, 1, and X (unknown) states. The unknown state is defined as current flow through the switch that is between the cutoff and saturation regions. When a node goes to an unknown state, all nodes that derive their states from that node go to unknown states also. In addition, COSMOS stores no information pertaining to transistor strength or node size. As a result, COSMOS simulation requires nodes in the circuit to be specifically connotated as to strength, information direction flow, and storage capability. Using these special Magic labels gives the designer insight into whether or not the physical and geometric relationships in the design of the circuits transistors are sufficient for their intended jobs.

7.1.4 COSMOS LOGICAL SIMULATOR TEST OF THE MOMENT GENERATOR

COSMOS Logic Simulator Version 1.2 first was used to test the summation function of the moment generator for correct logical operation. The "AdderCntrl" had to be set to zero (off) and all multiplier inputs to zero (off or low) to set the outputs of the accumulator register to zero. This was done to ensure that the data path could feed zero values from the input to the output. Once the summation function accumulator was zeroed, the AdderCntrl was set to 1, allowing the summation function to add the results of the multiplication function to the contents of the accumulator. The multiplier multiplied 1X1 and allowed a result (or product term) of 1 to be fed to the second pipe stage, which is the summation function. This allowed the adder to be observed incrementing its way to 256, one increment at a time. This test

indicated that the summation function could add properly and that the accumulator also was functioning properly.

After ensuring that the summation function was correct logically the multiplier function was tested. The multiplier is a static design with no clocks for control. As soon as the two operands are present at the multiplier inputs, the multiplier function begins its operation. The output of the multiplier is latched into the pipe register that follows the multiplier section. Data are latched into the pipe register during the high pulse of the clock and are output for observation on the following low clock. The observation points here are the outputs from the first pipe register labeled "R_i," where *i* varies from 0 to 10.

7.1.5 PSPICE THE ANALOG SIMULATOR

The basis for Pspice is a matrix of equations relating currents, voltages, and impedances through nodal analysis and backward integration prediction algorithms. Pspice uses a detailed model of the transistor in which values such as threshold voltage, supply voltage, temperature, and mobilities can be varied among the models. Pspice has accurate simulations capabilities but requires a great deal of time to set up and simulate circuits that have more than 10 to 15 transistors. In general, Pspice is used to simulate simple or small circuits used as building blocks for larger blocks or functions. Pspice is extremely thorough in testing of the VSLI circuits and has become the authorization for whether or not a circuit will work. For this reason, Pspice was used several times as a final step to verify operation of the design, even though it took a great deal of time to set up and simulate the moment generator circuit that consisted of 2801 transistors without the I/O pads (4399 with the I/O pads).

Pspice allows the designer to examine timing relationships between signals in the design. Tables 7.1 and 7.2 summarize the timing relationships of the Manchester carry conditions adder cell and the static register cell. Note that the propagation time from the inputs to the output summation node of the adder cell is 9.5 nanoseconds. The 9.5 nanosecond propagation time is caused by the propagation of the SumIn to the SumOutBuf node. Between these two points exist two EXOR gates that both take approximately 4.5 nanoseconds of propagation time. The carry signal will take 2.5 nanoseconds only to propagate through an adder cell. This is the result of having two inverters and one transmission gate only in the carry chain.

Table 7.1. Manchester Carry Conditions Adder Cell Propagation Time Chart.

		PI	PIbar	SumOut	SumOutBar	COut
		[ns]	[ns]	[ns]	[ns]	[ns]
Bi	tpLH	4.5	6.0	7.5	9.5	8.0
	tpHL	2.5	3.5	5.0	6.0	5.0
SumIn	tpLH	2.5	3.5	6.5	8.0	-
	tpHL	2.0	3.0	4.5	6.0	-
Cin	tpLH	-	-	1.5	3.0	2.5
	tpHL	-	-	1.5	3.5	2.5

Table 7.2 lists the propagation times for the static register cell signals. This cell is a clocked device, so most of the signals are measured with respect to the clock that caused the transition once the input necessary to cause a transition was present. Note that the worst case propagation time is 2.0 nanoseconds, which would allow the register cell to operate correctly with a 100 to 120 MHz internal clock if necessary. The adder cell has the dominating propagation delay, and its use in the multiplier will dictate the maximum internal clocking frequency.

Table 7. 2 Static Register Cell Propagation Time Chart

		FFQ1	FFQ1bar	FFQ2	FFQ2bar	ParOut
		[ns]	[ns]	[ns]	[ns]	[ns]
ParIn	tpLH	2.0	1.5	1.0	0.8	0.9
	tpHL	1.0	2.0	0.5	1.0	1.0
ShftIn	tpLH	2.0	1.5	1.5	0.5	1.0
	tpHL	1.0	2.0	0.5	1.0	1.0

The worst case propagation delay through the multiplier can be calculated by multiplying the carry propagation time delay of 2.5 nanoseconds by 8, and adding that to the product of the SumIn to SumOutBuf propagation delay of 9.5 nanoseconds multiplied by 3. This makes the worst case propagation delay through the multiplier 48.5 nanoseconds. The multiplier then also can be clocked at an internal clock rate of 100 to 120 MHz. The limiting time element remaining, therefore, is the speed with which data can be clocked into the ASIC. With the TTL I/O pads of the ASIC, the input data rates will be limited to 10 to 20 MHz. The multiplier and adder accumulator of the prototype moment generator ASIC should have no problem keeping up with the 7.68 MHz target clock rate necessary to process the digitized images at frame rates.

The worst case multiplication times for the final design can be calculated with the same method used to estimate the prototype. The final design will require a 26X8 multiplier array. Multiplying the carry propagation time of 2.5 nanoseconds by 26 allows 65 nanoseconds for the carry signal to propagate the maximum length of the first adder bank. Multiplying the worst case internal propagation time through the adder of 9.5 nanoseconds by 8 yields 76 nanoseconds. The sum of these two propagation times is 141 nanoseconds. The longest time allowed for any operation in the moment generator is 127 nanoseconds. To keep the final design within this limit, the multiplier must be broken into two 26X5 multiplier array stages. This can be done by inserting another pipe in the middle of the array. The new worst case propagation time is 112 nanoseconds.

7.2 TESTING

The first test on the chip is performed at the fabrication facility. The fabricators of the chip include test circuits on the silicon wafer. These test circuits are distributed strategically throughout the wafer. Once the fabrication process is complete, the test circuits are hooked up to test equipment and tested. This indicates to the vendors the yield probability for chips on that wafer.

The designer wishing to optimize an ASIC design for reliability must be aware of current-density levels of the internal circuits, peak current levels of the chip, power mesh-

current distribution, and the nature of the integrated-circuit processing faults. The power supply must be able to drive the largest peak current demands of the ASIC. For CMOS, these peak loads occur during switching mainly during switching of buffers driving large capacitive loads such as the I/O pads. The initial set of tests on the ASIC is done to ensure that the design of the power distribution system in the ASIC can supply power adequately to all of its functions. The first test of the chip is to measure its resistance between power and ground to ensure that there are no shorts. The next test is to apply power and measure the quiescent current flow. High values of current flow indicate internal shorting of the activated devices. The quiescent current flow of the ASIC should be under a few milliamperes. The last initial test is measuring power supply voltage and current levels while operating the chip at its target speed, 7.68 MHz in the case of the moment generator ASIC.

7.2.1 USING SCAN PATHS TO TEST THE FUNCTIONS

WITHIN THE CHIP

If faults are found in the circuit, the two scan paths of the moment generator circuit can be used to trace the errors to a particular function. The scan path that separates the multiplier function from the summation function can be used to examine the result of the multiplication without the adder or to input special test patterns into the adder to isolate problems within it. The accumulator register scan path allows special test vectors to be scanned into the accumulator register for input in the adder to test the second set of inputs into the adder. A system reset function makes starting such tests from a known state very easy. Otherwise, an initialization vector has to be scanned into the chip and clocked through the data path to set the internal nodes to a known state. Final testing of the entire chip should occur at or above the intended speeds of chip operation. There are many anomalies that can occur within CMOS ASIC designs that allow the chip to work properly at low speeds but fail at high speeds.

CHAPTER 8.

CONCLUSIONS

By combining the performance of custom VLSI special purpose processors with pattern recognition algorithms, such as moment invariants used in image processing, high-performance computational engines that can process digitized images in real time become possible. This thesis describes the design of a prototype ASIC used to implement ordinary and centralized moment calculations used for moment invariant algorithms for pattern recognition applications. The examination of the moment invariant algorithms, the determination of a suitable architecture, and the design and simulation of the cells used to build the basic functions through the simulation of the final moment generator ASIC have been presented.

