Investigation of a programmable systolic cell using the RNS.

Bruce. Erickson

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
Erickson, Bruce., "Investigation of a programmable systolic cell using the RNS." (1989). Electronic Theses and Dissertations. 2500.
https://scholar.uwindsor.ca/etd/2500
NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.
INVESTIGATION OF A PROGRAMMABLE SYSTOLIC CELL
USING THE RNS

by

Bruce Erickson

A Thesis
Submitted to the Faculty of Graduate Studies through the
Department of Electrical Engineering in Partial Fulfillment
of the Requirements for the Degree of
Master of Applied Science at the
University of Windsor

Windsor, Ontario
April, 1989
The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

ABSTRACT

This thesis explores a programmable bit-level cellular structure that has been investigated for implementing digital signal processing (DSP) algorithms. The motivation for this work originated from recent favourable results obtained from a fabricated implementation of a fixed coefficient bit-level inner product step processor, operating over a finite ring (BIPSP_m). The new cell, presented in this thesis, reduces the limitation of fixed coefficient multiplication by replacing the computational array of the prior design with a programmable structure. The unique implementation results in minimum programming overhead, per cell, yet retains the features of the fixed coefficient BIPSP_m. Resulting in a much higher utility for the presented design. Also included in this thesis is a study of several currently available analog circuit simulators. Results from comparisons between simulators is presented, with recommendations and techniques to allow for their efficient use in the design process.
ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to Dr. G. A. Jullien and Dr. W. C. Miller for their tremendous support and guidance throughout the progress of this thesis. In addition, I would also like to thank Mr. J. Novosad and Mr. A. Johns for many favours. Thanks must also go to Melinda Burgess for her efforts and patience during my stay in Windsor.
TABLE OF CONTENTS

ABSTRACT iv
ACKNOWLEDGEMENTS v
TABLE OF CONTENTS vi
LIST OF FIGURES ix
LIST OF TABLES xii

CHAPTER 1

Introduction 1
1.1 Introduction 1
1.2 Thesis Organization 2

CHAPTER 2

Simulation and The MOS Transistor 3
2.1 Introduction 3
2.2 Circuit Simulators 4
2.2.1 SPICE 6
2.2.2 SPLICE 7
2.2.3 RELAX 8
2.3 Simulator Comparisons 8
2.4 MOS Transistor Operation 14
2.5 MOS Transistor Modelling 18
2.5.1 SPICE Level One Model 18
2.5.2 SPICE Level Two Model 23
2.5.3 SPICE Level Three Model 27
2.6 Research Results 29
2.6.1 D.C. Non-convergence 30
2.6.2 Transient Non-convergence 31
2.6.3 Power Simulations 34
2.7 Conclusions 36
CHAPTER 3

Bit Level Systolic Cells

3.1 Introduction 38
3.2 A Finite Ring Bit Level Inner Product Step Processor \((BIPSP_m)\) 39
3.3 A VLSI Implementation of \(BIPSP_m\) 45
3.4 Discussion of the Problem 48
3.5 ROM Based Designs 49
  3.5.1 Design I 50
  3.5.2 Design II 53
  3.5.3 Design III 56
  3.5.4 Design IV 61
3.6 RAM Based Designs 62
  3.6.1 Design V 62
  3.6.2 Design VI 65
3.7 Conclusions 75

CHAPTER 4

VLSI Implementation of a Programmable Systolic Cell 76

4.1 Introduction 76
4.2 Read/Write Memory Structures 76
  4.2.1 Static Read/Write Memory 77
  4.2.2 Dynamic Read/Write Memory 81
    4.2.2.1 Three Transistor Dynamic Memory 82
    4.2.2.2 Single Transistor Dynamic Memory 84
4.3 Summary of Memory Designs 86
4.4 Cell Design 86
4.5 Conclusions 87

CHAPTER 5

Conclusions and Summary 89

5.1 Conclusions 89
5.2 Summary 92
REFERENCES

APPENDIX A   Simulator Comparisons
APPENDIX B   VLSI Implementation of BIPSP<sub>m</sub>
APPENDIX C   Five-Bit Pipelined RNS Adder
APPENDIX D   Memory Test Structures
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Current research directions of circuit simulators</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Comparison of simulators using a boot-strapped inverter</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>MOS transistor cross section</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>$I_D$ vs $V_D$ Characteristics of an nmos device with physical dimension of $W=48$ and $L=24$ microns</td>
<td>16</td>
</tr>
<tr>
<td>2.5</td>
<td>$I_D$ vs $V_S$ of an nmos device with physical dimension of $W=48$ and $L=24$ microns</td>
<td>19</td>
</tr>
<tr>
<td>2.6</td>
<td>MOSFET device geometry</td>
<td>20</td>
</tr>
<tr>
<td>2.7</td>
<td>SPICE2G.6 level 1 model equations</td>
<td>22</td>
</tr>
<tr>
<td>2.8</td>
<td>Cross section showing small channel effects</td>
<td>24</td>
</tr>
<tr>
<td>2.9</td>
<td>Modified threshold to represent weak inversion</td>
<td>26</td>
</tr>
<tr>
<td>2.10</td>
<td>SPICE2G.6 level 2 model equations</td>
<td>27</td>
</tr>
<tr>
<td>2.11</td>
<td>SPICE2G.6 level 3 model equations</td>
<td>29</td>
</tr>
<tr>
<td>2.12</td>
<td>Power meter circuit</td>
<td>33</td>
</tr>
<tr>
<td>2.13</td>
<td>Power meter SPICE deck</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>The BIPSP cell</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>Finite ring BIPSP cell</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>Hardware implementation of a five bit BIPSP$_m$ system</td>
<td>44</td>
</tr>
<tr>
<td>3.4</td>
<td>Block diagram of the BIPSP$_m$ implementation</td>
<td>45</td>
</tr>
<tr>
<td>3.5</td>
<td>BIPSP$_m$ timing requirements</td>
<td>46</td>
</tr>
<tr>
<td>3.6</td>
<td>ROM array for a five bit system</td>
<td>48</td>
</tr>
<tr>
<td>3.7</td>
<td>Offset accommodated by using a modulus m adder</td>
<td>51</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.8</td>
<td>Five bit modulo adder</td>
<td>53</td>
</tr>
<tr>
<td>3.9</td>
<td>Design II block diagram</td>
<td>55</td>
</tr>
<tr>
<td>3.10</td>
<td>Shift register cell designs</td>
<td>56</td>
</tr>
<tr>
<td>3.11</td>
<td>8*8 barrel shifter schematic</td>
<td>57</td>
</tr>
<tr>
<td>3.12</td>
<td>3μ CMOS mask layout of an 8*8 barrel shifter</td>
<td>58</td>
</tr>
<tr>
<td>3.13</td>
<td>SPICE simulation of an nMOS barrel shifter</td>
<td>60</td>
</tr>
<tr>
<td>3.14</td>
<td>Design III block diagram</td>
<td>61</td>
</tr>
<tr>
<td>3.15</td>
<td>Method of implementing the offset</td>
<td>64</td>
</tr>
<tr>
<td>3.16</td>
<td>Design V block diagram</td>
<td>65</td>
</tr>
<tr>
<td>3.17</td>
<td>Required data path for initialization of a programmable structure</td>
<td>67</td>
</tr>
<tr>
<td>3.18</td>
<td>Required data path for initialization of a programmable array structure</td>
<td>68</td>
</tr>
<tr>
<td>3.19</td>
<td>Classic token ring structure</td>
<td>69</td>
</tr>
<tr>
<td>3.20</td>
<td>Block diagram of programmable BipSp_m and required system controller</td>
<td>69</td>
</tr>
<tr>
<td>3.21</td>
<td>Preliminary cell architecture for a programmable BipSp_m</td>
<td>71</td>
</tr>
<tr>
<td>3.22</td>
<td>Architecture of a dynamic programmable BipSp_m cell</td>
<td>74</td>
</tr>
<tr>
<td>3.23</td>
<td>System timing for the programmable cell</td>
<td>75</td>
</tr>
<tr>
<td>4.1</td>
<td>Read/Write memory structures</td>
<td>78</td>
</tr>
<tr>
<td>4.2</td>
<td>Five transistor static memory structure</td>
<td>80</td>
</tr>
<tr>
<td>4.3</td>
<td>Simulated characteristics of 5-transistor cell</td>
<td>81</td>
</tr>
<tr>
<td>4.4</td>
<td>Three transistor dynamic R/W memory</td>
<td>82</td>
</tr>
<tr>
<td>4.5</td>
<td>Three transistor dynamic R/W memory cell mask</td>
<td>83</td>
</tr>
<tr>
<td>4.6</td>
<td>SPICE simulations of three transistor R/W memory cell</td>
<td>83</td>
</tr>
</tbody>
</table>
4.7 Comparison of programmable cell vs $\text{BIP}_{m}$
LIST OF TABLES

2.1 Comparison of a conventional circuit simulator with SPLICE
2.2 Comparison of a conventional circuit simulator with RELAX
2.3 SPLICE vs SPICE features
2.4 Level three new and modified parameters
2.5 Parameters that aid in d.c. convergence
2.6 Parameters that aid in transient convergence
3.1 Characteristics of the implemented cell
3.2 Offset index relationship for a moduli 5 system
3.3 Characteristics of design I
3.4 Characteristics of design II
3.5 Characteristics of design III
3.6 Modulo five addition table
3.7 Characteristics of design V
3.8 Characteristics of design VI
1.1 INTRODUCTION

In the preceding decade the growth of integrated circuit technology has moved from large-scale integrated (LSI) circuits to very large-scale integrated (VLSI) circuits. The implementation of VLSI circuit technology has resulted in an increase in circuit density leading to logic gate counts three to five times greater than previously attained. This advance in integrated circuit technology has demonstrated itself in ROM technology as larger and faster ROM's are made commercially available. With larger and faster ROM's available, the use of Residue Number Systems (RNS) implemented as look up table operations has been the focus of researchers in recent years.

However, a new structure proposed by Taheri [25] uses small distributed ROM's arranged into standard cells. When grouped into linear arrays, this cell is capable of performing general arithmetic operations at a very high data rate. Several other properties of this cell, namely regular structure and testability make this cell very powerful and quite interesting. The limitation of the proposed architecture is that its function must be programmed at fabrication. The focus of this work is to examine modification of the cell...
architecture in such a way that programmability may be incorporated into the cell.

1.2 THESIS ORGANIZATION

This thesis consists of five chapters. The first chapter serves to present the problem which initiated this work and to lay out the structure of the remaining chapters. Chapter two investigates one of the key design tools used in the VLSI arena; analog circuit simulation. This topic is first introduced by a survey of presently available packages. After an initial evaluation, the theory required to make efficient use of such a package is presented. The chapter ends with a summary of techniques which were formulated through the course of this work.

Chapter three returns to the design problem presented earlier. It first presents the existing cell proposed by Taheri, followed by several possible design architectures. Comparison of these preliminary designs is based on simulations, where physical circuit parameters have been extracted from mask layouts where possible.

From these preliminary designs, a best design was chosen using the Area.Period product as the design criteria. The implementation of the chosen design in the VLSI realm is the subject of chapter four.

The final chapter concludes the thesis by stating the results derived in the previous chapters. Suggestions are also made in this chapter to give direction to future research in this area. Having covered the thesis outline, the body of the work done may now begin.
2.1 INTRODUCTION

The purpose of this chapter is to introduce the topic of computer-aided circuit analysis, in context to the integrated circuit engineer, using a metal oxide process. Circuit analysis programs are a crucial aid for today's design engineers, they eliminate the expense and delay of fabricate and test design procedures. This is key when one considers the cost of implementing a design in silicon, only to find out it does not operate as expected. For this reason the designer must spend a great deal of time early in the design cycle to verify his/her design. One such verification step in the design process is a check of the circuit for functionality. Aside from a functional verification the designer may use simulations to aid in engineering the circuit; to improve performance parameters such as speed, power dissipation and critical path analysis. Since simulations play such an important role early in any design, it was the starting point for this research.
2.2 CIRCUIT SIMULATORS

Two circuit simulators that have proven their worth for nearly a decade are SPICE (Simulation Program with Integrated Circuit Emphasis) [1] developed at the University of Berkeley and ASTAP (Advanced Statistical Analysis Program) [2] developed at IBM. Both of these simulators use what are called conventional methods and are essentially based on three techniques [3].

(i) Stiffly stable implicit integration methods, such as Backward Euler formula, for obtaining a system of nonlinear algebraic equations from the original system of nonlinear algebraic-differential equations describing the behavior of the circuit.

(ii) Newton-Raphson iteration to linearize the system of nonlinear algebraic equations of (i).

(iii) Sparse Gaussian elimination to solve the system of linear algebraic equations of (ii).

The problem however has arisen with the advent of VLSI technology of verifying the performance of large circuits containing hundreds or even thousands of transistors. The conventional circuit simulators mentioned above were designed initially for cost-effective analysis of circuits containing hundreds of transistors or less. This need to verify the performance of large circuits has resulted in several researchers actively pursuing the problem. Basically the research has taken two different directions for solving this problem, as shown in figure 2.1 below.
CONVENTIONAL SIMULATOR

ASTAP \rightarrow SPICE

- Hardware Enhancements
- New Circuit-Simulation Algorithms

- multiprocessor based simulation schemes
- waveform relaxation methods
  * RELAX
  * SPLICE
  - look-up-table implementations

Figure 2.1 Current research directions of circuit simulators

The dedicated-hardware approach, such as multiprocessor based simulation schemes can drastically reduce the circuit simulation time. However two factors keep researchers actively pursuing the second path. These are the fact that the same algorithms are used, thus the problem of convergence that has been well documented for large MOS circuits is still present. Secondly this dedicated hardware is not accessible to the masses, as are the simulators developed at institutions such as Berkeley. For this reason the hardware approach only benefits a small portion of user's. The second path promises to offer more than an order of magnitude speed-up as compared with conventional circuit simulators. Second generation simulators such as MOTIS, SPLICE RELAX and DIANA have been implemented using these new algorithms with varied success.

In the course of this research two of these second generation simulators have been evaluated and compared to the standard simulation program SPICE. A brief description of the methods used in conventional simulators, in
particular the SPICE2G.6 program is given below. Then the results of this research for the simulators RELAX and SPLICE will be presented.

2.2.1 SPICE

The equations which describe the electrical topology of a circuit are a system of algebraic differential equations of the form:

$$F(x', x, t) = 0$$  \hspace{1cm} (2.1)

SPICE by default solves the d.c solution prior to transient analysis to determine the transient initial conditions. For the d.c solution all time derivatives are zero, resulting in (2.1) reducing to the form shown below, since the vector $x'$ is zero.

$$F(x, t) = 0$$ \hspace{1cm} (2.2)

If initial conditions are not specified then for d.c analysis all capacitors are treated as open circuits and inductors as short circuits. However if initial conditions are specified then capacitors are treated as current sources and inductors as voltage sources. In this research the .NODESET command has been used successfully in specifying initial conditions, aiding circuits that have d.c. convergence problems. SPICE then uses the Newton-Raphson algorithm to linearize all non-linear model equations. The remaining system of linear equations is then solved using sparse matrix techniques. This solution is then used as the next guess and the process is continued until convergence. The d.c operating point solved above is used as the starting point for the transient analysis. The transient solution is determined computationally by dividing the time interval (0, T) into discrete time points. At each time point a numerical integration technique is used to transform the
differential model equations for energy storage elements into equivalent algebraic equations. After this transformation the time point solution is determined iteratively in the same fashion as the d.c. operating point [1]. SPICE offers two numerical integration methods, the default method is trapezoidal however the user may specify Gear's method if so desired.

This discussion has been kept brief, it's purpose is to introduce the unfamiliar reader with the basics of circuit simulators. The interested reader should refer to reference [4] for a discussion of circuit simulators or references [1], [5] for an in depth discussion of the SPICE simulator.

2.2.2 SPLICE

SPLICE is a mixed-mode simulator for integrated circuits composed of MOS devices. In contrast to SPICE which may perform D.C., A.C. and transient analysis, SPLICE is capable of performing transient analysis, switch level analysis and mixed-mode simulations. The latter allows different parts of a given circuit or system to be simulated in either of the previously mentioned methods. Switch level simulation is accomplished using supplied gates and allows the designer to take advantage of the time and memory savings available from higher level descriptions of blocks of the circuit. However for the purpose of comparison with SPICE only the transient analysis will be discussed in this research.