The moment invariant algorithms were examined and the statistics from the type of arithmetic operations used to implement the equations required to determine the moment invariant constants were tabulated. The statistics formed the basis for the selection of the moment as the most time consuming part of the algorithm and the candidate for implementation into a VLSI ASIC. Space limitations on the chip and the number of I/O pins held the design effort to a proof-of-principle effort, which was composed of a multiply function and a summation function necessary to do an ordinary moment calculation. A review of the mechanics, which defined the operations of the MOSFET devices that were used in the custom design of the cells on the ASIC, allowed the cells to be designed with low-propagation times while maintaining high transition rates of the signals at the outputs through the use of additional buffering. The knowledge of the MOSFET mechanics also aided in understanding the simulation tools used in the design and simulation of the cells. Extensive simulation of the cells as well as simulations of the complete design gave a high degree of confidence in the success of the fabricated ASIC.

At the chip level, a regular structured design allowed a hierarchy of moment generator chips to be used to compute all the moments necessary for the moment invariant algorithms.

This design allowed the same ASIC design, at the chip level, to be used multiple times and for different purposes. At the silicon level, a structured design approach divided the two main functions into two halves of a data path-based design that allowed the moment at the silicon level to be calculated without the use of a state machine. The data path design, without any state machines, meant no synchronous feedback paths which eliminated the need for a two phase non-overlapping clock scheme. The design was implemented with a single-phase complimentary clock scheme, which reduced the amount of clocking circuitry required in the synchronous register cells. The use of a parallel register as a pipe between the multiplication and summation function, made the implementation of a scan path for testing the output of the multiplier and feeding the input to the summation function inexpensive in design effort and in terms of required real estate on the chip. The accumulation register in the summation function allowed the incorporation of a scan path in this function also.

8.1 SUMMARY OF THE PROTOTYPE DESIGN

The use of VLSI allowed custom modeling of the architecture to the application. The simple orthogonal and hexagonal two-dimensional array chosen for the architecture allowed quick and easy implementation of the multiplier function and the possibility of adding intermediate scan paths into the multiplier array to speed up the design and allow a higher degree of testability and observability in the chip. The design can be easily expanded to multiply larger multiplicands without any new design work.

The functions layout in the moment generator on the chip called for the design of two types of adder cells and two types of register cells, both of which had the same functional requirements but with different orientations of control and data signals layouts through the cell. Because of the amount of time required to design and test the cells, integrate the cells into the moment generator, and test the multiplier function with the second set of cells, the second set of cells never were implemented. The final proof-of-principle implementation of the moment generator consisted of a 4X8 multiplier and an 11-bit adder accumulator with a pipe between the adder and accumulator that acted as a scan path for test purposes. The accumulator also acted as a scan path with no serial output port as the data was available at the I/O pads. This allowed a regular design of the moment generator function with the necessary control signals in the memory registers using "metal2" to flow orthogonally to the data which was implemented primarily with "metal 1."

The cells were designed with speed optimization as the highest priority, high priority was given also to designing a cell that would not be susceptible to the problems associated with excessive loading. This was done by the addition of buffer pairs to prevent internal nodes in the chips from feeling the effects of external loading. The desire to use minimum size transistors in a regular layout with additional buffers to prevent any of the signal transitions from taking excessive amounts of time due to loading had a significant impact on the area required to implement the adder circuit. Even though the priority of latchup prevention affected the size of the individual cells, the area of the individual cells could be reduced significantly allowing a larger multiplier to be designed on the supplied real estate. The larger multiplier would require more I/O than pins are available on the 40-pin package used in the design, but it would allow the possible use of more pipeline stages, more sophisticated designs incorporating look-ahead carry functions, or the implementation of the modified Booth's algorithm for multiplication. In addition, the smaller area required by a moment generator implemented with smaller cells would allow the use of serial I/O functions for getting the data in and out of the chip with the 40 pins supplied on the NSF platform.

8.2 INTEGRATION OF TOOLS

Magic's incremental, hierarchical DRC makes it a user friendly interface as well as an instructional tool for learning about the technology's design rules. This makes it easier for VLSI designers to go from one technology, such as nMOS to another technology, such as scalable CMOS. In addition, as the CMOS dimensions are scaled down to even smaller dimensions it is not necessary to retrain the designers. Magic can be improved further by adding tests performed by implementations requiring more elaborate circuit extractors. This function could be added to Magic's present extractor utility so that unconnected nodes (nodes which short power and ground together) and transistors that have an unacceptable threshold voltage and therefore low noise margins (caused by improper selection of length-to-width ratios) can be reported to the designers before going to analog timing simulators such as Pspice. Logic simulators often do not detect such errors. Unconnected nodes, in these cases, are deemed "uninteresting" by the simulator (for example COSMOS) and are dropped from the circuit description. Transistors with improper thresholds are replaced with switches and resistance and possibly capacitance with little to no consideration given to the effects of improper

threshold values. The inclusion of such a capability would further reduce debugging time and would allow more efficiently VLSI designs.

The integration of the tools used to do all aspects of the VLSI needs a great deal of work. Tools are needed to improve the interface between the circuits extracted from the graphical editors and the simulators. Simulators often know about node numbers only and even though the node names are available the simulators do not use them. This can result in hours or days of work for the designer to set up a single test run. In addition, if changes are made in the circuit design, all the node names stay the same and the node numbers change. All control files that use node numbers must be redone, before simulation can be performed, with simulators that accept only node numbers rather than node names. Pspice is the most reliable simulator that was used to simulate the circuit; however, it uses only node numbers, not node names. Pspice takes an excessive amount of time to set up for large circuits and this has to be redone after every circuit change.

8.3 FUTURE WORK

A moment generator that would allow the 256 grey-scale 512X512 pixel image to be computed at frame rates can be made with the cells designed in the prototype chip. The chip necessary to implement a 9X26 multiplier and a 48-bit adder accumulator would need to be much larger and have more I/O pins than provided on the 40-pin 1800X1800 NSF-supplied platform. The cells in the final design should have with the two orientations that use the remaining chip area to the maximum advantage while maintaining an orthogonal relationship between the data flow and the flow of the control signals. The cells also could be designed to give an equal priority to speed and area, thereby significantly reducing the size of each cell. In addition, the adder banks could be converted to subtractors by inverting the inputs and changing the least significant carry in signal to a one rather than a zero. This opens up the possibility for designing the multiplier to function as a divider also. This would enable most of the calculations for the moment invariant algorithms to be done in silicon.

The moment generator chip then can be added to other chips used for pattern recognition applications to produce a complete pattern recognition engine. With the moment invariants constants from the moment generator chip and a conventional computer the constants could be used to generate distances from a known reference point allowing the data to be used in minimum-distance classification schemes. The resulting distance could be

compared with other distances in a VLSI design implementation of a binary tree or quad tree architecture used to implement a searching mechanism for the pattern comparison and identification scheme.