As in the SPICE program the first step in solving the set of ordinary differential equations given by (2.1) is to transform them into a set of difference equations using a numerical integration technique. In contrast to SPICE the integration technique used by SPLICE is fixed and uses a Backward-Euler formulation. The first equation of this set is then linearized using the
Newton-Raphson method and is iterated to convergence using a modified relaxation technique. This is then carried out with all equations of the system. This is in contrast to the SPICE program which uses sparse matrix techniques to solve the system of equations. The modified relaxation technique used by SPLICE is known as Iterated Timing Analysis (ITA) and interested readers should refer to reference [6] for a survey of relaxation methods and reference [7] for a discussion of the SPLICE program.

2.2.3 RELAX

RELAX performs transient analysis only on MOS integrated circuits. The program uses a mixture of direct methods and a form of a "Waveform Relaxation Algorithm". This combination of methods can greatly improve the computational efficiency of simulations. However there are problems in the algorithm, and for this reason they will not be discussed in this material. The interested reader could refer to reference [8]. More will be said about these problems in the next section.

2.3 SIMULATOR COMPARISONS

The accuracy of all simulators and their models have been compared experimentally using a boot-strapped inverter circuit. SPICE and the SPLICE simulators produce output waveforms that are reasonably close. However it can be seen in Figure 2.2 that SPLICE produces results which differ greatly with SPICE at time points near zero. This is due to the different initial value assumptions used by each simulator. SPLICE uses initial conditions of zero as would occur upon power-up of the circuit, whereas SPICE performs a d.c. analysis and uses these values as initial conditions. The implication of the analysis used by SPLICE, is that the circuit should not be clocked until it has
settled after power-up. If the circuit is clocked too early the simulation results will still be correct, however the circuit may operate unexpectedly.

![Simulator Comparison](image)

**Figure 2.2** Comparison of simulators using a boot-strapped inverter

The RELAX simulator produces varying results near seventy-five nanoseconds, this may be associated to the number of model parameters the user has access to.

It is important when making comparisons, such as above, that models of the same accuracy are used. For the above comparisons the Schichmann-Hodges [9] model was used by all simulators. Several other circuits have been simulated and the results are presented in appendix A.

The main advantage of simulators using relaxation techniques is the saving in CPU time and memory requirements. Comparisons between both
SPLICE vs SPICE and RELAX vs SPICE are given in the literature [3] and are presented here in tables one and two. The results reported in table one were obtained using a VAX-11/780 running under the UNIX operating system. One important observation that is not evident from these tables is the fact that for small circuits the solution time is smaller using standard methods.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>CDE</th>
<th>Digital Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mosfets</td>
<td>1326</td>
<td>705</td>
</tr>
<tr>
<td>Nodes</td>
<td>553</td>
<td>393</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>Memory (Kbyte)</td>
<td>Time (sec)</td>
</tr>
<tr>
<td>SPICE2G</td>
<td>115,840</td>
<td>2,420</td>
</tr>
<tr>
<td>SPLICE1.7</td>
<td>1,740</td>
<td>68.9</td>
</tr>
</tbody>
</table>

Table 2.1 Comparison of a conventional circuit simulator with Splice

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Shift Register</th>
<th>Two-bit Full Adder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mosfets</td>
<td>6</td>
<td>283</td>
</tr>
<tr>
<td>Nodes</td>
<td>4</td>
<td>45</td>
</tr>
<tr>
<td>SPICE2G (sec)</td>
<td>21.30</td>
<td>1,334.80</td>
</tr>
<tr>
<td>RELAX2.2 (sec)</td>
<td>1.08</td>
<td>22.30</td>
</tr>
</tbody>
</table>

Table 2.2 Comparison of a conventional circuit simulator with Relax

One might wonder, if relaxation techniques offer such a large saving in memory and speed requirements why is SPICE still so popular? The answer to this question is different for each simulator and is discussed below.

Problems with the algorithms used in RELAX have limited its success. The problem arises because on occasion the techniques used will cause the simulator to converge to an incorrect solution. For the circuit designer this is
not satisfactory. Designers are willing to use a program which gives them a correct answer or no answer, but are unable to find use with a simulator that occasionally produces the wrong answer. This problem has occurred in this research with simple circuits consisting of between ten to twenty transistors. For this reason the use of RELAX is not recommended.

SPLICE however uses a modified relaxation technique which by nature will converge to the correct solution if the initial guesses are reasonably close, eliminating the convergence problem mentioned above. In fact for large digital circuits SPLICE will produce results of the same accuracy as SPICE level one with a saving in CPU time. However the problem with SPLICE is that it cannot compete with the large amount of documentation and improvements made to the SPICE program over the last decade. A comparison of the features of both programs is shown in table 2.3 below. From this table it is evident that SPICE has many features beyond that of SPLICE. Namely, it allows the user to form d.c. characteristic curves, and also the number or parameters and choice of MOSFET models makes the program very flexible. For this reason it will be discussed in the next section when the MOSFET equations are developed and will be used as the primary simulator in the remainder of this work.
<table>
<thead>
<tr>
<th>Simulator</th>
<th>Models Available</th>
<th>Model Parameters</th>
<th>Device Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPICE</td>
<td>1. Schichman-Hodges</td>
<td>37 model parameters as seen in the users guide</td>
<td>- Width &amp; Length of channel</td>
</tr>
<tr>
<td></td>
<td>2. Analytical One-Dimensional</td>
<td></td>
<td>- Area of drain &amp; source used to determine drain capacitance</td>
</tr>
<tr>
<td></td>
<td>3. Semi-Empirical</td>
<td></td>
<td>- Perimeter of drain &amp; source used to determine sidewall junction capacitance</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Number of squares of the parasitic series resistance of drain &amp; source.</td>
</tr>
<tr>
<td>SPLICE</td>
<td>1. Schichman-Hodges</td>
<td>Five model parameters</td>
<td>Width &amp; length of channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- vto</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- kp</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- gamma</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- phi</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- lambda</td>
<td></td>
</tr>
<tr>
<td>Source Code</td>
<td>FORTRAN</td>
<td>ASSEMBLER</td>
<td>R-A-FOR</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Input Format</td>
<td>free format</td>
<td>names must be a character string less than eight integers</td>
<td>free format</td>
</tr>
<tr>
<td>Types of Analysis</td>
<td>D.C., A.C., Transient and analysis at different temps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Simulator</td>
<td>SPICE</td>
<td>S-P-LICE</td>
<td>S-P-LICE</td>
</tr>
</tbody>
</table>
2.4 MOS TRANSISTOR OPERATION

This section on MOS device characteristics is intended to be qualitative, rather than mathematically rigorous. The purpose is to introduce the unfamiliar reader with an understanding of MOS device operation. For those readers familiar with semiconductor physics the MOS transistor is essentially nothing more than a MOS-Capacitor with two pn junctions placed immediately adjacent to the region of the semiconductor controlled by the gate [20]. There are two possibilities for the silicon (Si) substrate, it can be either p-type or n-type material. The latter forms an pmos (or p-channel) transistor and the former is referred to as a nmos (or n-channel) device. The following discussion will apply for an nmos transistor (p-type substrate), however pmos operation is similar but will not be covered in an effort to keep this discussion brief.

A cross section of the above mentioned structure is shown in Figure 2.3 below.

![Figure 2.3 Mos Transistor Cross Section](image)
In simplest terms, a voltage applied to the gate electrode is used to set up an electric field that controls conduction between the heavily doped n-type (n⁺) source and drain regions.

This use of an electric field, gives rise to the devices suffix, FET (field-effect transistor). The device prefix comes from the fact that the gate is completely insulated from active region by a layer of thermally formed oxide (SiO₂). Resulting in the term IGFET (insulated-gate field-effect transistor). Early versions of the device used metal gates resulting in the more recent nomenclature Metal Oxide Semiconductor (MOS). The process supported by the Canadian Microelectronics Corporation uses a gate of polysilicon, but will still be referred to as a MOS device.

With the gate, source and drain all tied to ground the path between the source and drain consists of two pn junction diodes in series, back-to-back, with the p-type body common to both. Thus looking along the surface between the n⁺ islands one sees a virtual open circuit. As the gate voltage is increased electrons are drawn into the region under the gate from the n⁺ source and drain to form what is known as the channel. At some point an inversion layer adjacent to Si surface is formed containing mobile electrons. Now looking along the surface between the n⁺ islands one sees an induced n-type region (the inversion layer) or conducting channel connecting the source and drain islands. Naturally the greater the gate to substrate voltage the greater the accumulation of electrons near the Si surface and the greater the conductance of the channel, since more mobile electrons are available. If a voltage is now applied between the source and drain the mobile electrons
in the channel will result in a current between the drain and source electrodes.

If this drain to source voltage is small the $I_{DS} - V_{DS}$ relationship can be modelled by a simple linear resistor. This portion of the $I_{DS} - V_{DS}$ relationship is shown in figure 2.4 between the points O and A. Where the transistor curves have been obtained from devices fabricated by the Canadian Microelectronics Corporation (CMC) using the Northern Telecom 3µ CMOS process.

![I-V Characteristics](image)

Figure 2.4 $I_{DS}$ vs $V_{DS}$ Characteristics of an nmos device with physical dimension of $W=48$ and $L=24$ microns.

However once $V_{DS}$ is increased above a few tenths of a volt the device enters a new phase of operation. This new region is known as the non-saturation
region and results from the fact that as $V_{DS}$ is increased the depletion region increases resulting in a decrease of induced mobile conduction charge available in the channel. This reduction in the number of mobile carriers results in a decrease in the channel conduction, which in turn is reflected as a decrease in the slope of the $V_{DS}$-$I_{DS}$ characteristics. Eventually as $V_{DS}$ is increased the inversion layer completely vanishes in the vicinity of the drain (assuming a positive voltage is applied between the drain and source). At this point the channel is said to be pinched-off, this point is shown in Figure 2.4 as the point B and also the middle cross section of Figure 2.6. The voltage at which pinch-off occurs is referred to as the saturation voltage ($V_{D_{sat}}$) of the device. Increases in $V_{DS}$ above $V_{D_{sat}}$ produces little change in the drain to source current. In other words $I_{DS}$ has saturated, or reached a limit at which the voltage $V_{DS}$ has little effect on the drain to source current.

From the above discussion and the current vs voltage characteristics depicted in Figure 2.4 it is evident that the MOS transistor has at least three regions of operation which need to be accounted for in any simulation:

- Linear Region
- Non-Saturation Region
- Saturation Region

The above discussion has been intended to familiarize the reader qualitatively with the operation of the MOS device. The overall characteristics of the device has been illustrated. Basically for digital design applications the device is used as a switch whose state (open/closed) depends on the voltage applied to the gate.
2.5 MOS TRANSISTOR MODELLING

The previous discussion qualitatively described the operation of the MOS transistor when used as a switch, by examining the d.c. characteristic curves. However the integrated circuit designer requires a mathematical model which describes the device accurately throughout all regions of operation. It would also be advantageous if such a model lent itself to computer analysis techniques.

One such model has been proposed by Schichmann and Hodges [9] and has been implemented in several computer packages for integrated circuit simulation (SPICE, SPLICE, RELAX, WATAND). The theory outlined in the next section is based on this model, as implemented in the SPICE2G.6 simulator developed at The University of Berkely. The theory is a simplified approximation which will be built upon in section 2.5.2 when the improvements added for level two of SPICE2G.6 are discussed. The reader interested in a detailed discussion of the mathematical models describing MOS devices should refer to the book by Tsvidis [20].

2.5.1 SPICE LEVEL ONE MODEL

An important characteristic of the MOS transistor that has not yet been addressed is the fact that for small gate voltages the channel does not form and conduction between the source and drain is not appreciable. This phenomena can be seen in Figure 2.5 below. The minimum gate voltage required to initiate the formulation of the conducting channel is referred to as the threshold voltage ($V_{TH}$). It arises from the fact that there is a contact potential between the gate and the substrate material and a fixed charge related to the oxide beneath the gate. This requires an opposing voltage
applied to the gate to overcome these two factors. This voltage has been proposed by Schichmann and Hodges [9] as:

\[
V_{TH} = V_{FB} + 2\Phi_p + \gamma \sqrt{2\Phi_p + V_{SB}} \tag{23}
\]

Figure 2.5 \(I_{DS}\) vs \(V_{GS}\) of an nmos device with physical dimension of W=48 and L=24 microns.

where \(\gamma\) is given by:

\[
\gamma = \frac{\sqrt{2\varepsilon_s q N_A}}{C_{ox}} \tag{24}
\]

and \(V_{FB}\) is the flat-band voltage which is defined as the gate voltage required for equal carrier concentrations throughout the substrate. The second term \((2\Phi_p)\) is the potential required for inversion to occur in the channel.
Figure 2.6 MOSFET device geometry
\[ V_{FB} = \Phi_{MS} - q \frac{N_{SS}}{C_{ox}} \]

= contact potential - interface charge

The last term \( \gamma \sqrt{2 \Phi_p + V_{SB}} \) takes into account what is known as the back bias or body effect, which causes the threshold voltage to increase with increasing source-to-substrate voltage.

The Schichmann and Hodges model also breaks the drain current expression into two different region as shown below:

**Non-Saturation Region**

where \( V_{GS} > V_{TH} \) and \( V_{DS} < V_{GS} - V_{TH} \)

\[ I_{DS} = \mu C_{ox} \left( \frac{W}{L_{eff}} \right) \left[ (V_{GS} - V_{TH}) V_{DS} - \frac{V_{DS}^2}{2} \right] \]  

(2.6)

**Saturation Region**

where \( V_{GS} > V_{TH} \) and \( V_{DS} > V_{GS} - V_{TH} \)

\[ I_{DS} = \frac{\beta}{2} (V_{GS} - V_{TH})^2 \]  

(2.7)

and

\[ \beta = \mu C_{ox} \left( \frac{W}{L_{eff}} \right) \]  

(2.8)

From equation 2.7 it is seen that in saturation the drain current \( I_{DS} \) is completely independent of the drain to source voltage. However as device geometries get smaller this is not the case. As can be seen from Figure 2.6 as \( V_{DS} > V_{DSat} \) the pinch-offed section begins to extend a larger distance into the channel. If \( \Delta L \) becomes compatible in size to \( L \), the case in small geometry devices, the same voltage drop \( V_{DSat} \) will appear across a shorter channel (L-
ΔL) resulting in a slight increase in $I_{DS}$. The SPICE level one model takes this into account by introducing a term into the model equation as an empirical correction of the conduction in the saturation region. To avoid discontinuities in the $I_{DS}$-$V_{DS}$ characteristics the empirical term $(1 + \lambda V_{DS})$ is also included in the non-saturation region. The resulting level one SPICE2G.6 equations are given below.

\[
\text{Non-Saturation Region}
\]

\[
I_{DS} = K_P \left( \frac{W}{L - 2X_{ij}} \right) (V_{GS} - V_{TH} - \frac{V_{DS}}{2}) V_{DS} (1 + \lambda V_{DS})
\]

\[
\text{Saturation Region}
\]

\[
I_{DS} = \frac{K_P}{2} \left( \frac{W}{L - 2X_{ij}} \right) (V_{GS} - V_{TH})^2 (1 + \lambda V_{DS})
\]

where $X_{ij}$ is the lateral diffusion as shown in Figure 2.3 and the following variable are given by:

- $LD = X_{ij}$
- $K_P = \mu C_{ox} = UO C_{ox}$
- $\gamma = \sqrt{\frac{2 \varepsilon_s q N_A}{C_{ox}}} = \sqrt{\frac{2 \varepsilon_s q N_{SUB}}{C_{ox}}}$
- $\Phi = 2\phi_p = 2\frac{KT}{q} \ln \left( \frac{N_A}{n_i} \right) = 2\frac{KT}{q} \ln \left( \frac{N_{SUB}}{n_i} \right)$
- $C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}} = \frac{\varepsilon_{ox}}{TOX}$

Figure 2.7 SPICE2G.6 Level 1 Model Equations

The parameters required for the above equations can each be entered in several different methods. Either on the device card or the model card, however there is a close correlation between each location and the SPICE
documentation should be consulted for a comprehensive description of all parameters and their default values.