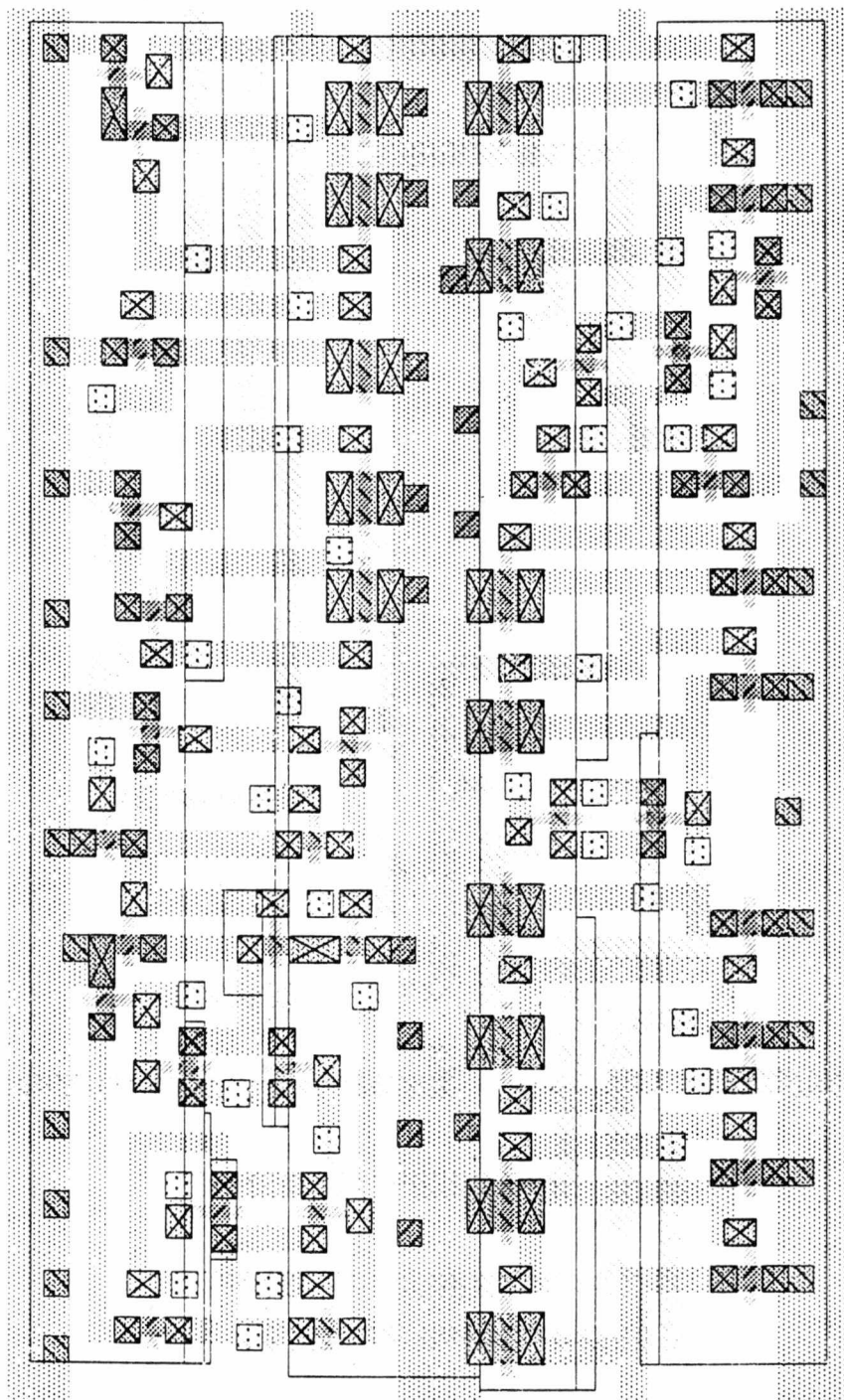
LIST OF REFERENCES

- [1] Banerjee, P., and Abraham, J.A., "Characterization and Testing of Physical Failures in MOS Logic Circuits," IEEE Design and Test of Computers, August 1984.
- [2] Beatty, D., Brace, K., Bryant, R. E., Cho, K., and Huang, L., "User's Guide to COSMOS: A Compiled Simulator for MOS Circuits", CMU-CS-86-01-068, Carnegie-Mellon University, Pittsburg, 1986.
- [3] Courtois, B., and Baschiera, D., "Testing CMOS: A Challenge," VLSI Systems Design, October 1984.
- [4] Fu, King-sun, "VLSI for Pattern Recognition and Image Processing," Reading, Spring Series in Information Sciences, New York, Tokyo, Springer-Verlag Berlin Heidelberg, 1984.
- [5] Glasser, L. A., and Dobberpuhl, D.W., "The Design and Analysis of VLSI Circuits," Reading, Mass.:Addison-Wesley, 1985.
- [6] Gonzales, R. C., and Wintz, P., "Digital Image Processing," Reading, Mass.:Addison-Wesley, 1987.
- [7] Hu, M. K., "Visual Pattern Recognition by Moment Invariants," IRE Transactions on Information Theory, pp. 179-187, February 1962.
- [8] Kuo, F. F., Editor, "An Introduction to Computer Logic," Reading, Prentice-Hall Computer Applications in Electrical Engineering Series.
- [9] MacSorley, O. L., "High-Speed Arithmetic in Binary Computers," Proceedings of the IRE, pp. 67-91, January 1961.
- [10] Maitra, S., "Moment Invariants," Proceedings of the IEEE, Vol. 67, No. 4, April 1979.
- [11] Mayo, R. N., Ousterhout, J. K., and Scott, W. S., "1983 VLSI Tools: Selected Works by the Original Artists," Rep. UCB/CSD 83/115, Computer Science Division, University of California at Berkeley, Calif., March 1983.
- [12] Mead, C., and Conway, L., "Introduction to VLSI Systems," Reading, Mass.:Addison-Wesley, 1980.
- [13] Mukherjee, A., "Introduction to nMOS and CMOS VLSI Systems Design," Reading, New Jersey: Prentice Hall, 1986.
- [14] Perry, R.E., and Tozun, O., "Design ASICs for High Reliability," VLSI Systems Design, May 1988.
- [15] Rubenfield, L. P., "A Proof of the Modified Booth's Algorithm for Multiplication," IEEE Trans. Computers C-24: pp. 1014-1015, 1975.
- [16] Simpson, M. L., "Moment Invariants for Automated Inspection of Printed Material," submitted for publication, Instrumentation and Controls Division, Oak Ridge National Laboratory, 1989.

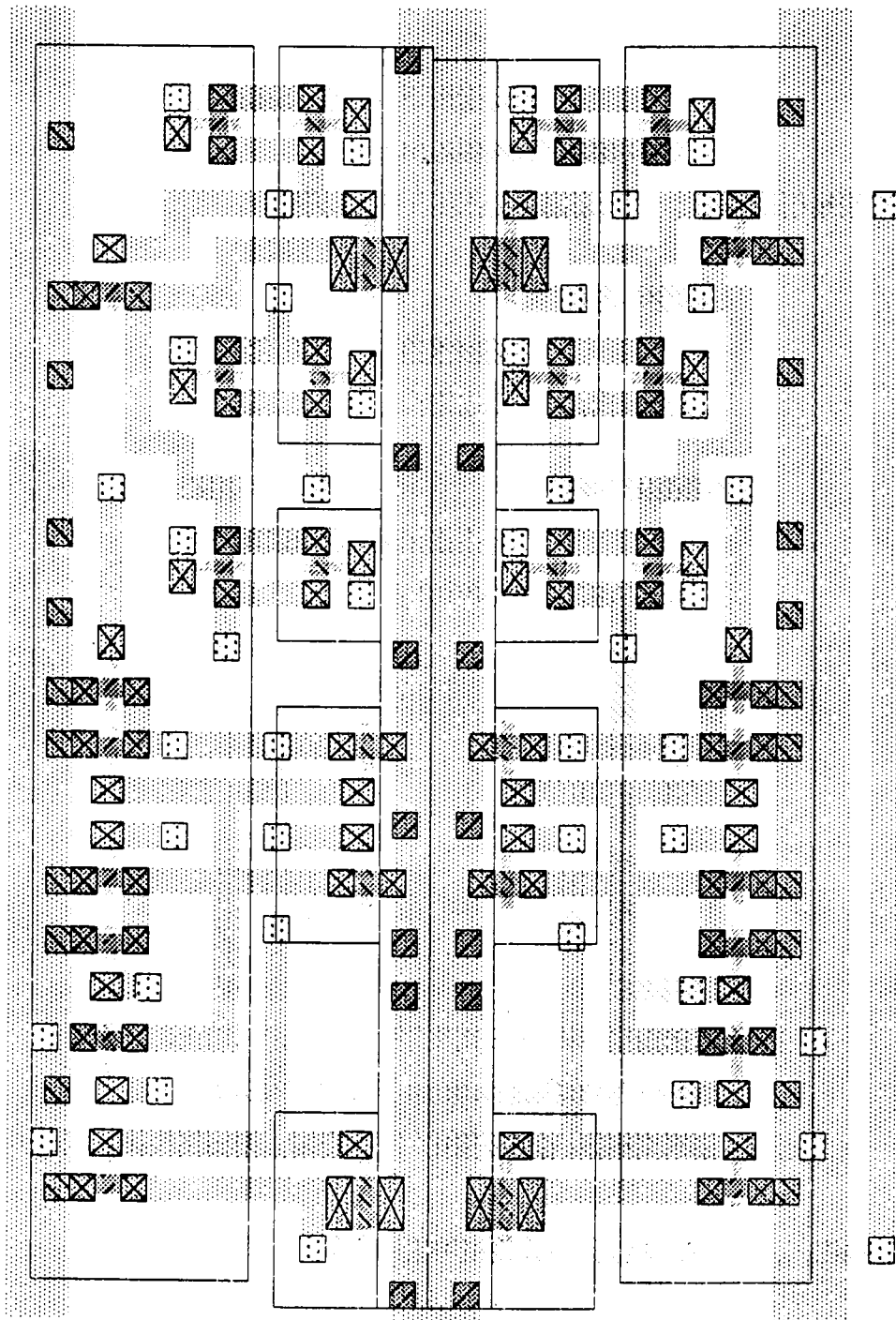
- [17] Tuinenga, P. W., "SPICE: A Guide to Circuit Simulation & Analysis Using PSpice," Reading, New Jersey: Prentice Hall, 1988.
- [18] Weste, N., and Eshraghian, K., "Principles of CMOS VLSI Design: A Systems Perspective," Reading, Mass.:Addison-Wesley, 1988.