2.5.2 SPICE LEVEL TWO MODEL

This section presents the features that have been added to the level one model to incorporate second order effects. The following effects have been incorporated in the level two model:

- Small-geometry effect on the threshold voltage
- Saturation due to the limited drift velocity of carriers
- Finite voltage dependent output conductance in saturation

However the addition of the above effects does not come freely. Level two has the greatest complexity of all three models, resulting in the largest amount of CPU time required for calculations. Level two also has a discontinuity or 'kink' in it's $I_D$ vs $V_{DS}$ characteristics as reported by Doganis [13]. This discontinuity can become a problem during transient analysis, resulting in the 'INTERNAL TIMESTEP' failure. The remainder of this section will discuss the enhancements included in the level two model.

Threshold Voltage

The threshold voltage discussed in relation to the level one model applies for large devices (typically $W$ and $L > 20 \mu m$) [21]. It does not take into account the two dimensional effect of the channel dimensions. Physically the threshold voltage is dependent on the source/drain diffusion depths and the channel length, decreasing threshold voltage with decreasing channel dimensions. As explained in section 2.5.1 and equation 2.3 - 2.5 the threshold
voltage is a function of the charge stored beneath the gate region and its corresponding electric field. In the case of short channel MOS devices some of the field lines caused by this charge are terminated in the n+ islands rather than the channel [22]. This can be best visualized through a cross section, where it is seen that the area of charge may be represented by a trapezoid rather than the ideal rectangle.

![Cross Section Diagram](image)

Figure 2.8 Cross section showing small channel effects

An equation which sums up the above features can be expressed as:

\[
V_{TH} = V_{BIN} + \gamma_s \sqrt{\frac{2\phi_F - V_{BS}}{}}
\]

where

\[
V_{BIN} = V_{BI} + \text{DELTA}\left(\frac{\Pi E_{Si}}{4 C_{ox} W (2\phi_F - V_{BS})}\right)
\]

\[
V_{BI} = V_{FB} + 2\phi_F
\]

\[
V_{BIN} \text{ is the corrected built-in voltage for a narrow channel device and } \gamma_s \text{ is the corrected GAMMA for a short-channel device [21].}
\]

\[
\gamma_s = \text{GAMMA} (1 - \alpha_s - \alpha_d)
\]
where $\alpha_D$ and $\alpha_S$ are the correction factors for the depletion charge at the drain and source.

$$\alpha_S = \frac{X_I}{2L} \left( \sqrt{1 + 2 \frac{W_S}{W_I}} \right)$$  \hspace{1cm} (2.13)

$$\alpha_D = \frac{X_I}{2L} \left( \sqrt{1 + 2 \frac{W_D}{W_I}} \right)$$  \hspace{1cm} (2.14)

and

$$W_S = X_D \sqrt{2\phi_F V_{BS} + V_{DS}}$$  \hspace{1cm} (2.15)

$$X_D = \sqrt{\frac{2\varepsilon_S}{q N_A}}$$  \hspace{1cm} (2.16)

it should also be noted that in level two $X_I$ is also used as a flag to implement this correction for short channel effects, if not included the threshold voltage is evaluated as given for level one.

**Drain-Source Current**

In order to correct for small dimensions the drain source current equations is slightly modified to include the effect of bulk charge [21]. The current in it is given by [21]:

**Non-Saturation Region**

where $V_{GS} > V_{TH}$ and $V_{DS} < V_{GS} - V_{TH}$

$$I_{DS} = \beta \left( (V_{GS} - V_{BIN} - \frac{\eta V_{DS}}{2}) V_{DS} - \frac{2}{3} \gamma_s [(2\phi_F + V_{DS} - V_{BS})^{1.5} - (2\phi_F - V_{BS})^{1.5}] \right)$$  \hspace{1cm} (2.17)

where

$$\eta = 1 + \text{DELTA} \frac{\Pi \varepsilon_S}{4C_{ox}W}$$  \hspace{1cm} (2.18)
\[ \beta = \frac{W}{L} \mu_s C_{ox} \]  

(2.19)

Saturation Region

where \( V_{GS} > V_{TH} \) and \( V_{DS} > V_{GS} - V_{TH} \)

\[ I_{DS} = I_{D, sat} \left( \frac{1}{1 - \lambda V_{DS}} \right) \]  

(2.20)

and \( I_{D, sat} \) is calculated from 2.17 at \( V_{DS} = V_{D, sat} \). From equation 2.17 it can be seen that the level two equations are quite complicated and little insight can be obtained from them. For this reason they are presented here but not discussed.

Weak Inversion

In the previous section (level one) the MOSFET was modelled as an ideal switch which started conducting abruptly when the voltage on the gate was greater than the threshold voltage \( (V_{TH}) \). However this is not the case, there is a current flowing below the threshold voltage.

Level two handles this by including a new voltage \( V_{ON} \) as shown in Figure 2.9, which marks the transition from weak inversion to strong inversion. The value of \( V_{ON} \) is given by [5]:

![Figure 2.9 Modified threshold to represent weak inversion](image-url)
\[ V_{ON} = V_{TH} + \frac{n kT}{q} \]  
\[ n = 1 + \frac{qN_{FS}}{C_{ox}} + \frac{C_d}{C_{ox}} \]

where:
\[ C_d = \frac{\gamma}{2 \sqrt{2\phi_p - V_{BS}}} \]

The resulting level two equations are summarized in Figure 2.10 below.

**Non-Saturation Region**
\[ I_{DS} = \beta \left[ (V_{GS} - V_{BIN} - \frac{\eta V_{DS}}{2}) V_{DS} - \frac{2}{3} \gamma_s \left( (2\phi_p + V_{DS} - V_{BS})^{1.5} - (2\phi_p - V_{BS})^{1.5} \right) \right] \]

**Saturation Region**
\[ I_{DS} = I_{D_{sat}} \left( \frac{1}{1 - \lambda V_{DS}} \right) \]

where
\[ \eta = 1 + \text{DELT}A \frac{\Pi \varepsilon_{si}}{4C_{ox} W} \]
\[ \beta = \frac{W}{L} \mu_s C_{ox} \]

Figure 2.10 SPICE2G.6 Level 2 Model Equations

### 2.5.3 SPICE LEVEL THREE MODEL

Though level two and level three share most of the same model parameters, different values for the same parameters must be used to produce similar results [21]. The reason being that level three model was developed to
address the problem of computational efficiency. Resulting in many of the
equations used in the model being empirical. The purpose of this semi-
empirical model is to improve the precision and reduce the complexity of the
required calculations. Four parameters with new meaning in level three are
shown below in Table 2.4:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>SPICE* Keyword</th>
<th>Parameter Description</th>
<th>Default Value</th>
<th>Typical Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>η</td>
<td>ETA</td>
<td>Static feedback on threshold voltage</td>
<td>0.0</td>
<td>1.0</td>
<td>dimensionless</td>
</tr>
<tr>
<td>δ</td>
<td>DELTA</td>
<td>Width effect on threshold voltage</td>
<td>0.0</td>
<td>1.0</td>
<td>dimensionless</td>
</tr>
<tr>
<td>θ</td>
<td>THETA</td>
<td>Mobility modulation</td>
<td>0.0</td>
<td>0.05</td>
<td>V⁻¹</td>
</tr>
<tr>
<td>κ</td>
<td>KAPPA</td>
<td>Saturation field correlation factor</td>
<td>0.20</td>
<td>0.50</td>
<td>dimensionless</td>
</tr>
</tbody>
</table>

Table 2.4 Level three new and modified parameters

Again the equations are developed from the level two physical phenomena,
and little insight is derived from their formulation. For this reason they are
presented in Figure 2.11 and again the interested reader should refer to the
references [5], [21].

It has been reported that the level three model has an average quadratic error
less than that of level two [5], and runs 40% faster [21]. For this reason it is
suggested as the level to use for all simulation, however to date The VLSI
Research Group does not have a utility to extract the required parameters.
2.6 RESEARCH RESULTS

In this section techniques will be presented which enhance the use of the SPICE program. The material presented here is the result of the experience obtained during the course of this research and material presented by others in the literature. This section is divided into four areas. The first two discuss the most frequently encountered SPICE errors, d.c. non-convergence and transient non-convergence. The cause of these errors is explained and steps to circumvent them are given. The following segment gives a technique to enhance the power of the SPICE program. Finally, limitations with the simulator are discussed in the last section.
2.6.1 D.C. NON-CONVERGENCE

As explained in section 2.2.1 the d.c. operating point is obtained by an iterative process. Which is terminated when both of the following conditions are true [5].

1. The nonlinear branch currents converge to within a tolerance of 0.1 percent or $10^{-12}$ A, whichever is larger.

2. The node voltages converge to within a tolerance of 0.1 percent or $10^{-6}$ V, whichever is larger.

Failure to converge in d.c. analysis is catastrophic since it is required for transient analysis. For this reason the algorithm used for the d.c. operating point has as a first priority reliable convergence characteristics, and of secondary importance is the iterations required to converge. Thus convergence errors in d.c. analysis are usually due to an error in specifying the input circuit. However circuits with positive feedback may exhibit d.c. convergence problems.

To verify that a given input deck is an accurate representation of a circuit, use the LIST and NODE parameters on the .OPTIONS card. If it has been determined that the input deck is correct the .NODESET card can be used to help improve the initial guesses for each node. By default SPICE chooses node voltages of zero for all nodes. The initial node voltages may be roughly calculated by hand analysis for small circuits or for larger circuits a source-stepping method may be used[1]. In this method the d.c. voltage source $E$ is stepped from zero to its final d.c. voltage ($S$):

$$E = \alpha S$$

where $\alpha$ stepped from 0 to 1
and the intermediate solutions are used as initial conditions on the .NODESET card for the following solution. Re-labeling of the source and drain may also aid in convergence since the drain current is used in the check for convergence and the source is not [10]. If positive feedback is present the OFF option for devices in the feedback path may also be used [5].

If none of the above modifications have aided in d.c. convergence, the parameters that determine convergence may be modified. However the user must change these values cautiously in accordance with his/her need for accuracy. Relaxing these values may allow convergence but will introduce errors that could accumulate and may allow the circuit to enter an unreal state. These parameters are shown in table 2.5 below. When modifying these parameters the user should pay close attention to the simulated results.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELTOL</td>
<td>Resets the relative error tolerance of the program.</td>
<td>0.001</td>
</tr>
<tr>
<td>ABSTOL</td>
<td>Resets the absolute current error tolerance.</td>
<td>1 pA</td>
</tr>
<tr>
<td>VNTOL</td>
<td>Resets the absolute voltage error tolerance of the program.</td>
<td>1 μV</td>
</tr>
</tbody>
</table>

Table 2.5 Parameters that aid d.c. convergence

2.6.2 TRANSIENT NON-CONVERGENCE

Transient solutions are also obtained using an iterative process which terminates when both of the conditions previously mentioned for d.c.
analysis are true. Failure to converge in the transient analysis usually results in the error message.

**INTERNAL Timestep TOO SMALL IN TRANSIENT ANALYSIS**

The cause of this error is that during the transient analysis SPICE is constantly modifying the internal integration timestep. If convergence occurs before a preset number of iterations occur, the timestep is doubled to improve computational efficiency. However if non-convergence occurs the timestep is reduced by a factor of eight until convergence occurs, or a preset limit is reached. The latter causes the "internal timestep" error.

It has been reported in the literature[11] that usually these large transients occur at the first timestep. A solution for this is to use the UIC option on the .TRAN card without specifying any initial conditions. In this case the circuit will reach steady-state gradually as if the circuit had just been powered up. However the circuit must reach steady-state before the time dependent sources are applied. It is also important to specify all d.c. source voltages on the .IC card since d.c. bias is not computed before the transient analysis.

If this method does not work 10 GΩ shunt resistors and 1 Ω series resistors may be placed at each node in an attempt to slow them down [12].

If non-convergence errors occur while using MOS level two this may be due to discontinuities in the I-V characteristics as mentioned in the literature [13]. It seems there is a "kink" in the I-V characteristics between the non-saturation and saturation regions. A fix for this problem is to upgrade to MOS level three, if extracted parameters are available.
Changing the integration method from the default trapezoidal to gear by using the METHOD parameter on the .OPTION card may also help convergence problems, but the user will pay the price in increased CPU time. Relabeling of the source and drain may also aid in convergence since the drain current is used in the check for convergence and the source is not. The drain charge is also used in timestep control but not the source charge, therefore, a different number of points is used with different labeling of the source and drain [10].

As a last attempt to aid in convergence the user may change SPICE's error tolerances, however these must be changed carefully and the user must pay close attention to the simulation results. The parameters that influence transient analysis are given in the table below.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVLTIM=2(^1)</td>
<td>Causes truncation-error timestep to be used.</td>
<td>2</td>
</tr>
<tr>
<td>TRTOL</td>
<td>Transient error tolerance. If TRTOL is increased less importance is given to the truncation error and it is less likely that internal timestep will be divided by eight.</td>
<td>7</td>
</tr>
<tr>
<td>LVLTIM=1</td>
<td>Iteration timestep control is used.</td>
<td>2</td>
</tr>
<tr>
<td>ITL4</td>
<td>Transient analysis time point iteration limit. If LVLTIM=1 this is the parameter that effects the timestep. Increasing ITL4 will aid in convergence but increase CPU time.</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 2.6 Parameters that aid in transient convergence

\(^1\) If the GEAR method for integration is chosen and MAXORD \(> 2\) then LVLTIM is set to 2.
2.6.3 POWER SIMULATIONS

The scale of integration is often limited by the acceptable maximum power dissipation allowable for a given technology [14]. One of the major reasons for CMOS technology becoming a major factor in VLSI is its reduced power requirements, when compared to other technologies. However even with this reduced power need the designer must still minimize power dissipation, while meeting other design objectives such as speed, area and cost. Therefore it is quite useful to have power data available throughout simulations. SPICE however does not accommodate power information and to date either post processors were used to calculate power information, or methods of estimating power consumption based on formulas were used. One such formula [15] is:

\[ P = C_T V_{DD}^2 f \]  \hspace{1cm} (2.24)

where \( C_T \) is the total effective capacitance affected by the logic change in the cycle \( f \). Application of this formula to large circuits is impractical because of the difficulty in calculating \( C_T \).

However a technique for measuring average power dissipation during a SPICE simulation has recently been presented by S. M. Kang [14] and enhanced by G. J. Fisher [16], the later will be given here. The circuit used is shown in Figure 2.12 and can be include in a subcircuit that possess no loading on the circuit under simulation.
The spice deck for this circuit is given below in figure 2.13.

The power meter circuitry consists of a time dependent voltage source, a nonlinear second-order dependent current source and an RC network. With a proper choice of the current gain parameter $\beta$ the voltage ($V_c$) across the capacitor $C_1$ will register the average power consumption $P_{avg}$.

```
FC  0  1  POLY 2  VDD  VSW  0  0  0  0  10E6
R1  1  0  1
C1  1  0  1
VSW 2  0  1
RSW 2  0  1
```

Figure 2.13 Power meter SPICE deck

The proper value of $\beta$ can be derived as in [16]:

$$P_{avg} = \frac{1}{T} \int_0^T \frac{V_{DD}}{T} \int_0^T i_{dd}(t) \, dt$$

and from the circuit shown in Figure 2.12
\[ C_1 \frac{dV_c}{dt} = \beta i_{dd} i_{sw} \frac{V_c}{R_1} \]  
(2.26)

solving for \( V_c(t) \) under the constraint

\[ R_1 C_1 \gg T \]  
(2.27)

results in

\[ V_c(t) = \frac{\beta}{C_1} \int_0^t i_{dd}(\tau) i_{sw}(\tau) d\tau + V_c(0) \]  
(2.28)

If the voltage source \( V_{sw} \) is pulsed between values of zero and one volt for \( T \) seconds starting at \( t = t_0 \) with \( R_{sw} = 1.0 \Omega \) the current \( i_{sw}(t) \) will be given by:

\[
i_{sw}(t) = \begin{cases} 
1.0 & \text{for } t_0 \leq t \leq t_0 + T \\
0.0 & \text{otherwise}
\end{cases}
\]

Now if the value of \( \beta \) is such that:

\[ \beta = \frac{C_1 V_{DD}}{T} \]  
(2.29)
then the capacitor voltage \( V_c(t \geq t_0 + T) \) will be equal to the average power dissipated.