APPENDIXES

APPENDIX A ADDER CELL LAYOUT



APPENDIX B MEMORY CELL LAYOUT



APPENDIX C COSMOS LOGIC SIMULATOR FILES

COSMOS Logic Simulation Control File for Manchester Carry Conditions Adder Cell

```
>comment Initialize the mcc cell for to simultaneous runs
>comment limit step:1000
>initialize 16
>clock
>comment watch the following nodes in the mcc cell
>watch %* SumIn
>watch %* Ai
>watch %* Bi
>watch %* CIn
>watch %* Ki
>watch %* Gi
>watch %* Pi
>watch %* COut
>watch %* SumOut
>watch %* SumOutBuf
>comment Set the following input nodes for the indicated logic values
>set SumIn:{0101010101010101}
>set Ai:{0011001100110011}
>set Bi:{0000111100001111}
>set CIn:{0000000011111111}
>set SumOut:0
>set SumOutBuf:0
>set COut:0
>set Pi:0
>set Ki:0
>set Gi:0
>comment Cycle the circuit one time - this cycles once through both sets
>comment of test conditions
>cycle
SumIn:{ 0101 0101 0101 0101 }
Ai:{ 0011 0011 0011 0011 }
Bi:{ 0000 1111 0000 1111 }
CIn:{ 0000 0000 1111 1111 }
Ki:{ 1010 1000 1010 1000 }
Gi:{ 1111 1110 1111 1110 }
Pi:{ 0101 0110 0101 0110 }
COut:{ 0000 0001 0101 0111 }
SumOut:{ 0101 0110 1010 1001 }
SumOutBuf:{ 0101 0110 1010 1001 }
CIn:{ 0000 0000 1111 1111 }
Pi:{ 0101 0110 0101 0110 }
SumOut:{ 0101 0110 1010 1001 }
SumOutBuf:{ 0101 0110 1010 1001 }
```

COSMOS Logic Simulator Control File for Static Register Cell

```
>limit step:1000
>comment /initialize the pmcc cell for to simultaneous runs
>initialize 8
>comment clock the Ph1 and Ph1bar clock lines
>clock Ph1:10 Ph1bar:01
>comment watch the following nodes in the pmcc cell
>watch %* ParIn
>watch %* ShftIn
>watch %* ParOut
>watch %* FF1Q
>watch %* FF1Qbar
>watch %* FF2Q
>watch %* FF2Qbar
>watch %* Reset
>watch %* Shft
>watch %* Shftbar
>watch %* Ph1
>watch %* Ph1bar
>comment Set the following input nodes for the indicated logic values
>set ParIn:{00100000}
>set ShftIn:0
>set Reset:{10000000}
>set Shft:0
>set Shftbar:1
>set FF1Q:{0xxxxxxx}
>set FF1Qbar:{1xxxxxxx}
>set FF2Q:{0xxxxxxx}
>set FF2Qbar:{1xxxxxxx}
>comment Cycle the circuit one time - this cycles once through both sets
>comment of test conditions
>cycle
1.1| ParIn:{00100000} ShftIn:0 ParOut:X FF1Q:{00100000} FF1Qbar:{11011111}
FF2Q:X FF2Qbar:X Reset:{10000000} Shft:0 Shftbar:1 Ph1:1 Ph1bar:0
1.2| ParIn:{00100000} ShftIn:0 ParOut:{00100000} FF1Q:{00100000} FF1Qbar:{1101
1111} FF2Q:{00100000} FF2Qbar:{11011111} Reset:{10000000} Shft:0 Shftbar:1
Ph1:0 Ph1bar:1
>
>quit
```

APPENDIX D COSMOS FAULT SIMULATOR FILES

COSMOS Fault Simulator Control File for Manchester Carry Conditions Adder Cell

```
copy mcc.cpy.fault
limit step:1000
clock
observe SumOutBuf COut
fault *@0 *@1
fault *\@0 *\@1
status fault *
set SumIn:0 Ai:0 Bi:0 CIn:0
cycle
set SumIn:0 Ai:0 Bi:0 CIn:1
cycle
set SumIn:0 Ai:0 Bi:1 CIn:0
cycle
set SumIn:0 Ai:0 Bi:1 CIn:1
cycle
set SumIn:0 Ai:1 Bi:0 CIn:0
cycle
set SumIn:0 Ai:1 Bi:0 CIn:1
cycle
set SumIn:0 Ai:1 Bi:1 CIn:0
cycle
set SumIn:0 Ai:1 Bi:1 CIn:1
cycle
set SumIn:1 Ai:0 Bi:0 CIn:0
cycle
set SumIn:1 Ai:0 Bi:0 CIn:1
cycle
set SumIn:1 Ai:0 Bi:1 CIn:0
cycle
set SumIn:1 Ai:0 Bi:1 CIn:1
cycle
set SumIn:1 Ai:1 Bi:0 CIn:0
cycle
set SumIn:1 Ai:1 Bi:0 CIn:1
cycle
set SumIn:1 Ai:1 Bi:1 CIn:0
cycle
set SumIn:1 Ai:1 Bi:1 CIn:1
cycle
status fault *
```

COSMOS Fault Simulator source file for Static Register Cell

```

copy pmccf.cpy
clock Ph1:10 Ph1bar:01
observe %2 ParOut
fault *@0 *@1
fault *\*@0 *\*@1
status fault *
comment Test ParIn
set ParIn:1
set Shft:0
set Shftbar:1
set Reset:0
cycle
set ParIn:0
cycle
observe %1 ParOut
set ParIn:1
cycle
clock Ph1:0 Ph1bar:1
observe %1 ParOut
set ParIn:0
cycle
limit soft:2
set ParIn:1
cycle
comment Test ShftIn
clock Ph1:10 Ph1bar:01
set Shft:1
set Shftbar:0
set ShftIn:1
cycle
set ShftIn:0
cycle
observe %1 ParOut
set ShftIn:1
cycle
clock Ph1:0 Ph1bar:1
observe %1 ParOut
set ShftIn:0
cycle
limit soft:2
set ShftIn:1
cycle

```

APPENDIX E PSPICE SIMULATOR CONTROL FILES

Pspice Control File for the Static Manchester Carry Conditions Adder

```
*
*
Vdd 1 0 DC 5.0 ;Vdd Supply = 5.0 Volt
*
* Transient Input Conditions - Piecewise Linear Models (PWL)
*
VIN1 22 0 PWL(0NS 5.0 10NS 5.0 13NS 5.0 20NS 5.0 23NS 5.0 40NS 5.0) ;Ai
VIN2 21 0 PWL(0NS 5.0 10NS 5.0 13NS 5.0 20NS 5.0 23NS 0.0 40NS 0.0) ;Bi
VIN3 7 0 PWL(0NS 0.0 10NS 0.0 13NS 0.0 20NS 0.0 23NS 0.0 40NS 0.0) ;SumIn
VIN4 15 0 PWL(0NS 5.0 10NS 5.0 13NS 5.0 20NS 5.0 23NS 5.0 40NS 5.0) ;CIn
*
* Inputs Initial Conditions (.IC used for transient analysis only)
*
.IC V(22) = 5.0 ;Ai
.IC V(21) = 5.0 ;Bi
.IC V(7) = 0.0 ;SumIn
.IC V(15) = 5.0 ;CIn
*
* Outputs Initial Conditions
*
.IC V(24) = 5 ;Gi
.IC V(27) = 0 ;Ki
.IC V(9) = 5 ;Pi
.IC V(10) = 0 ;Pibar
.IC V(12) = 5 ;PiBuf
.IC V(19) = 5 ;COut
.IC V(16) = 0 ;SumOut
.IC V(18) = 0 ;SumOutBuf
*
VSN 4 0 DC 0 ;mMOS ( CMOSN ) substrate to ground ( 0 Volts ) connection
VSP 20 0 DC 5.0 ;pMOS ( CMOSP ) substrate to Vdd ( 5.0 Volts ) connection
*
*
.INC mcc.spice ;Include spice deck from magic
*
*
.MODEL CMOSN NMOS(LEVEL=2 LD=0.200918U TOX=383.000026E-10
+ NSUB=1.671395E+16 VTO=.933381 KP=5.037E-05
+ GAMMA=0.8288 PHI=.6 UO=558.68 UEXP=0.189459
+ UCRIT=132832 DELTA=1.95323 VMAX=83157.9 XJ=0.25U
+ LAMBDA=6.465015E-3 NFS=2.332234E+12 NEFF=1.0
+ NSS=1.0E+12 TPG=1.0 RSH=21.200001 CGSO=1.81141E-10
+ CGDO=1.81141E-10 CJ=4.018E-4 MJ=0.4289 CJSW=4.275E-10
+ MJSW=0.3161 PB=0.73 CGBO=1.227032E-10)
```

```

.MODEL CMOSF PMOS(LEVEL=2 LD=0.165004U TOX=383.000026E-10
+   NSUB=8.038905E+15 VTO=-.76899 KP=1.703E-05 GAMMA=0.575
+   PHI=.6 UO=188.872 UEXP=0.266876 UCRIT=107223 DELTA=1U
+   VMAX=48228 XJ=0.25U LAMBDA=0.05415827 NFS=6.116363E+11
+   NEFF=1.001 NSS=1.0E12 TPG=-1.0 RSH=70.63
+   CGSO=1.487621E-10 CGDO=1.487621E-10 CJ=2.422E-4
+   MJ=0.3659 CJSW=2.05E-10 MJSW=0.1425 PB=0.45
+   CGBO=4.82879E-10)
*
.TRAN/OP 1NS 40NS 5NS      ; /OP = Save Initial Transient (bias) conditions
*
* Plot Input Waveforms
*
*.PLOT TRAN V(22)      ;Ai
*.PLOT TRAN V(21)      ;Bi
*.PLOT TRAN V(7)       ;SumIn
*
* Plot Intermediate Output Waveforms
*
*.PLOT TRAN V(24)      ;Gi
*.PLOT TRAN V(27)      ;Ki
*.PLOT TRAN V(9)       ;Pi
*.PLOT TRAN V(10)      ;Pibar
*.PLOT TRAN V(12)      ;PiBuf
*
*
* Plot Output Waveforms
*
*.PLOT TRAN V(16)      ;SumOut - Summation Out
*.PLOT TRAN V(19)      ;COut - Carry Out
*.PLOT TRAN V(18)      ;SumOutBuf - Buffered Summation Out
*
* Generate probe.dat file ( pspice -d0 a.cir )
*
.PROBE V(22) V(21) V(7)
.PROBE V(24) V(27) V(9) V(10) V(12)
.PROBE V(16) V(19) V(18)
*
.END

Pspice Simulation Control File for the static register cell
*
*
Vdd 1 0 DC 5.0      ;Vdd Supply = 5.0 Volt
*
*   DC Analysis Conditions for FFQ2
*
*.DC LIN VDCSWEEP 0.0 5.0 0.1      ; DC Sweep Analysis
*VDCSWEEP 22 0 DC      5.0      ; DC Sweep Independent Voltage source
*
* Transient Input Conditions - Piecewise Linear Models (PWL) and PULSE
*
VIN1 23 0 PWL( 0NS 0.0 30NS 0.0 33NS 5.0 50NS 5.0 53NS 5.0 ) ;ShftIn
VIN2 13 0 PWL( 0NS 0.0 10NS 0.0 13NS 0.0 20NS 0.0 23NS 5.0 40NS 5.0 ) ;ParIn
VIN3 11 0 PWL( 0NS 0.0 10NS 0.0 13NS 0.0 20NS 0.0 23NS 0.0 40NS 0.0 ) ;Reset

```



```

*
* Plot Transient Analysis Input Waveforms
*
*.PLOT TRAN V(13)      ;ParIn
*.PLOT TRAN V(23)      ;ShftIn
*.PLOT TRAN V(11)      ;Reset
*.PLOT TRAN V(18)      ;Shft
*.PLOT TRAN V(14)      ;Shftbar
*.PLOT TRAN V(19)      ;Ph1
*.PLOT TRAN V(16)      ;Ph1bar
*
* Plot Transient Output Waveforms
*
*.PLOT TRAN V(12)      ;ParOut / ShftOut
*
* Plot DC Analysis Output Waveforms
*
*.PLOT DC V(21)          ;FF2Q
* Generate probe.dat file ( pspice -d0 a.cir )
*
.PROBE V(13) V(23) V(11) V(18) V(14) V(19) V(16)
.PROBE V(21) V(12)
*
.END

```

Pspice Simulation Control File for the Moment Generator Version 3A ASIC

```

*
*.OPTIONS ITL5 = 0      ;Total Iterations limit set to infinity
*.OPTIONS VNTOL = 3u RELTOL = .005 ; Best Accuracy of volt ,Rel accuracy V and I
*
Vdd 1 0 DC 5.0          ;Vdd Supply = 5.0 Volt
*
* Transient Input Conditions - Piecewise Linear Models (PWL)
*
VINA0 63 0 PWL(0NS 0.0 10NS 0.0 13NS 5.0 70NS 5.0 73NS 0.0) ;AI0.I
VINA1 70 0 PWL(0NS 0.0 10NS 0.0 13NS 5.0 70NS 5.0 73NS 0.0) ;AI1.I
VINA2 77 0 PWL(0NS 0.0 10NS 0.0 13NS 5.0 70NS 5.0 73NS 0.0) ;AI2.I
VINA3 84 0 PWL(0NS 0.0 10NS 0.0 13NS 5.0 70NS 5.0 73NS 0.0) ;AI3.I
*
VINB0 56 0 PWL(0NS 0.0 10NS 0.0 13NS 5.0 70NS 5.0 73NS 5.0) ;BI0.I
VINB1 49 0 PWL(0NS 0.0 10NS 0.0 13NS 0.0 70NS 0.0 73NS 5.0) ;BI1.I
VINB2 42 0 PWL(0NS 0.0 10NS 0.0 13NS 0.0 70NS 0.0 73NS 5.0) ;BI2.I
VINB3 35 0 PWL(0NS 0.0 10NS 0.0 13NS 0.0 70NS 0.0 73NS 5.0) ;BI3.I
VINB4 28 0 PWL(0NS 0.0 10NS 0.0 13NS 5.0 70NS 5.0 73NS 5.0) ;BI4.I
VINB5 21 0 PWL(0NS 0.0 10NS 0.0 13NS 0.0 70NS 0.0 73NS 5.0) ;BI5.I
VINB6 14 0 PWL(0NS 0.0 10NS 0.0 13NS 0.0 70NS 0.0 73NS 5.0) ;BI6.I
VINB7 6 0 PWL(0NS 0.0 10NS 0.0 13NS 0.0 70NS 0.0 73NS 5.0) ;BI7.I
*
VCLOCK 139 0 PULSE ( 0.0V 5.0V 40NS 3NS 3NS 10NS 60NS ) ;Clock
*
VRST 153 0 DC 0.0      ; SysReset
VSHFT 146 0 DC 0.0     ; Shift
VPS 98 0 DC 5.0        ; PipeShftIn ***** Shift Test Only *****

```


* Watch Voltages of First Register outputs

*
.PROBE V(165) ; R0
.PROBE V(1390) ; R1
.PROBE V(1343) ; R2
.PROBE V(1212) ; R3
.PROBE V(1077) ; R4
.PROBE V(942) ; R5
.PROBE V(807) ; R6
.PROBE V(672) ; R7
.PROBE V(537) ; R8
.PROBE V(402) ; R9
.PROBE V(263) ; R10
*

* Watch Voltages of Accumulator outputs

*
.PROBE V(1428) ; A0
.PROBE V(1381) ; A1
.PROBE V(1334) ; A2
.PROBE V(1203) ; A3
.PROBE V(1068) ; A4
.PROBE V(933) ; A5
.PROBE V(798) ; A6
.PROBE V(663) ; A7
.PROBE V(528) ; A8
.PROBE V(393) ; A9
.PROBE V(250) ; A10
*

* Watch voltages of Moment outputs

*
.PROBE V(167) ; MO0.O
.PROBE V(174) ; MO1.O
.PROBE V(181) ; MO2.O
.PROBE V(237) ; MO3.O
.PROBE V(230) ; MO4.O
.PROBE V(223) ; MO5.O
.PROBE V(216) ; MO6.O
.PROBE V(209) ; MO7.O
.PROBE V(202) ; MO8.O
.PROBE V(195) ; MO9.O
.PROBE V(188) ; MO10.O
*

* Watch Control and Diagnostic Signals

*
.PROBE V(139) ; Clock.I
.PROBE V(146) ; Shift.I
.PROBE V(153) ; Reset.I
.PROBE V(98) ; PipeShftIn.I
.PROBE V(119) ; AccShftIn.I
.PROBE V(105) ; AddCntrl.I
.PROBE V(91) ; MultOverFlow.O
.PROBE V(112) ; AddOverFlow.O
.PROBE V(160) ; PipeShftOut.O
*