The virtue of this method is that it places no loading on the circuit to be measured, does not require any modification to the circuit under test and holds a constant final value at the end of the desired simulation interval that reflects the average power consumed.

2.7 CONCLUSIONS

This chapter has introduced the basics of computer-aided circuit analysis programs, and in particular has developed the MOS models used in the SPICE2G.6 program. Details on the convergence problems with SPICE have been explained and the experience obtained during the course of this
research to circumvent this problem has been presented. Finally a unique method for power measurement, from the literature, has been introduced. This discussion is not exhaustive, the literature pertaining to this subject is quite voluminous. Currently most research in this area is directed towards modelling small geometry MOSFET's using a semi-empirical modelling approach [13], [17]-[18], with parameters that are extracted using automated systems.

Finally it is commonly agreed upon that circuit simulator results are only as good as the data that is input to them. This is true, but it tends to give the circuit designer a sense that if he/she inputs good data then the results must be positively correct. Examining the literature it becomes evident that close attention should be paid to simulation results. Problems have been reported about inaccuracies in the SPICE MOS models. Namely non-conservation of charge, omission of the capacitances associated with the depletion region under the channel for small-signal models and numerous programming errors. For these reasons it is strongly suggested that close attention is paid to all simulation results.
3.1 INTRODUCTION

The discussion of the preceding section has brought some insight into the design procedure required to formulate a VLSI solution, at the circuit level. This process was based on the designers' knowledge of the theory and operation of the MOSFET, and also the efficient use of an analog circuit simulator. This chapter will investigate the design procedure at the system or architectural level initially, and then at the circuit level to determine the most viable solution, for a given problem.

The problem selected is in an area which members of the VLSI Group at the University of Windsor are actively pursuing. A class of pipelined VLSI
architectures, based on operations over finite rings. The class of algorithms mapped to VLSI architectures are based on a single operation common to many digital signal processing (DSP) applications, the inner product step. This operation is symbolically given below: [24]

\[ C \leftarrow C + A \cdot B \]  \hspace{1cm} (3.1)

which essentially performs a multiply with accumulation.

A particular structure introduced by Taheri [25] will be discussed below and several variations of his design will be presented.

3.2. A FINITE RING BIT LEVEL INNER PRODUCT STEP PROCESSOR (BIPSP\textsubscript{m})

Before the cell introduced by Taheri is discussed a brief summary of the work done by previous researchers [26]-[27], will be introduced, which laid the framework for the finite ring cell.

It has been demonstrated that the systolic array principle can be applied at the bit level. Providing an increase in throughput, the reason being that the circuit is pipelined at the bit level rather than the word level. Typically the bit-level cell consists of a full adder, some simple logic, and a number of latches. The power of this standard cell is that the function it performs is determined by the structure of the array it is located in. The above mentioned bit level cell is derived from the inner product step processor (IPSP) by slicing it into bits, each cell comprised of a full adder with four inputs \( a_{in}, y_{in}, x_{in} \) and
c_{in} and four outputs \( a_{out}, y_{out}, x_{out}, \) and \( c_{out} \). The operation of the Bit Level Inner Product Step Processor (BIPSP) is described by: [26]

\[
\begin{align*}
    y_{out} &= y_{in} \oplus_2 c_{in} \oplus_2 (a_{in} \oplus_2 x_{in}) \quad \tag{3.2a} \\
    c_{out} &= \text{Maj}(y_{in}, a_{in}, c_{in}) \quad \tag{3.2b} \\
    a_{out} &= a_{in} \quad \tag{3.2c} \\
    x_{out} &= x_{in} \quad \tag{3.2d}
\end{align*}
\]

where \( \text{Maj}(x_i) \) selects the boolean element in majority and \( \oplus_2 \) is the exclusive OR boolean function (XOR) and \( \oplus_2 \) is the boolean AND function. Figure 3.1 depicts the above mentioned cell [25]. It is possible to connect several of these cells in a two dimensional array which will perform word level operations [26].

Figure 3.1 The BIPSP cell
The above study is far from exhaustive, it's purpose is only to introduce the subject and demonstrate the origin of the cell proposed by Taheri. Interested readers should refer to the literature [26], [25], [28].

The structure introduced by Taheri is equivalent to the above mentioned BIPSP, except that the basic arithmetic function performed by each processor is calculated over a given ring. However, implementing this arithmetic over any general ring using standard binary logic gates is not a trivial task. Subsequently the hardware for each cell must be modified for each ring, resulting in a non-regular footprint for the hardware realization. For this reason a ROM is used to implement the required arithmetic function. The cell proposed by Taheri for a four bit operand is shown in Figure 3.2. The cell consists of a ROM which uses the input variable as an address, a set of steering switches and pipeline latches. With the cell operating over a finite ring both the inputs and outputs are bounded to a B bit representation, where B is a function of the chosen ring. Due to this arithmetic property over finite rings the size of the ROM required is small in comparison to that for a binary system [Jullien], resulting in efficient cell area. Also shown in figure 3.2 are a set of pipeline latches that allow the cell to be incorporated into a linear array, and a collection of steering switches. The purpose of the latter will be given below in a discussion of the cells operation. The operation of the fixed multiplier BIPSP_m (where the subscript m is included to enforce that the operation is over a ring) can be defined by: [25]
Figure 3.2 Finite ring BIPSP

\[
y^{(i+1)} = y^{(i)} \oplus_m [A \otimes_m x^{[i]} \otimes_m 2^i]
\]

\[
x^{(i+1)} = x^{(i)}
\]

where it is the spatial array index which determines the input and output of each cell, \( A_{in} \) is the fixed multiplier that is programmed into each ROM cell and \( x^{[i]} \) is the \( i \)-th bit of \( X_{in} \). The \( x^{[i]} \) bit is available at the \( i \)-th cell because of the switches present in each cell that cyclically shift \( X_{in} \). In general the cell computes:
For \( x^{(i)}[0] = 1 \):
\[
y^{(i+1)} = y^{(i)} \oplus_m [2^i \oplus_m A]
\]
where 3.4 a) is computed by the ROM and 3.4 b) uses the value of \( x^{[i]} = 0 \) to enable the steering switches to pass the input to the output. To reinforce the technique for 5-bit modular addition using the cell shown in figure 3.2 an example will be given below.

**Example 3.1**

Consider adding two five bit numbers \( A \) and \( B \):

\[
(A + B)_{\text{mod}_m} = ( [A_{4}A_{3}A_{2}A_{1}A_{0}] + [B_{4}B_{3}B_{2}B_{1}B_{0}] )_{\text{mod}_m}
\]
\[
= (2^4A_4 + 2^3A_3 + 2^2A_2 + 2^1A_1 + 2^0A_0 + B)_{\text{mod}_m}
\]

decomposing the above into five partial sums:

\[
\begin{align*}
\text{sum}_1 &= (2^0A_0 + B)_{\text{mod}_m} \\
\text{sum}_2 &= (2^1A_1 + \text{sum}_1)_{\text{mod}_m} \\
\text{sum}_3 &= (2^2A_2 + \text{sum}_2)_{\text{mod}_m} \\
\text{sum}_4 &= (2^3A_3 + \text{sum}_3)_{\text{mod}_m} \\
\text{sum}_5 &= (2^4A_4 + \text{sum}_3)_{\text{mod}_m}
\end{align*}
\]

The five stage representation of this decomposition is shown in Figure 3.3. Where the inputs to each stage are \( B \), \( \text{sum}_1 \), \( \text{sum}_2 \), \( \text{sum}_3 \) and \( \text{sum}_4 \) respectively, representing the address to a precalculated ROM location. This figure also depicts the ROM contents and switch closures for the addition of \( X = 11 \) and \( Y = 27 \), with a modulus of 32. From this figure one can also see how a fixed multiplication could be accounted for in each ROM.
Figure 3.3 Hardware implementation of a five bit BIPSP\textsubscript{m} system
As with any pipelined structure, there is some latency to the system. The latency for the five bit array illustrated above is five clock cycles. However, once the pipeline is full (after 5 cycles), a valid sum is present at the output every cycle. Now that a brief description of the BISP$^m$ cell has been accomplished, a specific implementation of the cell in the VLSI arena will follow.

3.3. A VLSI IMPLEMENTATION OF BISP$^m$

The BISP$^m$ cell described in the previous section has been implemented in a double metal three micron CMOS process. A high level representation of the implementation is shown in Figure 3.4. It differs slightly from that proposed in Figure 3.2 [25], to permit an efficient physical layout. The major differences being that the pipeline latches are distributed and the cyclic shift occurs at the output latches rather than the input.

Figure 3.4 Block diagram of the BISP$^m$ implementation
As seen in Figure 3.4, the cell is comprised of two major functional components, the ROM array and the pipeline latches. The details of the cell and these components at the circuit level will not be discussed here, only its functional operation. The interested reader may refer to Appendix B for a full set of schematics. The clocking scheme for the cell is based on a two phase non-overlapping clock. During \( \phi_1 \), the latches located before the ROM activate, thus capturing the data from the previous stage. Once valid data is present, the row and column decoders begin to evaluate. While the decoder is evaluating, \( \phi_2 \) initiates the precharge of the ROM's bit lines, and also passes the \( A_{in} \) variable to be shifted. After the bitlines have been precharged the row and column select lines either cause the parasitic capacitance of the bitlines to hold the charge, or if a transistor is present at the selected site the line will be discharged through the site nMOS transistor. Figure 3.5 depicts the timing requirements for the implemented design, obtained from extensive SPICE2 simulations. Where the geometric input parameters required by SPICE2 have been extracted from the mask, included in Appendix B, using the MASKAPII software [29]. Physical semiconductor parameter are those supplied by the Canadian Microelectronic Corporation (CMC).

![Figure 3.5 BiPSP\(_m\) timing requirements](image)
Through extensive simulation the functionality of the cell has been verified and an estimated throughput rate of 43 MHz has been obtained. The critical path for the design has been determined to consist of only a NAND gate, two inverters and a transmission gate in series [Jullien]. Table 3.1 summarizes the characteristics of the implemented cell.

<table>
<thead>
<tr>
<th></th>
<th>Memory Cell</th>
<th>Decoders</th>
<th>Latches</th>
<th>Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>250μ x 150μ</td>
<td>60000μ²</td>
<td>90000μ²</td>
<td>420μ x 378μ</td>
</tr>
<tr>
<td>Number of Transistors</td>
<td>160</td>
<td>107</td>
<td>88</td>
<td>355</td>
</tr>
<tr>
<td>Max. clock Rate</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>43 MHz</td>
</tr>
<tr>
<td>Function</td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>C ← C+A*B</td>
</tr>
<tr>
<td></td>
<td>——</td>
<td>——</td>
<td>——</td>
<td>where A is fixed</td>
</tr>
</tbody>
</table>

Table 3.1 Characteristics of the implemented cell

The preceding portion of this chapter has reviewed the pertinent work of several researchers belonging to the VLSI Research Group at the University of Windsor (Taheri, Bird, Jullien). In summary an architecture has been proposed and implemented that performs a finite ring IPSP at the bit level, where the multiplier is fixed.

However it has been suggested (Jullien) that some investigation into developing/modifyng the existing cell such that it would perform a finite ring IPSP at the bit level, with a multiplier that could be field programmed or
even changed on the fly. The remainder of this work will focus on this problem. Several possible solutions will be presented in the remainder of this chapter, from which one will be chosen and implemented in the following chapter.

3.4 DISCUSSION OF THE PROBLEM

Since all operation in the $\text{BIPSP}_m$ cell take place over a finite ring, the result of operations on a B bit operand will result in a B bit solution. Therefore as explained previously the ROM array required for the $\text{BIPSP}_m$ cell is area efficient. The typical cell contents for a five bit operand modulus thirty-two is shown in figure 3.6 a)

![Arrays](image)

Figure 3.6 ROM array for a 5-bit system

The locations have been arranged in a two-dimensional array of eight rows and four columns. This arrangement has proven to produce an area of favorable dimension for hardware implementation (roughly the same dimensions in each direction). The storage cell for the ROM requires only
one nMOS transistor at each site, this results in a very efficient layout area as seen in the mask (Appendix B). The ROM array occupies approximately 250 X 150 microns (using a 3 micron double metal CMOS process). As seen from figure 3.6 b) the cell can also be used for any moduli $m_i$, where $m_i \leq 32$. The power of the RNS is that fixed operations, such as multiplication, may be precalculated and programmed into the ROM at fabrication, resulting in a free operation, hardware-wise [30]. However once these fixed operations have been incorporated in the design procedure the function of the cell is also fixed. Therefore, it is desirable to be able to field program these fixed operations after the final fabrication of the chip. Several solutions to this problem will be given below.

3.5 ROM BASED DESIGNS

As seen from the tables shown in figure 3.6 all possible values can be stored in a ROM lookup table. For a B bit operand all entries can be stored in an array containing $2^B$ locations. This B bit word size will also accommodate any moduli less than $2^B - 1$, as seen in figure 3.6 b). It is evident that all required information is available in the ROM cell, only a method of addressing the array based on the predetermined function must be found. In other words a value which we will call the offset must also be supplied to the ROM. This value causes the first entry and subsequent entries to be shifted. Table 3.2 illustrates this for a modulus of five.
Table 3.2 Offset index relationship for a modulus 5 system

3.5.1 DESIGN I

The simplest method of implementing this offset is shown in figure 3.7. In this figure a modₘ adder is used to modify the input address Bₘ to the ROM by simply adding the offset. Based on previous research [31][32] it has been found that for modulo in the range that we require (5-bit), a pipelined ripple-carry adder is desirable. This selection has been arrived at by using the minimization of the Area.Period product as the design criterion. The architecture of a general RNS adder is illustrated in figure 3.8 [33]. Two binary adders are required to perform modulo addition. The first adder computes Adder_one = A + B, while the second adder computes Adder_two = A + B - M. The carry generated by the second adder is used as a control to the multiplexer, which chooses the correct solution. To achieve a high throughput rate, an area trade off must be made from the architecture exhibited in figure 3.8. The block diagram for this architecture, which requires two independent adders, is demonstrated in figure C.1, with the corresponding 3µ CMOS layout shown in figure C.2, both of which are in appendix C. For the sake of completeness, and as a first order approximation
of the utility of this design, simulations and mask layouts are presented in appendix C. From these simulations, it was found that the maximum throughput rate is determined by the worst case delay through any of the cells, since the adder is fully pipelined. This maximum delay was found to be in the Sum output, which was dependent on the evaluation of the carry. As explained previously the critical path for the Taheri/Bird design was small and located in the address decoders [Jullien]. However, the critical path will be dramatically increased by addition of the adder in series with this path. Table 3.3 below depicts the characteristics of this design using a fully pipelined array of fulladders [31].

![Diagram](image)

Figure 3.7 Offset is accommodated by using a modulo m adder
Operation of this modified cell is similar to that depicted in figure 3.4. The ROM array, steering switches and the cyclic shift all operate in the same manner. However, the input to the ROM must now be added to the offset before it may be used as an address to the ROM. This introduces latency in input path due to the pipeline structure of the adder.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Fast modula adders have been developed for the CMC process (ripple-carry) [31].</td>
<td>- Increase in area approx. 2200 X 1000 microns, or an increase in transistors of approx. 500.</td>
</tr>
<tr>
<td>- Allows the cell to be programmed on the fly without stopping the pipeline.</td>
<td>- Critical path is increased dramatically.</td>
</tr>
<tr>
<td>- Storage cell requires one transistor per storage bit.</td>
<td>- Adder pipelines must be full prior to operation of the BiPSPm pipeline.</td>
</tr>
<tr>
<td>- The latch required on the ROM input is incorporated in the pipelined adder.</td>
<td>- The modulus is fixed at fabrication time. Even though the adder may be programmed, the ROM is fixed at fabrication.</td>
</tr>
</tbody>
</table>

Table 3.3 Characteristics of Design I
3.5.2 DESIGN II.