.END

APPENDIX F PSPICE SIMULATOR WAVEFORMS

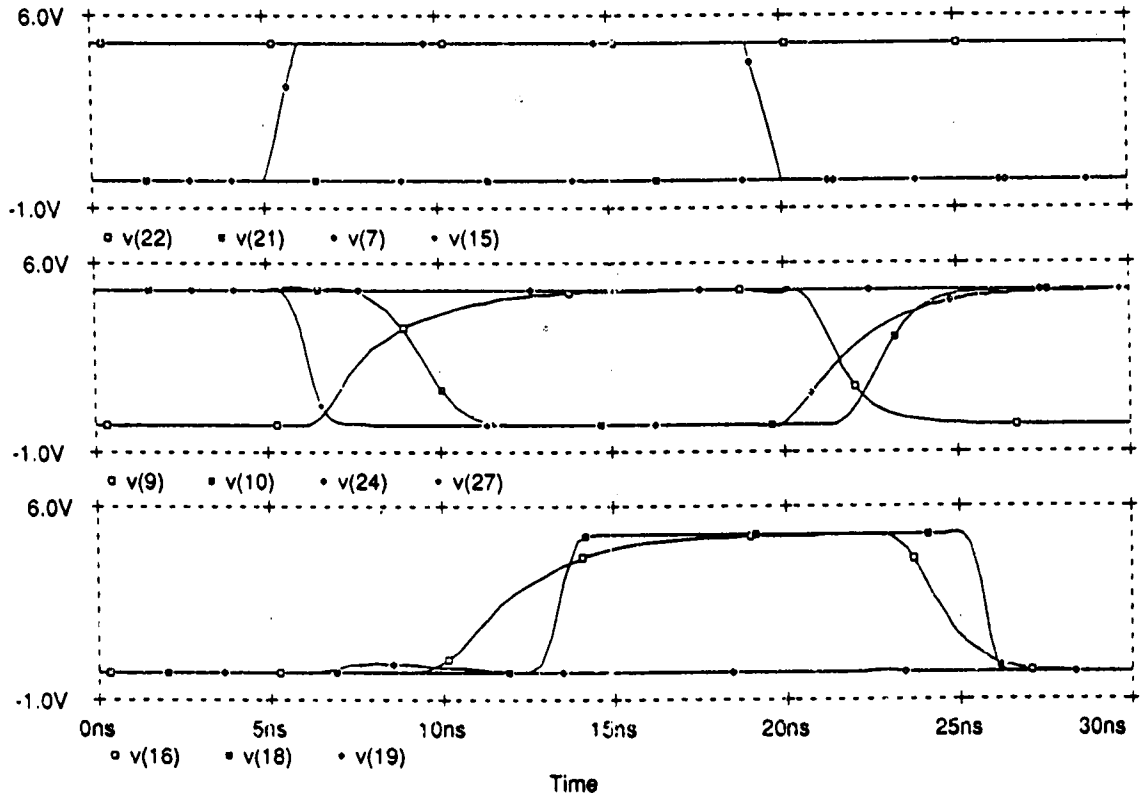
Manchester Carry Conditions Adder Cell Node Names

GND	0	
Vdd	1	
NMOS	2	
PMOS	3	
CMOSN	4	
3_187_350	5	
3_125_426	6	
SumIn	7	
3_155_412	8	
Pi	9	
Pibar	10	
3_187_292	11	
PiBuf	12	
3_54_346	13	
3_26_362	14	
CIn	15	
SumOut	16	
3_142_412	17	
SumOutBuf	18	
COut	19	
CMOSP	20	
Bi	21	
Ai	22	
3_109_236	23	
Gi	24	
3_18_296	25	
3_74_278	26	
Ki	27	
3_74_232	28	
3_187_224	29	
3_55_228	30	
3_96_220	31	

Manchester Carry Conditions Adder Cell SumIn to SumOut Test

Manchester Carry Conditions Generator SumIn to SumOut Test
Date/Time run: 10/29/89 15:11:06

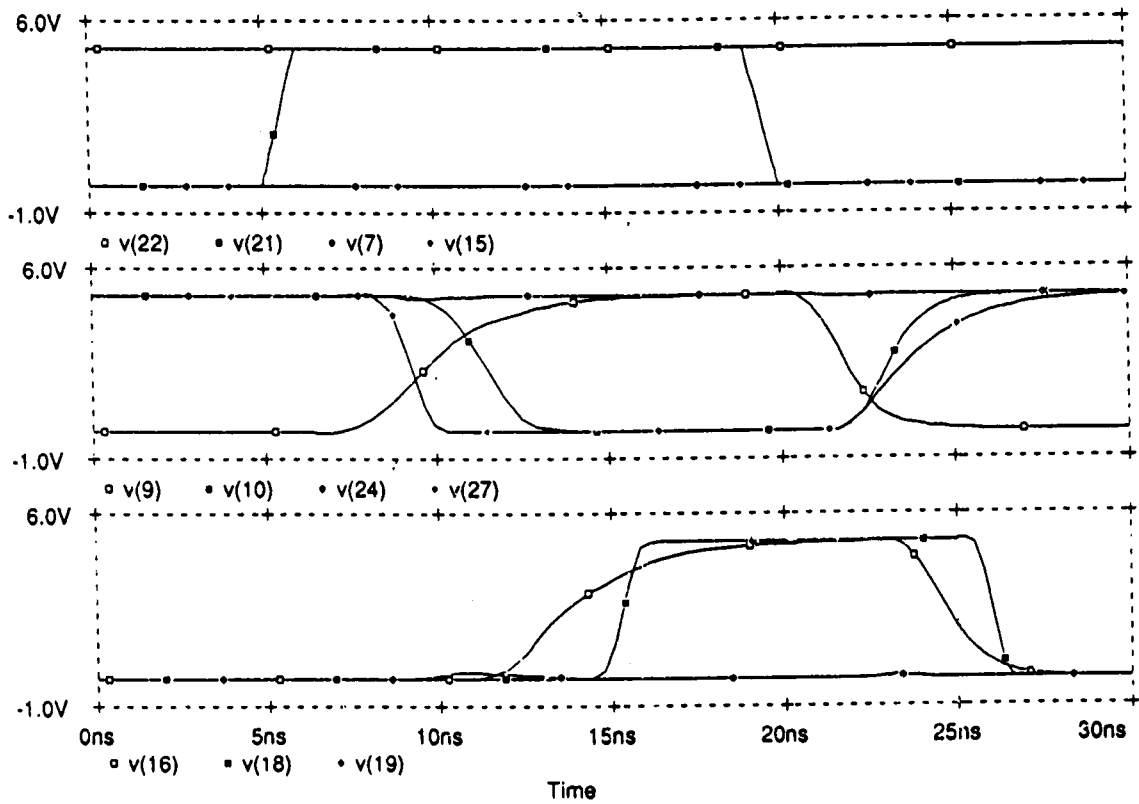
Temperature: 27.0



Manchester Carry Conditions Adder Cell Bi to SumOut Test

Manchester Carry Conditions Generator Bi to SumOut Test
Date/Time run: 10/29/89 14:47:25

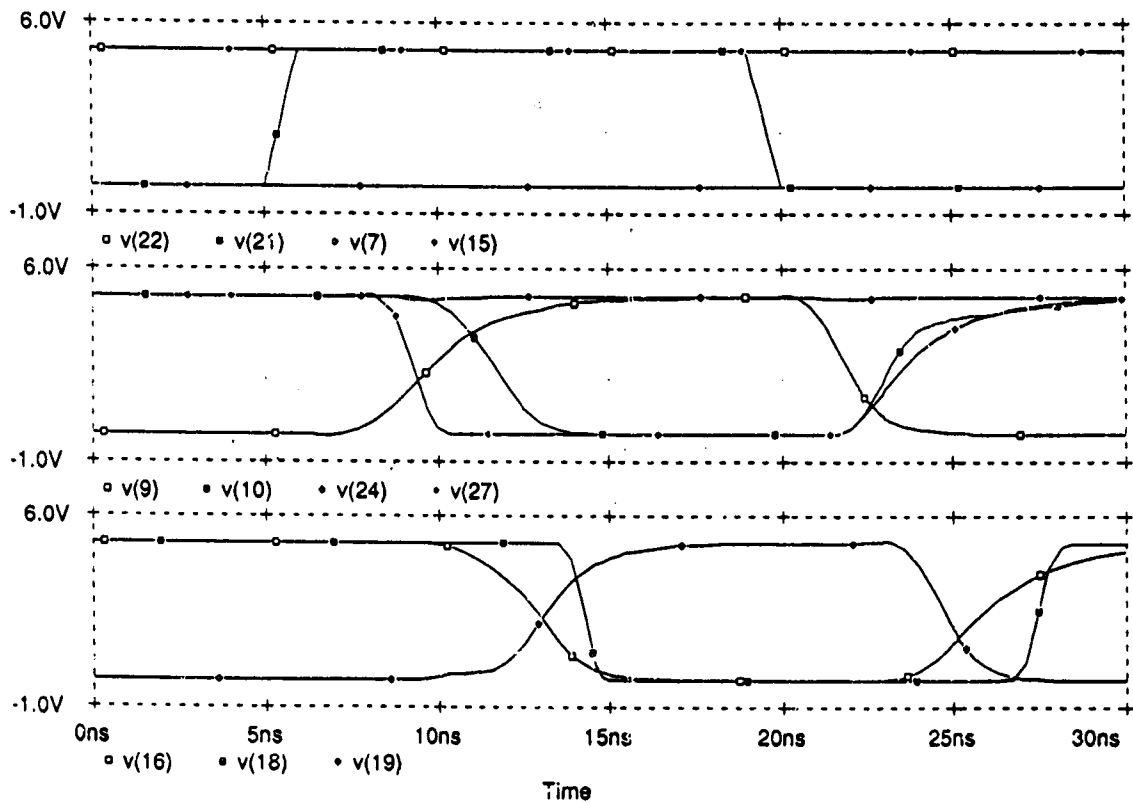
Temperature: 27.0



Manchester Carry Conditions Adder Cell Bi to COut Test

Manchester Carry Conditions Generator Bi to COut Test
Date/Time run: 10/29/89 14:21:54