The second ROM based design uses the cyclic property of modulo arithmetic to perform the index function, by shifting the row and column decoders accordingly. As illustrated in figure 3.9 the three-to-eight and two-to-four multiplexers must also incorporate a shift register with wraparound capabilities. Extra circuitry must also be included to decode the binary representation of offset into a control signal, compatible with the shift registers. This circuitry would take the binary representation of the offset as an input and produce a pulse stream as an output. Also required is a control circuit that detects a circulation of a one in the row shifter and then passes an extra shift signal to the column shifter. This design requires at best zero extra
cycles (an offset of zero) to shift. At worst a maximum of nine cycles is required to sequentially shift the row decoder eight times (offset greater than eight) along with an extra shift of the column decoder if the one in a field of zeros shifts around to the least significant row bit. The column decoder contains only four bits, thus it may be shifted in parallel with the row decoder. However, an extra shift may be required as explained above. The structure of the cell remains similar to that of the previous cell (figure 3.4) except that the row and column decoders required for the ROM (not depicted in figure 3.4) will be modified to incorporate the shift register required. A block diagram of these modifications is illustrated in figure 3.9.

Two designs have been considered initially for the eight stage shift register, both basic cells containing eight transistors. This design does not incorporate the parallel load function required, however, it is a starting point for a feasibility study. A single stage of each design is shown in figure 3.10. Both are dynamic designs, but, design a) has been found to operate at a higher clock rate with a limited drive capability. Since it is not required to drive a large load this is quite adequate. As a result of comprehensive SPICE simulations design a) has been found to operate with a shift cycle time of 23 ns; however, up to nine of these shifts could be required for each cell cycle. In fact, since the cells are pipelined it is not tolerable to have cells with different offsets operating at varying throughput rates. For this reason each cell would have to wait the maximum time possible for all shift units to finish their required operation. This is nine shifts resulting in a delay of 207 ns, or a decrease in throughput rate of almost five hundred percent, this is not justifiable. The
delay of the pulse generator to evaluate the input bits has not been considered and it would also be included in series with critical path. However, the efficiency of this design does increase as the number of columns approaches the number of rows, since shifting of each may occur in parallel.

Figure 3.9 Design II block diagram

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Simple system design and arbitration.</td>
<td>- The pipeline would have to wait for the longest shift of all cells belonging to the pipeline.</td>
</tr>
<tr>
<td>- Small increase in overall cell area.</td>
<td>- Critical path is dramatically increased approx. 500%.</td>
</tr>
<tr>
<td>- Row and column shifts may occur in parallel.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4 Characteristics of Design II
3.5.3 DESIGN III

The third ROM based design also uses the cyclic property of modula arithmetic to perform the index function. However, to overcome the cell cycle time constraint introduced by the serial shifting of the row and column selectors, a parallel shift module has been investigated. The module is an implementation of a barrel shifter with wraparound capability. This type of parallel shifter has been used to implement operand normalization, operand alignment and operand unpack/pack operations in mainframes for several years [34]. The structure for an nMOS implementation of such a module is illustrated in figure 3.11, where site transistors are present for shift$_0$ and shift$_1$ only. Essentially the shifter is a cross-point switch, whereby the control lines shift$_0$ - shift$_7$ are connected to the site transistors as described by 3.6:
Essentially each individual nMOS site transistor acts as a cross point switch, where each \textit{SWITCH}_{ij} connects bus \( a_i \) to output \( b_j \). The time required for an
arbitrary shift consists of two components; the time required for the offset to be decoded into the shift signal; the time required for the site transistor to charge/discharge the output lines. To conserve silicon either pMOS or nMOS pass transistor may be present at each cross point. The utilization of pMOS devices will degrade the passage of logic zero's, while the application of nMOS devices will inhibit the transmission of logic one's.

Figure 3.12 3μ CMOS mask layout of 8 x 8 barrel shifter
However the data present in the row and column selectors is a one in a field of zero's, for this reason a pMOS device would be expected more effective for this application. Except, the mobility ($\mu_n$) of electrons is higher than the mobility ($\mu_p$) of holes by a factor of two or more, resulting in p-devices having to be at least twice as wide to switch at a speed comparable to a n-device. From extensive SPICE simulation, it was determined that to reduce the Area.Period product of the module, nMOS site transistors need to be utilized. SPICE simulations for the nMOS design depicted in figure 3.13 are presented in figure 3.14. The geometric information required by SPICE has been extracted from the mask layout using the MASKAPII software. From this figure it can be seen how the nMOS device degrades the logic one level to approximately three volts. However this may be circumvented by following the shifter with a noninverting buffer. This layout has also been simulated and from these simulations the cycle time for a shift has been determined. To make effective use of the nMOS technology the output lines should be precharged high prior to evaluation of the shift. The result of this, would be excess power dissipation since all the row select transistors would be activated while the ROM precharged pull-up drivers were also active, creating a momentary path between $V_{DD}$ and $V_{SS}$. This dissipation may be acceptable but could only be determined from extensive SPICE power simulations using the techniques described in chapter two. The remaining component of the shift delay is the time required for the n-bit binary representation of the offset to be decoded into a 1-of-n signal compatible with the barrel shifter. Since this decoding may take place in parallel with the ROM address decoding it does not contribute to the overall cell cycle time.
### Table 3.5 Characteristics of Design III

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Simple system design and arbitration.</td>
<td>- Increase in cell area of 300 x 300 microns.</td>
</tr>
<tr>
<td>- Magnitude comparison and shift decode may occur in parallel with address decode.</td>
<td>- The modulus is fixed at fabrication time. Even though the offset may be programmed, the ROM is fixed at fabrication.</td>
</tr>
<tr>
<td>- Row and column shifts may also occur in parallel.</td>
<td></td>
</tr>
<tr>
<td>- Shift cycle is not dependent on the offset size.</td>
<td></td>
</tr>
</tbody>
</table>

#### Figure 3.13 SPICE simulation of nMOS barrel shifter
From the above investigation it is evident that the barrel shifter is a much more efficient means of shifter, for large offsets. But does not lend itself to simple detection of wraparound as the serial shifter did. A block diagram of the required modifications to the cell is shown in figure 3.15. In contrast to the previous design a magnitude comparison is required to implement the additional shift in the column decoder, if required. But this comparison could also occur in parallel with the ROM address decoding and precharge of the ROM bit lines.

![Design III block diagram](image)

Figure 3.14 Design III block diagram

3.5.4 DESIGN IV

The final ROM based structure concentrated on using a non-binary coded number representation of the operands. It was anticipated that a
representation could be derived that would allow the operand and offset to be combined with nominal combinatory logic, and possibly a parallel shift. After several coding methods had been investigated it was determined that the overhead required for the increased word length, required for any coding technique investigated was excessive. For this reason non-binary coding methods will no longer be considered in this work.

3.6 RAM BASED DESIGNS

As explained previously all possible values for modula arithmetic can be stored in a lookup table [30]. For a B bit operand all entries can be stored in an array containing $2^B$ locations. This B bit word size will also accommodate any moduli less than $2^B - 1$. It is evident that all required information can be stored in a lookup table, only a method of addressing the array, based on the predetermined function must be found. In contrast to the ROM based designs discussed previously RAM based designs will also accommodate field programming of the modulus. This programmability results in a very flexible cell, however the price is paid by increased size and complexity. The remainder of this chapter will focus on designs that facilitate field programming of the modulus.

3.6.1 DESIGN V

This implementation is again based on the fact that under modulo arithmetic the result belongs to a finite cyclic group. As seen in table 3.6 any row in the modulus five addition table is a permutation of any other row.
The method proposed is a variation of an implementation for modulus addition and subtraction proposed by Banerji [35].

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 3.6 Modulo five addition table

From table 3.56 it is seen that the $i^{th}$ element of the $j^{th}$ row corresponds to the sum $i + j \mod 5$. In this design the row, containing all residues $0$ to $m-1$ is stored in a register. Whose contents is available to the input bus of a barrel shifter. The offset will then be decoded into a signal compatible with the shift input of the barrel shifter. This will cause the linear storage element (barrel shifter) to be shifted with wrap around as desired. The result of this shift will cause the lookup table to be aligned as required according to the offset. The address would then be used to select the correct location. An illustration of this method is shown in figure 3.15, for the addition of five and four modulo five.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Simple system design and arbitration.</td>
<td>- Dramatic increase in area. Barrel shifter alone is 1200$\mu$X 6000$\mu$.</td>
</tr>
<tr>
<td>- Modulus can be field programmed.</td>
<td></td>
</tr>
<tr>
<td>- Throughput rate is comparable to the Bird implementation.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.7 Characteristics of Design V
A block diagram of the cell required for this design is illustrated in figure 3.16. The operation of the cell is similar to the previously described ROM based cells, and uses a two phase non-overlapping clocking scheme. During $\phi_1$, the latches located before the shift module activate, thus capturing the data from the previous stage. At this same time the modulo and offset would also be latched into the cell. Once valid data is present the shift decoder and 1-of-(m-1) selector would begin to evaluate. During $\phi_2$ the output bus of the barrel shifter would be precharged high, also the $A_{in}$ variable would be shifted and passed to the output of the cell. Prior to the occurrence of the next cycle the barrel shifter would evaluate presenting the shifted data at it's output.

The throughput rate of this cell is expected to be as great as the Bird implementation, however the Area.Period product is estimated to be several times that implementation. The major contribution to this cells area is the barrel shifter module which consists of five 32 by 32 shifter modules, covering an estimated area of 1200$\mu$m x 6000$\mu$m. Also required are a 5-to-32 decoder, 32...
latches for the modulo register and the 1-of-32 selector required for the output.

3.6.2 DESIGN VI

The final implementation investigated was the natural extension of the ROM based BISP cell. It was anticipated at the onset of this project that a novel implementation could be derived to reduce the redundant information stored in the ROM. But as seen in the previous designs none proved to be an eloquent solution. Basically this design replaces the static memory element used for computations (ROM) with a dynamic, or field
programmable memory element. Such an element is commonly referred to as a Random Access Memory (RAM) or more accurately a Read/Write (R/W) memory. When such a structure is incorporated into the proposed linear array, several unique problems arise. The remainder of this section will focus on these problems and present a new structure which forms a completely programmable systolic cell. Incorporation of this cell into an array will also form a programmable array, for testing purposes [25].

If one looks to the original design depicted in Figure 3.2 it is obvious that data flows from left to right. Where the \( y^{(i)} \) data stream is used as the address to the computational memory, or steered around the cell, exiting on the right as \( y^{(i+1)} \). If the cell contained a R/W memory as shown in Figure 3.17 the operation would be the same during computation, however to initialize the cell (programming), data and address would have to be supplied from opposite ends of the structure.

![Diagram](image)

**Figure 3.17 Required data path for initialization of a programmable structure**
This configuration is the same as used in commercial R/W memories. The ramification of this to the BIPSP_m cell is that the output drivers and latches would have to at least double in complexity to accommodate the imposed bi-directionality of the port.

If this extra complexity is deemed acceptable there is still the issue which arises when hundreds of these cells are cascaded in a linear array, immediate access to internal cells is not available. This problem is shown in Figure 3.18, where the latches have not been shown to simplify the diagram. Here it is seen that the steering switches could be used to circumvent undesired cells. However, the complexity of the input drivers and latches would have to at least double to allow for the required bi-directionality.

Figure 3.18 Required data path for initialization of a programmable array structure

A solution to the problem of bi-directional data flow during initialization of the array, is to broadcast all write data to each cell and let the cells
discriminate whether the data is destined for them, or another cell in the array. One way to do this is to give each cell its own unique address. A system controller would first broadcast the address of the desired cell, each cell would then interrogate this address to determine if the next block of data was meant for them. Once the desired cell was selected the controller would broadcast the required write data. The obvious disadvantage of such a technique is that each cell must be given a unique address at fabrication. Also the broadcast data path would have to be wide enough (bit-wise) to allow for a unique address for each cell. For example if there were 256 cells in an array the broadcast data path would have to be at least eight bits wide.

A scheme has been devised that does not require an address to uniquely identify the cell to be written to. This scheme is similar to the Token Ring used in digital communications. In this scheme a cell is said to be active or non-transparent when it is in possession of a token which circulates around the network. The implementation of this token ring using the modified BIPSP$_m$ cell is shown in Figure 3.20 where a controller is used to insert the write address on the $Y_{in}$ lines and also the corresponding broadcast data on the broadcast lines. By adding complexity to the system controller it is possible to decrease the complexity of the cell hardware.

![Figure 3.19 Classic token ring structure](image)
For the proposed cell the main design criteria is to maintain the cell complexity with respect to the implemented BIPSP\textsubscript{m} cell [29]. In a typical application a large number of cells would be present on a chip and the controller could be amortized over a large number of cells. Figure 3.21 shows a more detailed view of two cells, of the type shown in Figure 3.20. The token repository can be viewed as a N-bit shift register which holds on to a token for N write cycles, were N is equal to the number of words present in the memory array.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Regular cell layout. Each cell is exactly the same.</td>
<td>- Storage cell requires six transistor per storage bit.</td>
</tr>
<tr>
<td>- Cell area is comparable to the Bird cell.</td>
<td>- Additional hardware required for the chip controller</td>
</tr>
</tbody>
</table>

Table 3.8 Characteristics of Design VI
This allows each cell, containing N words, to be written before the next cell’s memory element is initialized. Resulting in the linear array being initialized sequentially from left to right as shown in Figure 3.20. When a token enters a repository the transmission gates (tgates) are enabled, allowing access to the broadcast data stream. Similarly when a token exits a repository the tgates are deactivated, isolating the cell from the broadcast data stream. With only a single token cycling through the linear array and controller loop only one cell at a time has access to the broadcast data stream. A preliminary evaluation of this design has taken place and satisfies the major design criteria, a summary of this design is shown in Table 3.8.

From the explained cell operation it is evident that the required controller will have to incorporate a minimum intelligence. Which could sequence the \( Y_{in} \) (write address) and broadcast data (address contents) in a pattern which is determined by the desired lookup table contents. Basically the controller would consist of two \( \left( \log_2 N \right) \) bit modulo counters and a minimum amount of control circuitry for token processing. The function of the controller is shown below for both initialization and computation.

**WRITE**

- Controller Sequences Address on \( Y_{in} \)
- Controller Supplies the \( X_{in} \) data stream to select the required address path
- Data is broadcast through all cells
- Token is supplied to first cell by controller
COMPUTE

- Controller transparently passes $Y_{in}$ data to first cell
- Controller transparently passes $X_{in}$ data to first cell
- Token is not supplied
- Broadcast data is irrelevant

A preliminary 3μ CMOS realization [36] was implemented using a standard six transistor memory element, resulting in a cell size of unacceptable magnitude for our application. In the original BIPSP$_m$ implementation [29] the memory

\footnote{Assuming static memory which does not require periodic refresh.}
element accounted for 40% of the cell transistors and a small 20% of the cell area! This is due to the fact the ROM used required a single transistor which form a very dense array.

To reduce the overall size of the programmable cell it is obvious that a more area efficient storage element would have to be utilized. The small size of the memory array, typically thirty-two words results in the common practice of minimizing storage cell size at the cost of larger peripheral circuits does not hold. This is due to the fact that there is a small number (compared to the Mega capacity commercial RAM's now available) of memory cells too amortize the peripheral circuitry over. For this reason the standard guideline of minimizing memory cell size is not completely valid in our application.

Several unique memory structure were investigated; a five transistor Schmitt trigger static cell [37]; self refreshing dynamic structures [38],[39]; a synchronous dynamic structure[] and a three transistor dynamic cell. The design which best suits this application and which is viable with the CMOS process available to us [36] appears to be the familiar three transistor dynamic RAM. However a method of refreshing the cell must now be provided with a 1/frequency of roughly 2ms. Typically this refresh is accomplished in hardware by a read-write cycle which reads a location contents and then updates the location by writing the value back. The overhead associated with this refresh involves both area and time. In the case of a large linear systolic array containing hundreds of cells the area overhead would have to be replicated in each cell. A novel method has been devised which lowers the throughput rate of the cell but only marginally increases the cell area by taking advantage of the controller previously
mentioned. In this design the controller is used to interleave computational data and refresh address eliminating the normally required read-write cycle.

The system architecture is similar to that depicted in Figure 3.20-3.21 except the system controller must contain a buffer to achieve synchronization of the input data stream \(Y_{in}\) and the refresh address. Typically a FIFO is used for such synchronization \([40]\) and is also used in this design. Initialization of the area is as previously described. However refresh is accomplished during computation by interleaving refresh addresses with the data stream, while broadcasting the desired location contents. The necessary modification to Figure 3.21 are shown in Figure 3.22, where the \(X_{in}\) data supplied with the refresh data is now a field of zeroes. The additional OR gate is used to select the cell currently being refreshed. In the final design the token repository is no longer an N-bit shift register but rather a D latch which is clocked by the system controller accordingly:

- every N cycles for initialization
- every \(N \times \text{refresh rate}\) cycles for refresh

The timing for a computation-refresh-computation period is shown in Figure 3.23. From preliminary simulations it is expected, using the target 3\(\mu\) CMOS process, where 25MS/sec is an upper limit for cell operation that 156 cells are capable of being refreshed with only a 10% loss in computation rate.
Figure 3.22 Architecture of dynamic programmable BIPSP<sub>m</sub> cell

Figure 3.23 System timing for the programmable cell
3.7 CONCLUSIONS

The purpose of this chapter was to introduce the BIPSP\textsubscript{m} cell proposed by Taheri [25] and realized in the VLSI realm by Bird [29]. Several possible architectures have been presented which accommodated field programmability of the BIPSP\textsubscript{m} cell. Of the six preliminary designs it has been determined that the final implementation (design VI) appears to be the most favorable, given the 3\textmu CMOS process [36] available to the VLSI Research Group. The next chapter will investigate different structures at the hardware level to implement the already defined architecture.
CHAPTER 4

VLSI IMPLEMENTATION OF A PROGRAMMABLE SYSTOLIC CELL

4.1 INTRODUCTION

As explained in the previous section this chapter's purpose is to elaborate on the design deemed most suitable, given the process at our disposal. The architecture has been specified in the previous chapter, now each component will be investigated independently at the circuit level. As discussed in chapter three an efficient memory element is critical for reducing cell area. For this reason it will be investigated in depth in the next section. The remaining cell components; decoders, transmission gates, latches, etc., will only be presented briefly, the interested reader should refer to the references [29].

4.2 READ/WRITE MEMORIES

Random access memories, or more accurately Read/Write memories have a write cycle that is comparable in time to a read cycle. This is in contrast to the read only memory (ROM) used in the BISP_\text{m} cell, or even
other commercially programmable memories, such as EPROM's [56]. Basically, semiconductor read/write memory is *volatile*, meaning that when power is removed the information stored is lost. Read/Write memories are commonly divided into two categories, either static or dynamic. Several examples of memories falling into either category are depicted in Figure 4.1. Basically static memories will retain their information for as long as the power is supplied to the cell. The price for this convenience is larger circuit area and increased power dissipation, with fast access time. In contrast to static memories, dynamic cells typically require less cell area and smaller power requirements. However each memory location requires access within a rigid time interval or the information being stored may be lost. As explained in the previous chapter it is expected from preliminary layouts that a three transistor dynamic cell will be most efficient for the programmable BIPSP$_m$. However a formal investigation of this is deemed important due to the fact that a comparative study has not been found in the literature for small arrays, as the type desired in a programmable BIPSP$_m$.

4.2.1 STATIC MEMORY STRUCTURES

Truly static memory devices store information using a bi-stable circuit design. Information is stored as a result of the state of conduction or non-conduction of a transistor. The classic static storage element uses two load devices and two switching transistor configured as a flip-flop. Such an arrangement is shown in Figure 4.1 a), access transistors are added to select the desired memory element. Variations of the six transistor memory
Figure 4.1 Read/Write memory structures
element are abundant in the literature from multi-ported RAM's [41] to content addressable memories [42], based on this structure. Of interest to this work is the five transistor structure [43] shown in Figure 4.1 b).

Aside from the obvious area saving of one less transistor, than in the cell of Figure 4.1 a), there is also a saving in the required transistor dimension of the branch without the access transistor. This is a result of the bit line current only flowing in one of the branches of the flip-flop. Allowing the unused branch to contain transistors of minimum size, since they are unloaded and do not effect the bit line current. The active branch transistors however must be scaled to prevent undesired write operation during a read cycle. Typically the p-load transistors in both branches can be of minimum size, as they only supply current to offset the effects of leakage at the gates of the switching devices. The access transistors contribute to the capacitive loading of the bit lines and for this reason are also chosen as minimum size. This leaves the switching nmos transistors to be scaled accordingly. Typically a conductance ratio of the switching transistor to access transistor of 1.5 to 5 is chosen [44]. From simulations it was determined a ratio of two was required for the target CMOS process [36].

In a typical large memory array the cell area must be as small as possible. Typically the cell area is reduced by adding a sense amplifier in each column. This amplifier is used to source current to the bit lines resulting in the cell transistors having to only sink current. It also increases the speed of operation by sensing small differential voltage on the bit lines. A sense amplifier is essentially a flip-flop similar to that present in the memory cell,
however the transistor dimensions are usually larger. The advantage of using a small memory array is that a sense amplifier is not required because of the reduced bitline capacitance of the relatively short bitlines. A five transistor static memory cell has been designed using the target 3μ CMOS process and is presented here in Figure 4.2. Corresponding characteristics obtained from SPICE simulations, using parameters extracted from the mask, are depicted in Figure 4.3.
A complete test structure containing thirty-two five bit words has been completed for fabrication to determine the performance of the fabricated memory and can be found in Appendix D.

4.2.2 DYNAMIC MEMORY STRUCTURES

In contrast to the previously mentioned static memories, dynamic memories store information by presence or lack of electrons (charge) on a capacitor. Due to the nature of these capacitances, typically parasitic, the electrons tend to leak across device junctions. The result of this leakage is that the memory must be accessed within a rigid time frame, or data may be corrupted. Several flavors of dynamic R/W memories are available, the two most common are shown in Figure 4.1 d) and e). There are several other configurations commonly available, however they will not be presented here and the interested reader is referred to the large body of work available on this subject.

4.2.2.1 Three Transistor Dynamic Memory

The basic operation of the three transistor R/W memory can best be explained using Figure 4.4 as a reference. During a write operation the bitline (BIT) is set to the desired state either high or low, then the Write line is activated causing a transfer of charge from the bit line to the parasitic gate capacitance at node B. The Write line is then deactivated resulting in a charge storage at node B. During a Read operation the bitline is first precharged and then the Read line is activated causing the bitline to remain precharged or to be pulled low through the series nmos transistors. Other than the obvious advantage of two fewer transistors than the
previously examined static R/W memory this memory employs minimum size devices, resulting in an obvious area saving. The disadvantages of the circuit is that it requires two separate read/write lines and the pulldown is not of what is typically referred to as a safe design [45].

The latter results from charge sharing which occurs when node A is a low logic level from the previous read operation and must now be charged to a logic high resulting in an imperfect logic high on the bitline. This problem has not been found to be critical during simulations, mainly because the bitline capacitance is much larger in magnitude than the parasitic capacitance present at node A. A solution to the former problem of two bitlines has been addressed in the literature [46],[47].
The scheme devised requires the generation of an intermediate voltage to drive the memory select line. The added complexity of the peripheral circuitry has been deemed not efficient for such a small array. The CMOS realization of the three transistor dynamic cell is shown in figure 4.5. Figure 4.6 presents the simulated results, again using parameters extracted from the mask, here the charge sharing at node A is evident. A test structure has also been designed which will allow testing of the cell alone. Details of this test layout are included in Appendix D.

![Three Transistor RAM Timing](image)

Figure 4.6 SPICE simulations of three transistor R/W memory cell

4.2.2.2 Single Transistor Dynamic Memory

The one transistor memory cell is typically by far the most efficient in cell area. This structure is shown in Figure 4.1 e), where the capacitor is no longer parasitic as in the three transistor cell. The nmos transistor is used as a switch to allow the desired bitline access to the correct storage capacitor. In
contrast to the previously mentioned cells readout of this cell is *destructive*; the amount of charge transferred into or out of the cell corrupts the original value stored. The result being that each read cycle must also perform a refresh operation immediately. The overall performance of the cell is determined by charge stored in the storage capacitor. Since the capacitor is charged by a source follower the maximum voltage stored on the capacitor is $V_{DD} - V_{TH}$, where $V_{TH}$ is the threshold voltage of an NFET. For the target 3μ CMOS process at our disposal this is typically 0.70 volts [36] resulting in a maximum voltage stored on the capacitor of 4.30 volts (5.0 - 0.70). For typical memory arrays the storage capacitance is much larger than the bitline capacitance resulting in a minor voltage change on the bitline reflecting a logic level change. This requires sense amplifiers required on each bit line.

From the above discussion it is obvious that the reduced cell array comes at the expense of increased peripheral circuits. The disadvantages of the single transistor cell, which arise from the elimination of the amplifying transistor used in the 3-transistor cell are [48]-[50]:

1) a parasitic capacitor, such as used in the 3-transistor cell (the gate) cannot be used as a storage capacitor.

2) the readout of the cell is destructive.

3) the output of the cell is damped so that a sensitive refresh amplifier is required.

For these reasons it is not expected that such an array would be practical for the small array required in a programmable BIPSP$_m$ cell. Also the fact that the target CMOS process supported by CMC is a P-well process results in an
inefficient design. This is caused by the separation required between the capacitor P-doped device well and the P-well (CMC rule G.3 - 16 μ sep.) [36].

4.3 SUMMARY OF MEMORY DESIGNS

The results from the above comparisons are summarized below in Table 4.1:

<table>
<thead>
<tr>
<th>Memory Structure</th>
<th>Number of Transistors/Cell</th>
<th>Cell Area</th>
<th>Array Area</th>
<th>Support Circuitry Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static</td>
<td>5</td>
<td>2016μ²</td>
<td>352.800μ²</td>
<td>none</td>
</tr>
<tr>
<td>Dynamic</td>
<td>3</td>
<td>1472μ²</td>
<td>168.480μ²</td>
<td>11.340μ²</td>
</tr>
</tbody>
</table>

Table 4.1 Comparison of memory structures

From this work it has been determined that a three transistor dynamic structure is most efficient for use in a programmable BIPSPₘ cell. However if an N-well process were available the single transistor dynamic cell will be begin to become more efficient. Also if the CMOS process supported merged contacts the three transistor cell would become smaller from the more aggressive spacing of the write transistor source and the pulldown transistor gate.

4.4 CELL DESIGN

With the optimum memory array structure selected and implemented for the programmable BIPSPₘ cell all that is left is to assemble the standard design entities to realize the architecture shown in figure 3.22. The new
architecture consists of all the elements used in the BIPSP$_m$ implemented by Bird plus a D-latch and an OR gate. All of which have been investigated by previous members of the VLSI Research Group [51], [29].

Details of the design may be found in Appendix F, however a preliminary feasibility layout has been performed and is shown in figure 4.7 compared to the BIPSP$_m$ cell implemented by Bird.

4.5 CONCLUSIONS

This chapter investigated the possible memory structures available for use in the architecture proposed in chapter 3. Test structures were created at the mask level and fully verified at the chip level with the available software. A preliminary layout was created to compare cell area against that of the ROM based implementation with favorable results.
Figure 4.7 Comparison of programmable cell (bottom) vs BIPSP \_m (top)
CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 CONCLUSIONS

The primary aim and objective of this thesis has been to formalize the simulation step in the design process and also to present a new architecture for implementing a programmable version of the reported inner product step processors (IPSP), operating over a finite ring (BIPSP_m). The structure is based on a decomposition of residue number system (RNS) tabular structures which takes advantage of the associativity property of an IPSP with fixed coefficients.

The motivation of this research resulted from the large number of linear DSP algorithms which can be implemented using computational elements consisting of IPSP. Using the previously reported decomposition, a fixed coefficient implementation has shown more than an order of magnitude improvement in (Area.Period) product over its binary counterpart. In fact, the bit-level decomposition is so efficient that the entire computational table may be trivially generated from prior knowledge of the first entry. The main limitation with the previous implementation is the requirement for fixed coefficient multiplication, resulting from the fact that the computational
tables must be generated at fabrication. The ramification of this is two-fold; chip re-design is required to implement coefficient changes; yield degradation is imposed by the inability to route around bad cells in the linear array which are spatially required to compute the summation required by the decomposition.

The natural extension of the ROM based BIPS\(_m\) cell would be to incorporate a read/write memory to replace the ROM array. However, the small size of the memory array, typically thirty-two words, violates the principles used in commercial size RAM design, i.e. minimizing storage cell array at the cost of larger peripheral circuits. The fact that hundreds of small cells are incorporated into linear pipelined arrays, removes access to internal cells for programming. Several unique memory structures were investigated: a five transistor Schmitt trigger static cell; self refreshing dynamic structures; three transistor and single transistor dynamic cells. The design which was determined to best suit this application and which is viable with the CMOS process available appears to be the familiar three transistor dynamic RAM. The three-transistor implementation used was the 2X-1Y 3 device/bit cell. The advantage of this cell for the programmable BIPS\(_m\) is that a common R/W data bus is used. The fact that separate R and W control lines are required is of no consequence because of the small size of the array.

With the use of a dynamic memory structure, refresh must occur within a rigid time frame or data corruption may occur. Typically, this refresh is accomplished in hardware by a read-write cycle which reads the contents of a selected location and then updates by writing the value back. The overhead
associated with this refresh involves both area and time. In the case of a large
linear systolic array the area overhead would be replicated in each cell.

A novel method, presented in this thesis, lowers the throughput rate of the
cell (up to a 10% decrease in throughput) but only marginally increases the
cell array, by taking advantage of a global system controller. In this scheme
the controller is used to interleave computational data and refresh address
eliminating the need for the normally required read-write cycle. This
architecture relegates most of the refresh operation externally, resulting in an
efficient cell area and allowing the cost of the controller to be amortized over
hundreds of cells.

The ramification of the presented design is that a general programmable
systolic array has been created, which by nature, lends itself to incorporate
fault tolerant techniques. Also, bad cells may now be routed around,
resulting in higher yields at the chip level, by using redundant cells on chip
in a similar fashion as used in large memory arrays.

A proof of concept cell has been developed using a 3μ double metal CMOS
process. The results of preliminary simulations have shown that a 10%
decrease in throughput rate is required for refresh, compared to the ROM
based BIPS  

A summary of the areas investigated during this work is shown below:

i) Investigation and comparison into several circuit simulators:

- SPICE2G.6
- RELAX2.2
- SPLICE1.2
Presentation of techniques to aid in convergence of SPICE simulation decks.

iii) Overview of a fixed coefficient BIPSP$_m$ implementation. This involved testing and power simulations.

iv) Investigation into several memory structures for use in a programmable BIPSP$_m$ implementation.

v) Introduction of a unique architecture for refresh of pipelined dynamic memory.

5.2 RECOMMENDATIONS

During the course of this research several deficiencies or enhancements were discovered that could make for a more efficient design process. Some of these recommendations were beyond the scope of this work while others were not within the time frame of this thesis.

i) Investigation into the WATAND circuit simulator. This simulator is of the same maturity as SPICE however it has been reported that the source code is more documented, allowing for the user to implement his/her own models.

ii) Investigation into the new version of SPICE (SPICE3B). The source code for this new version is 'C'. Resulting in similar benefits to WATAND, i.e. for user implemented models.

iii) Parameter extraction of simulator transistor parameters from experimentally obtained results.

iv) Standard cell verification of large array designs using MASKAPII.

v) Investigation into EPROM implementation using the single level polysilicon CMC process. This structure may lead to an interesting intermediate implementation between the ROM and RAM BIPSP$_m$ implementations.

vi) Investigation into behavioral and synthesis languages, such as 'VHDL' and 'V' respectively. VHDL is an IEEE
standard and is outlined in the document IEEE STD VHDL-1076 Language Reference Manual. The V language was developed at IBM T.J. Watson Research Centre and is one of the first languages to address the special needs of hardware synthesis.
REFERENCES


APPENDIX A

COMPARISON OF THE CIRCUIT SIMULATORS

1. SPICE2G.6
2. SPLICE1.7
3. RELAX2.2
APPENDIX A

SIMULATOR COMPARISONS

This appendix contains the circuit used in the comparison of the three simulators. The circuits used are a boot-strapped inverter and a ring oscillator. The inverter was used to determine if all simulators using an equivalent model equation would produce similar results. The oscillator circuit was chosen because it has been shown in the literature that circuits with local feedback cause convergence problems when using relaxation techniques. It was also available as a layout provided by the Canadian Microelectronic Corporation (CMC) which was used to verify the extraction module written for the MASKAPII software. The results for each circuit are given below.