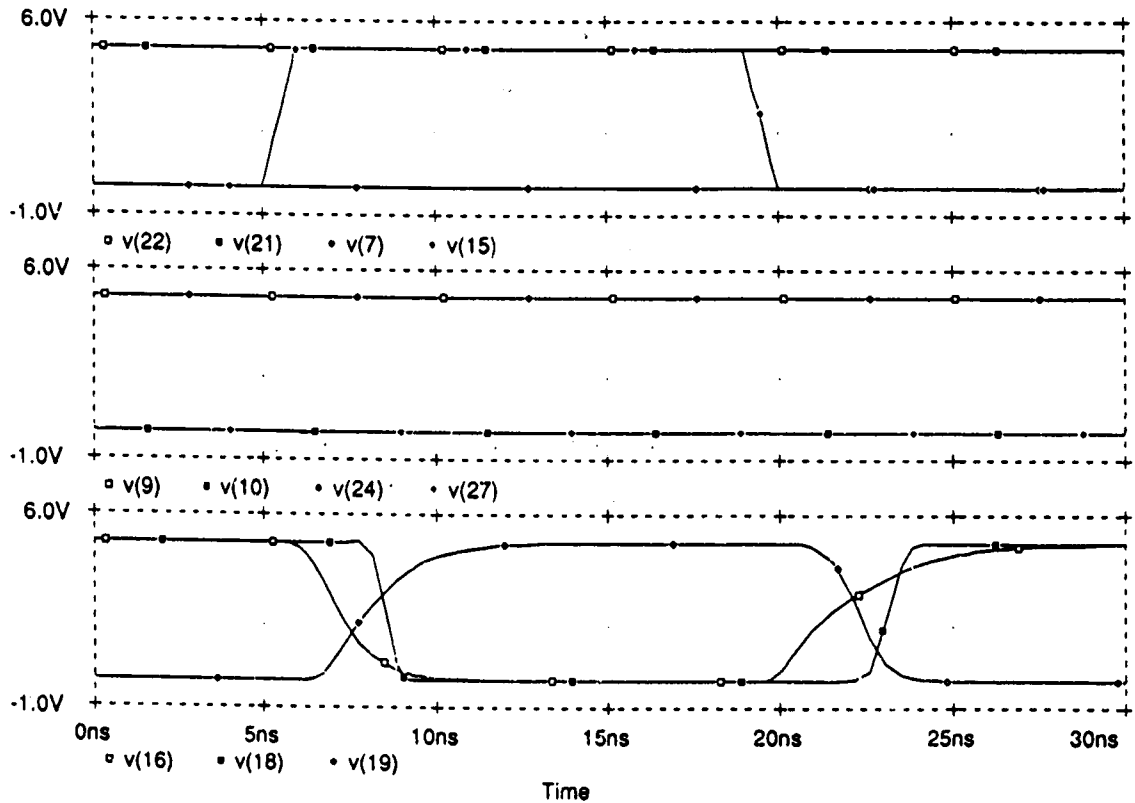
Temperature: 27.0



Manchester Carry Conditions Adder Cell CIn to COut Test

Manchester Carry Conditions Generator CIn to COut Test
 Date/Time run: 10/29/89 15:25:08

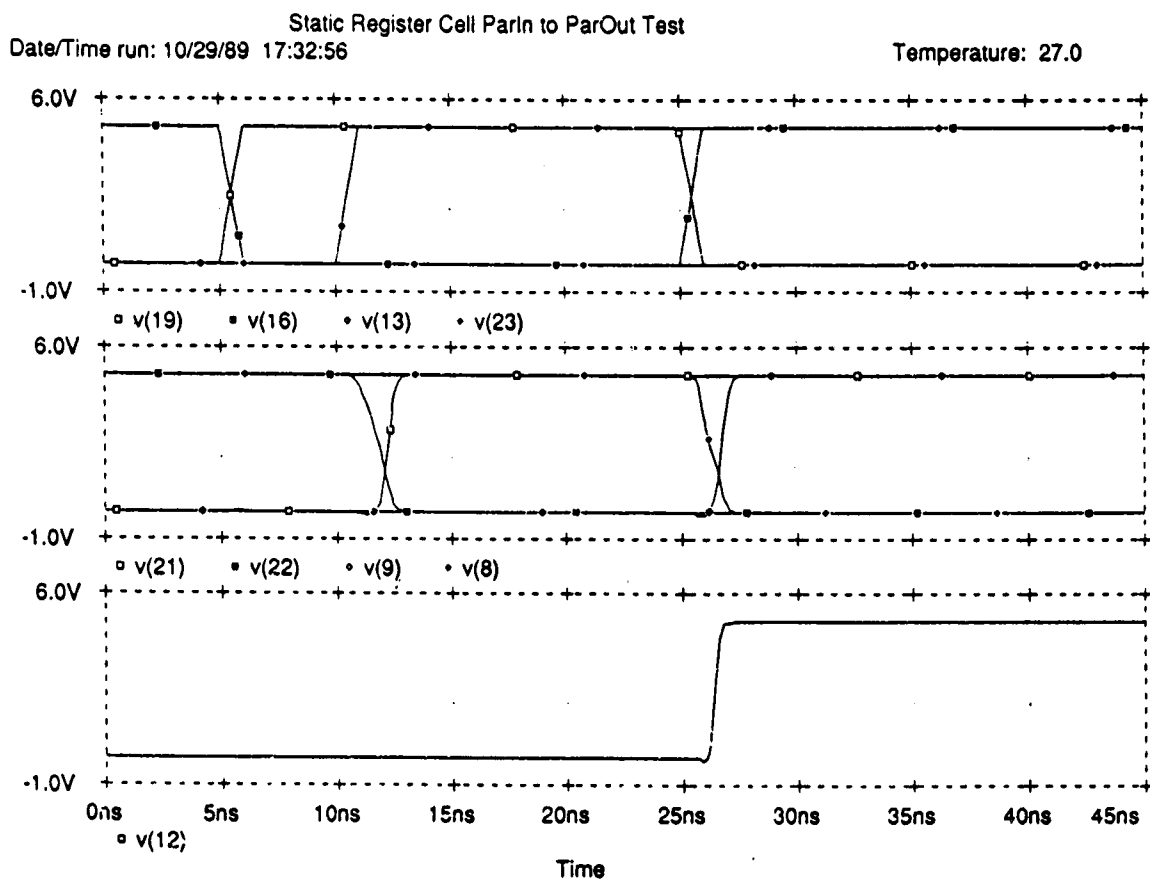
Temperature: 27.0



Static Register Cell Node Names

GND	0
Vdd	1
NMOS	2
PMOS	3
CMOSN	4
3_476_154	5
3_490_158	6
FFQ2barIn	7
FF2Qbar	8
FF2Q	9
FFQ2In	10
Reset	11
ParOut	12
ParIn	13
Shftbar	14
3_458_62	15
Ph1bar	16
CMOSP	17
Shft	18
Ph1	19
3_490_100	20
FF1Q	21
FF1Qbar	22
ShftIn	23
FF1QIn	24
FF1QbarIn	25