A.1 Boot-Strapped Inverter

The schematic used for the boot-strapped inverter is shown below in Figure A.1. It was chosen for two reasons, first to verify all simulator models and secondly to show that the newer waveform relaxation techniques do not have the problems related to circuits containing large floating capacitances as did their predecessors.

![Boot-Strapped Inverter Circuit](image)

Figure A.1 Boot-Strapped inverter circuit

As can be seen from Figure A.2 SPICE and SPLICE produce results that are similar. Except for time points near zero, as explained in section 2.3. However the RELAX simulator produced results that differ greatly around
seventy-five nanoseconds this may be attributed to the small number of model parameters available to the user.

Figure A.2 Boot-Strapped inverter simulation results
SPICE BOOT-STRAPPED INVERTER

* VERSION: SPICE2.G.6  *
* SITE: UNIVERSITY OF WINDSOR  *
* DATE: JUNE 24, 1988  *
* PROGRAMMER: B. ERICKSON  *

* POWER SUPPLY & INPUT SIGNAL  *

VD 1 0 DC 5
VIN 2 0 PWL(0NS 0 15NS 0 20NS 5 40NS 5 45NS 0 65NS 0
+ 70NS 5 90NS 5 95NS 0)

* NETLIST  *

M1 1 3 4 0 N
M2 1 3 0 N
M3 4 2 0 0 N
C1 3 0 .01P
C2 4 0 .01P
C3 3 4 .10P

* MODELS FOR CMOS 3 MICRON DOUBLE METAL PROCESS  *

.MODEL N NMOS ( LEVEL=1 VTO=1.0 KP=20E-6
+ GAMMA=0.31 PHI=0.6 LAMBDA=0.02)

* ANALYSIS OPTIONS  *

PRINT TRAN V(4)
TRAN 1NS 95NS
.END

Figure A.3 Boot-Strapped inverter SPICE deck.
splice

Boot-strapped inverter example

;**********************************************************************
; version: SPLICE1.7
; site: University of Windsor
; date: June 24, 1988
; programmer: B. Erickson
;**********************************************************************

;**********************************************************************
; Models
;**********************************************************************

.model tran nmos : (w=5u l=5u vto=1.0VS
kp=20u gamma=0.31 phi=0.6 lambda=0.02)
.model gcap gcapr : 0.01pf
.modelfcap fcapr : 0.10pf
.model vdd dc : 5V
.model src tsrc : 0V 5V 0n 50ns 0n 15ns 20ns $40ns 45ns

;**********************************************************************
; Netlist
;**********************************************************************

vdd 1  vdd
input 2 src
m01 1, 3, 4 tran
m02 1, 1, 3 tran
m03 4, 2, 0 tran
c01 3 gcapr
c02 4 gcapr
c03 3, 4 fcap

;**********************************************************************
; Analysis options
;**********************************************************************

.time 1ns 95ns
.print 4
.go
.end

Figure A.4 Boot-Strapped inverter SPLICE deck
splice
Boot-strapped inverter example

/*
   ************************************************************************
   */
   version: SPLICE1.7
   site: University of Windsor
   date: June 24, 1988
   programmer: B. Erickson
   ************************************************************************
   Models

   .model tran nmos : (w=5u l=5u vto=1.0V $
   kp=20u gamma=0.31 phi=0.6 lambda=0.02)
   .model gcav gcapr : 0.01pf
   .model fcav fcapr : 0.10pf
   .model vdd dc : 5V
   .model src tsr : 0V 5V 0n 50ns 0n 15ns $
   20ns 40ns 45ns
   *
   ************************************************************************
   *
   Netlist
   ************************************************************************
   *
   vdd 1 vdd
   input 2 src
   m01 1, 3, 4 tran
   m02 1, 1, 3 tran
   m03 4, 2, 0 tran
   c01 3 gcav
   c02 4 gcav
   c03 3, 4 fcav
   *
   ************************************************************************
   *
   Analysis options
   ************************************************************************
   *
   .time 1ns 95ns
   .print 4
go
.end

Figure A.5 Boot-Strapped inverter RELAX deck
A.2 Ring Oscillator

The ring oscillator circuit used is included on the test strip provided by the CMC. The mask for this circuit is shown below in figure A.6 and schematic is shown in the following figure A.7. Extracted simulator decks for both SPICE and SPLICE are shown in the figures A.7 and A.8. The parameters for these decks have been extracted by the MASKAPII program and modules written by members of the VLSI group. However to make the decks more readable they have been compiled by hand to make use of the hierarchical structure allowed by both simulators, this makes for a deck that is much easier to interpret.

The simulation results where only obtained for the SPICE program. Convergence problems occurred when using the other simulator. This may be the result of the local feedback encountered in a ring oscillator as mentioned in the literature. The simulated results have also been compared to experimentally determined results from a fabricated test cell with surprisingly good comparison.
Figure A.7 Ring oscillator circuit

SPICE RING OSCILLATOR CIRCUIT

* VERSION: SPICE2G.6
* SITE: UNIVERSITY OF WINDSOR
* DATE: JUNE 24, 1988
* PROGRAMMER: B. ERICKSON

* Power supply

* VCC 100 0 DC 5V

* Ring oscillator circuit description

* X1 100 1 2 INVERT
X2 100 2 3 INVERT
X3 100 3 4 INVERT
X4 100 4 5 INVERT
X5 100 5 6 INVERT
X6 100 6 7 INVERT
X7 100 7 8 INVERT

Figure A.8 SPICE input deck
X8 100 8 9 INVERT
X9 100 9 10 INVERT
X10 100 10 11 INVERT
X11 100 11 12 INVERT
X12 100 12 13 INVERT
X13 100 13 14 INVERT
X14 100 14 15 INVERT
X15 100 15 16 INVERT
X16 100 16 17 INVERT
X17 100 17 18 INVERT
X18 100 18 19 INVERT
X19 100 19 20 INVERT
X20 100 20 21 INVERT
X21 100 21 22 INVERT
X22 100 22 23 INVERT
X23 100 23 1 INVERT

ANALYSIS OPTIONS
* NOTE: UIC card was used because d.c. non-convergence occurred.

OPTION LIMPTS=2000 PIVTOL=1.0E-14
.TRAN .5NS 200NS UIC
.PRINT TRAN V(6)
.IC V(100)=5.0V

Inverter subcircuit for ring oscillator

.SUBCKT INVERT 1 2 3

-Vcc node is 1
-Gate is node 2
-Output is node 4

M1 1 2 3 1 PTRAN W=0.24000E-04 L=0.30000E-05
+AD=0.17280E-09 AS=0.13860E-09 PS=0.61200E-04
+PD=0.62400E-04 NRD=0.30000+00 NRS=0.27500E+00

M2 3 2 0 0 NTRAN W=0.12000E-04 L=0.30000E-05
+AD=0.66600E-10 AS=0.77760E-10 PS=0.37200E-04
+PD=0.38400E-04 NRD=.60000E+00 NRS=.55000E+00

Parasitic node capacitance

CN1 3 0 0.93234E-13
.ENDS INVERT

Figure A.8 cont.
Figure A.8 cont.

splice
splice ring oscillator circuit

;***********************************************************************
; Power supply
;***********************************************************************
vc 100dc : 5v
.model inputA tsrc : v0=5.0, v1=0.0, d=0ns, $
 p=500ns, 0.45ns, 50ns, 500ns
input1 1 inputA

;***********************************************************************
; Ring oscillator circuit description
;***********************************************************************
x1 100 1 2 invert
x2 100 2 3 invert
x3 100 3 4 invert
x4 100 4 5 invert
x5 100 5 6 invert
x6 100 6 7 invert
x7 100 7 8 invert
x8 100 8 9 invert
x9 100 9 10 invert

Figure A.9 SPLICE input deck
inverter subcircuit for ring oscillator

.model invert subckt: (1 2 3)

- Vcc node is 1
- Gate is node 2
- Output is node 4

.model ntran nmos: (vto=0.70 kp=50.0u $
gamma=phi=0.6 lambda=0.01)
.model ptran pmos: (vto=-0.80 kp=16.0u $
gamma=-0.6 phi=0.6 lambda=0.03)
.model fcapr gcapr : 0.93234E-13

m1 1 2 3 ptran : w=0.24000E-04 l=0.30000E-05
m2 3 2 0 ntran : w=0.12000E-04 l=0.30000E-05

Parasitic node capacitance

c1 3 fcapr
.ends invert

Control statements

.time 1ns 100ns
.print 1, 10
.go
.end

Figure A.9 cont.
A.3 Simulation Results Compared to Fabricated Device Measurements

A test insert consisting of a strip of test structures is added by CMC to most multiproject chips prior to fabrication [23]. The purpose of this strip is to verify the integrity of the process and also gives the design engineer a set of standard cells which he/she may use to extract process parameters for use in simulations. The structures included in the CMC test strip are:

- CMOS Ring Oscillator
- Several Van der Pauw Structures
- CMOS Contact Tester
- CMOS Process NPN Transistor
- CMOS Layer Tester
- CMOS N-Channel Transistor
- CMOS P-Channel Transistors
To verify SPICE2G.6 results, input decks were extracted from the n-channel transistors and the model parameters supplied by CMC for level one and two were used with results given in Figure A.12, A.14. As expected the level two results are more accurate than level one, however the level three results appear to be less accurate than level two. The reason for this is that as explained in section 2.5.3 the level three model is semi-empirical with parameters extracted from d.c. curves, using a curve fitting technique. But CMC does not provide these parameters and the defaults were used for the results shown in Figure A.16. Better results are expected if a curve fitting utility is developed and this has been suggested to current undergraduate students as a research project. The n-channel test structure is shown in Figure A.11, the device used was D, G and S1 which is a single transistor with $W/L = 48/24 \mu$. 

Figure A.11 Double Level Metal CMOS N-Channel Transistors Test Structure
Figure A.12 Level One SPICE Results vs Experimental Results

W/L = 48/24 NTRAN IV CURVES
* SPICE LEVEL=1

MN9 2 13 0 NTRAN W= 0.48000E-04 L= 0.24000E-04
+AD= 0.21600E-09 AS= 0.37440E-09 PS= 0.11160E-03 PD= 0.57000E-04
+NRD= 0.93750E-01 NRS= 0.16250E+00

* Parasitic Node Capacitors Follow

CN1 1 0 0.28451E-11
CN5 3 0 0.74144E-13

* Parasitic Coupling Capacitors Follow

CC1 1 0 0.54540E-14
CC2 1 2 0.16200E-14
CC4 2 0 0.22680E-14

Figure A.13 SPICE Input Deck for Level=1
Figure A.13 (cont.)

I-V Characteristics

Figure A.14 Level Two SPICE Results vs Experimental Results
APPENDIX A

W/L = 48/24 NTRAN IV CURVES
* SPICE LEVEL=2

MN9 2 1 3 0 NTRAN W=0.48000E-04 L=0.24000E-04
+AD=0.21600E-09 AS=0.37440E-09 PS=0.11160E-03 PD=0.57000E-04
+NRD=0.93750E-01 NRS=0.16250E+00

* Parasitic Node Capacitors Follow

CN1 1 0 0.28451E-11
CN5 3 0 0.74744E-13

* Parasitic Coupling Capacitors Follow

CC1 1 0 0.54540E-14
CC2 1 2 0.16200E-14
CC4 2 0 0.22680E-14

VSS 3 0
VDS 2 0
VGS 1

* STANDARD OPTIONS FOR RUN CONTROL OF SPICEY

.DC VDS 0 5 0.1 VGS 1 4 1
.PRINT DC I(VSS)
.WIDTH IN=72 OUT=133
.OPTIONS LIMPTS=500
.OPTIONS NOMOD

* STANDARD CMC CMOS PROCESS PARAMETERS

.MODEL NTRAN NMOS(TOX=50.0N KP=50U NSUB=1.7E16 XJ=0.6U
+ VTO=0.7 UF=775.0 LAMBDA=0.01 GAMMA=1.1 RD=0 RS=0
+ VMAX=1.0E7 PBE=1.0J5=10.0U LD=0.35U PHI=0.6 TPG=1
+ RSH=25.0 CJ=4.4E-4 MJ=0.5 CJSW=4.0E-10
+ MJSW=0.3 CGSO=3.0E-10 CGDO=3.0E-10 CGBO=5.0E-10
+ THETA=0.11 KAPPA=1.0 ETA 0.05
+ LEVEL=2)
.END

Figure A.15 SPICE Input Deck for Level=2
Figure A.16 Level Three SPICE Results vs Experimental Results

Figure A.17 SPICE Input Deck for Level=3
*Parasitic/Coupling Capacitors Follow*

CC1  1  0  0.54540E-14  
CC2  1  2  0.16200E-14  
CC4  2  0  0.22680E-14  

VSS  3  0  
VDS  2  0  
VGS  1  0  

*STANDARD OPTIONS FOR RUN CONTROL OF SPICE*

.DC VDS 0 5 0.1  VGS 1 4 1  
.PRINT DC I(VSS)  
.WIDTH IN=72 OUT=133  
.OPTIONS LIMITS=500  
.OPTIONS NOMOD  

*STANDARD CMC CMOS PROCESS PARAMETERS*

.MODEL NTRAN NMOS( TOX=50.0N  KP=50U  NSUB=1.7E16  XJ=0.6U  
+ VTO=0.7  UO=775.0  LAMBDA=0.01  GAMMA=1.1  RD=0  RS=0  
+ VMAX=1.0E7  PB=0.7  JS=10.0U  LD=0.35U  PHI=0.6  TFG=1  
+ RSH=25.0  CJ=4.4E-4  MJ=0.5  CJSW=4.0E-10  
+ MJSW=0.3  CGSO=3.0E-10  CGDO=3.0E-10  CGBO=5.0E-10  
+ THETA=0.11  KAPPA=1.0  ETA 0.05  
+ LEVEL=3)  
.END
APPENDIX B

VLSI IMPLEMENTATION OF BIPSP\textsubscript{m}
This appendix contains the realization of the architecture proposed by Majid Taheri [25] and originally fabricated by Peter Bird [29]. The implementation presented here has been modified by Alphonse Bastien. This chip has been fabricated and extensively tested and simulated with favorable results. Simulated results are shown below operating at a clock frequency of 43 MHz. The techniques described in chapter two have been utilized to simulate the entire cell, (all 355 transistors) a great increase in the previous limit of eighty-six transistors. The simulation shown in Figure B.1 shows the output $B_{out}$ with respect to the clocks, when the outputs ($B_{out}$) are fed back to the inputs ($B_{in}$). This cell is presented here as a starting point for this research.
THE QUALITY OF THIS MICROFICHE IS HEAVILY DEPENDENT UPON THE QUALITY OF THE THESIS SUBMITTED FOR MICROFILMING.

UNFORTUNATELY THE COLOURED ILLUSTRATIONS OF THIS THESIS CAN ONLY YIELD DIFFERENT TONES OF GREY.

LA QUALITE DE CETTE MICROFICHE DEPEND GRANDEMENT DE LA QUALITE DE LA THESE SOUMISE AU MICROFILMAGE.

MALHEUREUSEMENT, LES DIFFERENTES ILLUSTRATIONS EN COULEURS DE CETTE THESE NE PEUVENT DONNER QUE DES TEINTES DE GRIS.
Figure B.2 Mask for the BIPSPₘ implementation
Figure B.3 Schematic page one - Address Decoding and Latches
APPENDIX C

FIVE-BIT PIPELINED RNS ADDER
APPENDIX C

FIVE-BIT PIPELINED RNS ADDER

This appendix contains the results of previous researchers belonging to the VLSI group at the University of Windsor [Sayfo], [Vorjdani], [Bayoumi]. Presented here is a CMOS layout for a modula five-bit adder. The block diagram for this adder is shown in figure C.1, and is an implementation of the architecture shown in figure 3.10. Based on previous research it has been determined that for five-bit moduli a pipelined-ripple carry implementation is most efficient. The moduli adder consists of two independent adders that calculate, Adder_{one} = A + B, and Adder_{two} = A + B - M. The carry generated by Adder_{two} is then used as a control to the multiplexer, which selects the correct output (i.e. carry=0: Sum → A + B). From figure C.1 it is obvious that changing the moduli is trivial, all that is required is to apply the two's complement of the desired moduli to the inputs of Adder_{two}.