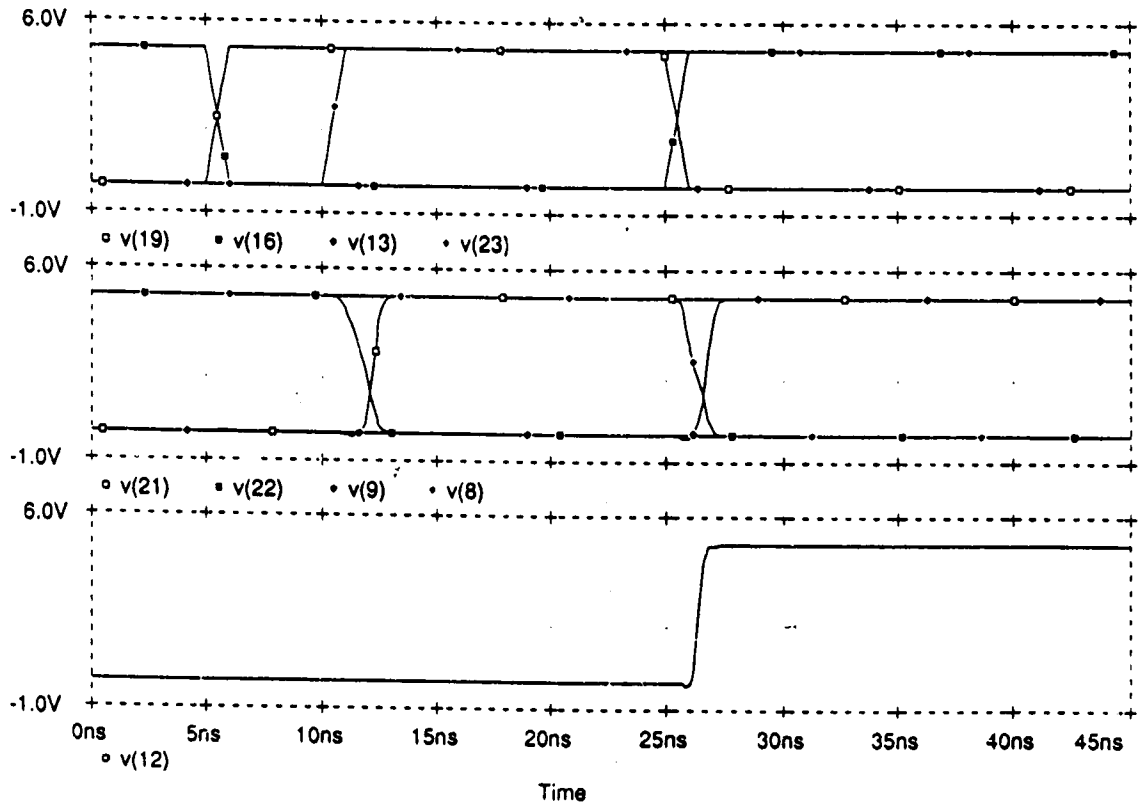
Static Register Cell ParIn to ParOut Low to High Test



Static Register Cell ShftIn to ParOut Low to High Test

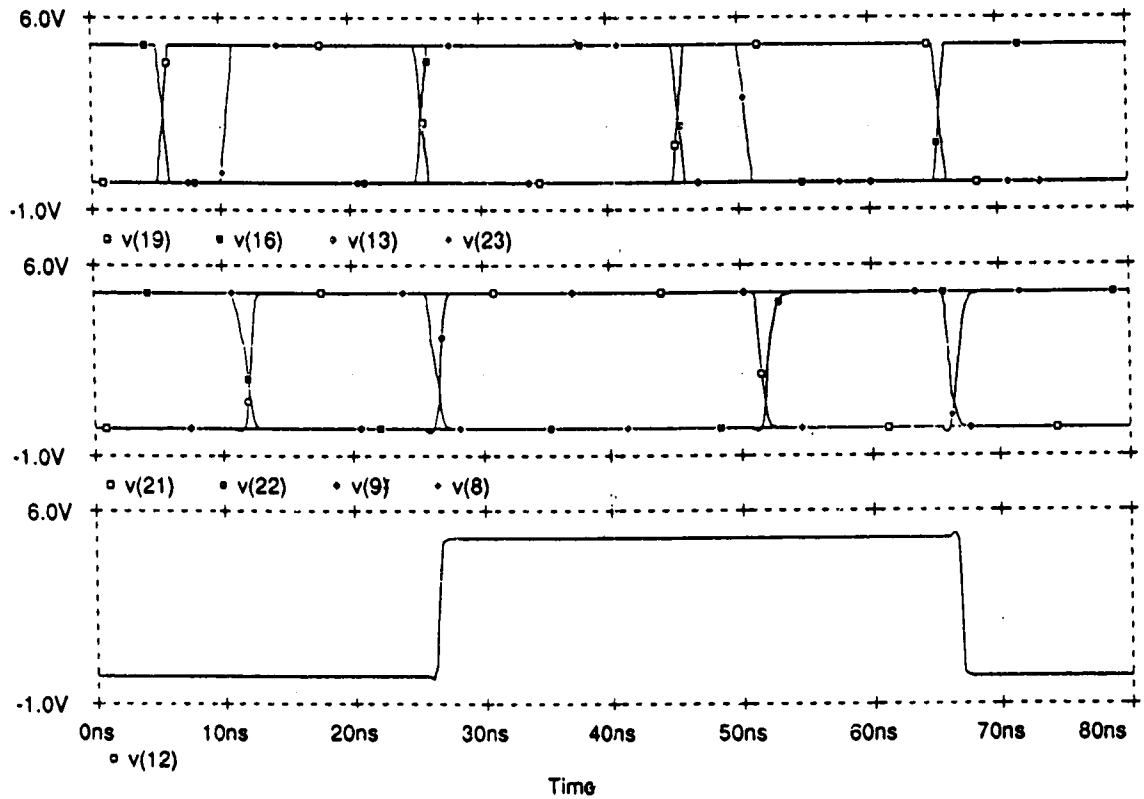
Static Register Cell ShftIn to ParOut Test tpHL
Date/Time run: 10/29/89 18:03:34

Temperature: 27.0



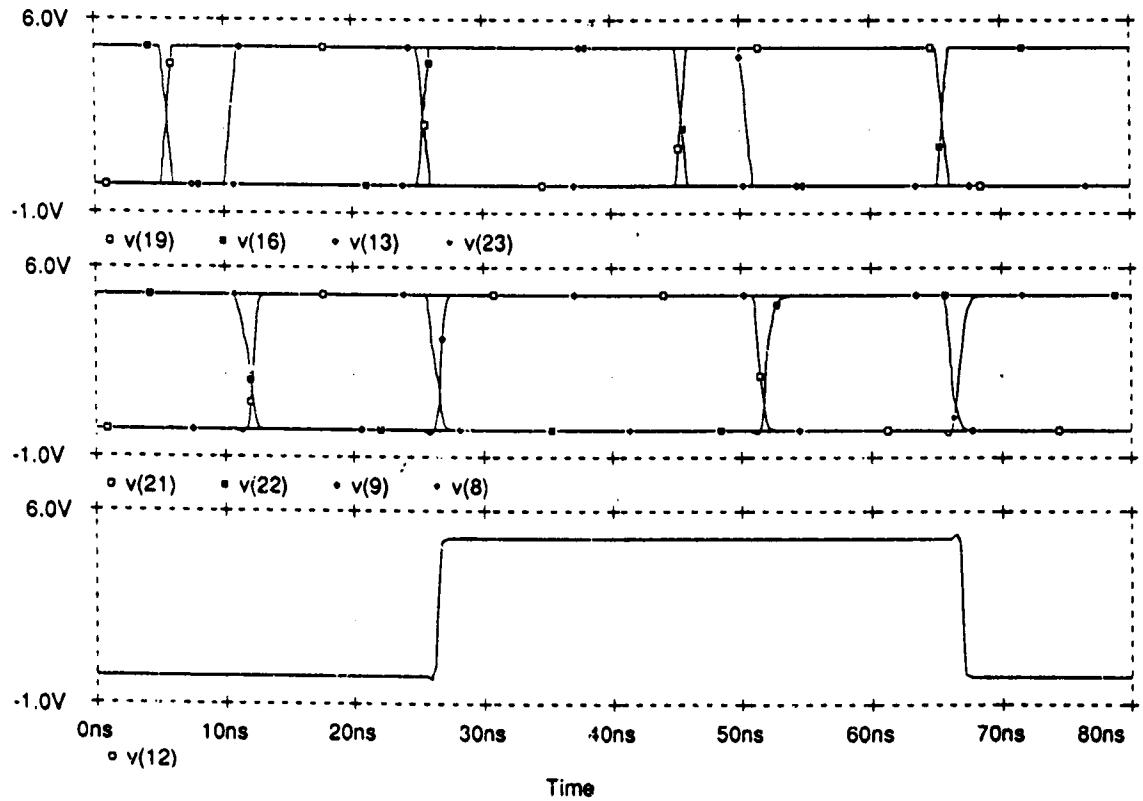
Static Register Cell ParIn to ParOut High to Low Test

Static Register Cell ShiftIn to ParOut Test tpHL
Date/Time run: 10/29/89 18:49:01 Temperature: 27.0



Static Register Cell ShftIn to ParOut High to Low Test

Static Register Cell ParIn to ParOut Test tpHL
Date/Time run: 10/29/89 19:06:32
Temperature: 27.0



VITA

Gary Armstrong received his B.S. in Electrical Engineering from the Tennessee Technological University, Cookeville, Tennessee, in 1981 and is currently a M.S. candidate in Electrical Engineering at the University of Tennessee, Knoxville.

Mr. Armstrong has been a development engineer in the Telerobotic Systems Section of the Instrumentation and Controls Division at the Oak Ridge National Laboratory (ORNL) since 1981. As a member of project design teams, Mr. Armstrong has participated in the design and implementation of a variety of computer-based acquisition and control systems. His contributions to these projects include software analysis, design and implementation, systems integration of high-performance computer systems, and design of high-performance discrete digital machines.

Mr. Armstrong is a member of the Institute of Electrical and Electronics Engineers, Inc., Instrument Society of America, Tau Beta Pi, Eta Kappa Nu, Omicron Kappa Delta, and Phi Kappa Phi.