An implementation of such an adder, in a 3μ CMOS process is shown in figure C.2 [Sayfo]. The adder is comprised of approximately five hundred transistors and occupies an area of 2200 by 1000 microns. As with any fully pipelined design the maximum throughput rate is determined by the worst case delay through any element within the design. This critical path has been determined from SPICE2G.6 analog simulations to be within the fulladder element. Specifically the Sum output which is dependent on the evaluation of the carry circuitry first. These simulations were obtained by using geometric data extracted from the mask layout shown in figure C.3 [Sayfo]. Results of the simulations are shown in figure C.4 a) - d) where the data rate has been varied to determine the maximum speed at which the cell may operate. Figure C.4 a) exhibits the critical path of the cell, while C.4 b) determines the minimum time required for valid data to be present on the input to produce valid output data. The remaining two plots represent valid data times that are not large enough.
Figure C.1 Block diagram a 5-bit modula adder
Figure C.2 Five bit modula adder test chip
Figure C.3 Full adder layout
Figure C.4 Simulation results for the fulladder cell
**Full-Adder Circuit**

- Ci
- a
- b
- C
- S

![Graph](image)

**Figure C.4 (cont.)**

vii
** Full-Adder Circuit **

*VERSION: SPICE2G.6*
*SITE: UNIVERSITY OF WINDSOR*
*DATE: OCT 12, 1988*
*PROGRAMMER: B. ERICKSON*

*Power supply*

VDD 110 DC 5V

*INPUTS*

VCIN 7 0 PWL(0NS 0V 2NS 0V 3NS 5V 5NS 5V 6NS 0V)
VB 1 0 PWL(0NS 5V 2NS 5V 3NS 0V 5NS 0V 6NS 5V)
VA 5 0 PWL(0NS 5V 2NS 5V 3NS 0V 5NS 0V 6NS 5V)

*N-Channel Enhancement Mosfets Extracted from Mask Layout Follow.*

MN1 4 1 3 0NTRAN W=0.30000E-05 L=0.30000E-05
+AD= 0.21780E-10 AS= 0.21780E-10 PS= 0.15600E-04 PD= 0.15600E-04
+NRD= 0.21000E+01 NRS= 0.21000E+01

MN2 6 1 5 0NTRAN W=0.30000E-05 L=0.30000E-05
+AD= 0.22680E-10 AS= 0.26280E-10 PS= 0.18600E-04 PD= 0.16200E-04
+NRD= 0.22000E+01 NRS= 0.26000E+01

MN3 1 5 6 0NTRAN W=0.30000E-05 L=0.30000E-05
+AD= 0.24480E-10 AS= 0.22680E-10 PS= 0.16200E-04 PD= 0.17400E-04
+NRD= 0.24000E+01 NRS= 0.22000E+01

MN4 3 5 0 0NTRAN W=0.30000E-05 L=0.30000E-05
+AD= 0.21780E-10 AS= 0.43200E-10 PS= 0.30000E-04 PD= 0.15600E-04
+NRD= 0.21000E+01 NRS= 0.40000E+01

MN5 1 3 4 0NTRAN W=0.30000E-05 L=0.30000E-05
+AD= 0.24480E-10 AS= 0.21780E-10 PS= 0.15600E-04 PD= 0.17400E-04
+NRD= 0.24000E+01 NRS= 0.21000E+01

MN6 8 4 7 0NTRAN W=0.30000E-05 L=0.30000E-05
+AD= 0.28980E-10 AS= 0.28980E-10 PS= 0.20400E-04 PD= 0.20400E-04
+NRD= 0.29000E+01 NRS= 0.29000E+01

MN7 7 6 9 0NTRAN W=0.30000E-05 L=0.30000E-05
+AD= 0.28980E-10 AS= 0.28980E-10 PS= 0.20400E-04 PD= 0.20400E-04
+NRD= 0.29000E+01 NRS= 0.29000E+01

Figure C.5 Extracted SPICE deck of Fulladder cell
### MN8 9 4 10 0 NTRAN

- W = 0.30000E-05
- L = 0.30000E-05
- \( AD = 0.28980E-10 \)
- \( AS = 0.21780E-10 \)
- \( PS = 0.15600E-04 \)
- \( PD = 0.20400E-04 \)
- \( NRD = 0.29000E+01 \)
- \( NRS = 0.21000E+01 \)

### MN9 8 6 5 0 NTRAN

- W = 0.30000E-05
- L = 0.30000E-05
- \( AD = 0.28980E-10 \)
- \( AS = 0.26280E-10 \)
- \( PS = 0.18600E-04 \)
- \( PD = 0.20400E-04 \)
- \( NRD = 0.29000E+01 \)
- \( NRS = 0.26000E+01 \)

### MN10 10 7 0 0 NTRAN

- W = 0.30000E-05
- L = 0.30000E-05
- \( AD = 0.21780E-10 \)
- \( AS = 0.42840E-10 \)
- \( PS = 0.28800E-04 \)
- \( PD = 0.15600E-04 \)
- \( NRD = 0.21000E+01 \)
- \( NRS = 0.38000E+01 \)

* P-Channel Enhancement Mosfets Extracted from Mask Layout Follow.

---

### MP11 3 1 6 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.30600E+00 \)
- \( AS = 0.30600E+00 \)
- \( PS = 0.16200E-04 \)
- \( PD = 0.16200E-04 \)
- \( NRD = 0.85000E+00 \)
- \( NRS = 0.85000E+00 \)

### MP12 4 1 5 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.32400E-10 \)
- \( AS = 0.39600E-10 \)
- \( PS = 0.19200E-04 \)
- \( PD = 0.16800E-04 \)
- \( NRD = 0.90000E+00 \)
- \( NRS = 0.11000E+01 \)

### MP13 1 5 4 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.36000E-10 \)
- \( AS = 0.32400E-10 \)
- \( PS = 0.16800E-04 \)
- \( PD = 0.18000E-04 \)
- \( NRD = 0.10000E+00 \)
- \( NRS = 0.90000E+00 \)

### MP14 3 5 11 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.30600E-10 \)
- \( AS = 0.70200E-10 \)
- \( PS = 0.36000E-04 \)
- \( PD = 0.16200E-04 \)
- \( NRD = 0.85000E+00 \)
- \( NRS = 0.20000E+01 \)

### MP15 6 3 1 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.30600E-10 \)
- \( AS = 0.36000E-10 \)
- \( PS = 0.18000E-04 \)
- \( PD = 0.16200E-04 \)
- \( NRD = 0.85000E+00 \)
- \( NRS = 0.10000E+01 \)

### MP16 7 6 8 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.45000E-10 \)
- \( AS = 0.45000E-10 \)
- \( PS = 0.21000E-04 \)
- \( PD = 0.21000E-04 \)
- \( NRD = 0.12500E+01 \)
- \( NRS = 0.12500E+01 \)

### MP17 10 6 9 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.30600E-10 \)
- \( AS = 0.45000E-10 \)
- \( PS = 0.21000E-04 \)
- \( PD = 0.16200E-04 \)
- \( NRD = 0.85000E+00 \)
- \( NRS = 0.12500E+01 \)

### MP18 8 4 5 11 PTRAN

- W = 0.60000E-05
- L = 0.30000E-05
- \( AD = 0.45000E-10 \)
- \( AS = 0.39600E-10 \)
- \( PS = 0.19200E-04 \)
- \( PD = 0.21000E-04 \)
- \( NRD = 0.12500E+01 \)
- \( NRS = 0.11000E+01 \)

---

**Figure C.5 cont.**
**APPENDIX C**

```
MP19  9  4  7  11 PTRAN  W=0.60000E-05  L=0.30000E-05
+AD=0.45000E-10  AS=0.45000E-10  PS=0.21000E-04  PD=0.21000E-04
+NRD=0.12500E+01  NRS=0.12500E+01

MP20  11  7  10  11 PTRAN  W=0.60000E-05  L=0.30000E-05
+AD=0.63000E-10  AS=0.30600E-10  PS=0.16200E-04  PD=0.33600E-04
+NRD=0.18000E+01  NRS=0.85000E+00

*****************************************************************************
* Non-Parasitic Capacitors Extracted from    *
* Mask Layout Follow.                     *
*****************************************************************************

*****************************************************************************
* Parasitic Node Capacitors Follow    *
*****************************************************************************
CN1  1  0  0.88138E-13
CN3  3  0  0.39026E-13
CN4  4  0  0.11261E-12
CN5  5  0  0.92704E-13
CN6  6  0  0.90600E-13
CN7  7  0  0.58233E-13
CN8  8  0  0.50000E-13
CN9  9  0  0.50000E-13
CN10 10  0  0.53598E-14
CN11 11  0  0.26515E-13

*****************************************************************************
* Parasitic Coupling Capacitors Follow    *
*****************************************************************************
CC1  1  4  0.39600E-14
CC2  4  5  0.13968E-13

* ANALYSIS OPTIONS*

.OPTION  LIMPTS=1000
.TRAN  .015NS  14.9NS
.PRINT  TRAN  V(7)  V(1)  V(5)  V(8)  V(9)

.MODEL  NTRAN NMOS(VTO=0.7  KP=40E-6  GAMMA=1.1  PHI=0.6
+LAMBDA=0.01  RD=40  RS=40  PB=0.7  CGSO=3.0E-10  CGDO=3.0E-10
+CGBO=5.0E-10  RSH=25  CJ=4.4E-4  MJ=0.5  CJSW=4.0E-10
+MJSW=0.3  JS=1.0E-5  TOX=5.0E-8  NSUB=1.7E16  TPG=1  XJ=6.0E-7
+LD=3.5E-7  UO=775  VMAX=1.0E5  LEVEL=1)

.MODEL  PTRAN PMOS(VTO=0.8  KP=12E-6  GAMMA=0.6  PHI=0.6
+LAMBDA=0.03  RD=100  RS=100  PB=0.6  CGSO=2.5E-10  CGDO=2.5E-10
+CGBO=5.0E-10  RSH=80  CJ=1.5E-4  MJ=0.6  CJSW=4.0E-10
+MJSW=0.6  JS=1.0E-5  TOX=5.0E-8  NSUB=5E15  TPG=1  XJ=5.0E-7
+LD=2.5E-7  UO=250  VMAX=0.7E5  LEVEL=1)

.END
```

Figure C.5 cont.
APPENDIX D

MEMORY TEST STRUCTURES

1. Five Transistor Static Cell
2. Three Transistor Dynamic Cell
APPENDIX D

MEMORY ARRAY TEST STRUCTURE DESIGNS

This Appendix contains the mask layouts and schematics for the test memory arrays. Two test arrays consisting thirty-two five bit words where designed using the target 3μ CMOS process. Schematics for the three transistor memory array have also been included as well as the LOGICV report file verifying complete logical-to-physical of the mask and schematic.

D.1 Three Transistory Dynamic Memory Array

Figure D.1 below depicts the mask layout for the three transistor test chip. Also incorporated into the test chip are several test structures for three transistor memories as well as some experimental photo transistors. The following figure represents the logic level schematic for the same test chip. Table 1 is the LOGICV report file to verify that a logical-to-physical mapping has occurred.

A.2 Five Transistor Static Memory Array

Figure D.3 shows the mask layout for the static memory test chip. The chip consists of a five bit thirty word test array as well as several five transistor test structure. Also incorporated into this chip are some experimental photo transistors.
Figure D.3 Five Transistor Memory Array Test Chip Mask
Figure D.3 Five Transistor Memory Array Test Chip Mask
CHIP NAME: TDCHIP

LOGIC VERIFICATION SEGMENT COMPARE REPORT
SEGMENT DEVICES
IN IN
ERROR SEGMENT NAME

SEGMENT COMPARE SUMMARY
SEGMENTS COMPARED: 1, SEGMENTS IN ERROR: 0, SEGMENTS
VERIFIED: 1
END OF REPORT
BOOLEAN CONNECTIVITY DISCREPANCY REPORT

END OF REPORT
DEVICE VERIFICATION REPORT

DEVICE: NTRAN
MATCH 0: 0 (DEVICES NOT CHECKED)
MATCH 1: 0 (DEVICES WHICH WERE NOT MATCHABLE)
MATCH 2: 0 (DEVICES VERIFIED PRIOR TO MISMATCH)
MATCH 3: 702 (DEVICES FULLY VERIFIED)

DEVICE: PTRAN
MATCH 0: 0 (DEVICES NOT CHECKED)
MATCH 1: 0 (DEVICES WHICH WERE NOT MATCHABLE)
MATCH 2: 0 (DEVICES VERIFIED PRIOR TO MISMATCH)
MATCH 3: 235 (DEVICES FULLY VERIFIED)

DEVICE: CAPCITOR
MATCH 0: 0 (DEVICES NOT CHECKED)
MATCH 1: 0 (DEVICES WHICH WERE NOT MATCHABLE)
MATCH 2: 0 (DEVICES VERIFIED PRIOR TO MISMATCH)
MATCH 3: 0 (DEVICES FULLY VERIFIED)

END OF REPORT
<table>
<thead>
<tr>
<th>LN</th>
<th>NODE</th>
<th>LOGIC NETWORK NODE NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>$001$</td>
<td>431</td>
<td>VDD</td>
</tr>
<tr>
<td>$002$</td>
<td>2</td>
<td>VSS</td>
</tr>
<tr>
<td>$003$</td>
<td>13</td>
<td>RS8</td>
</tr>
<tr>
<td>$004$</td>
<td>18</td>
<td>RS7</td>
</tr>
<tr>
<td>$005$</td>
<td>23</td>
<td>RS6</td>
</tr>
<tr>
<td>$006$</td>
<td>26</td>
<td>RS5</td>
</tr>
<tr>
<td>$007$</td>
<td>401</td>
<td>RS4</td>
</tr>
<tr>
<td>$008$</td>
<td>406</td>
<td>RS3</td>
</tr>
<tr>
<td>$009$</td>
<td>409</td>
<td>RS2</td>
</tr>
<tr>
<td>$010$</td>
<td>411</td>
<td>RS1</td>
</tr>
<tr>
<td>$011$</td>
<td>414</td>
<td>DATA4</td>
</tr>
<tr>
<td>$012$</td>
<td>413</td>
<td>DATA3</td>
</tr>
<tr>
<td>$013$</td>
<td>366</td>
<td>DATA2</td>
</tr>
<tr>
<td>$014$</td>
<td>172</td>
<td>DATA1</td>
</tr>
<tr>
<td>$015$</td>
<td>6</td>
<td>DATA0</td>
</tr>
<tr>
<td>$016$</td>
<td>34</td>
<td>RWBIT4</td>
</tr>
<tr>
<td>$017$</td>
<td>32</td>
<td>RWBIT3</td>
</tr>
<tr>
<td>$018$</td>
<td>30</td>
<td>RWBIT2</td>
</tr>
<tr>
<td>$019$</td>
<td>36</td>
<td>RWBIT1</td>
</tr>
<tr>
<td>$020$</td>
<td>38</td>
<td>RWBIT0</td>
</tr>
<tr>
<td>$021$</td>
<td>427</td>
<td>TEST1</td>
</tr>
<tr>
<td>$022$</td>
<td>424</td>
<td>TEST0</td>
</tr>
<tr>
<td>$023$</td>
<td>429</td>
<td>BIT</td>
</tr>
<tr>
<td>$024$</td>
<td>420</td>
<td>OUT5</td>
</tr>
<tr>
<td>$025$</td>
<td>416</td>
<td>OUT4</td>
</tr>
<tr>
<td>$026$</td>
<td>294</td>
<td>OUT3</td>
</tr>
<tr>
<td>$027$</td>
<td>40</td>
<td>OUT2</td>
</tr>
<tr>
<td>$028$</td>
<td>3</td>
<td>OUT1</td>
</tr>
<tr>
<td>$029$</td>
<td>10</td>
<td>RWBAR</td>
</tr>
<tr>
<td>$030$</td>
<td>7</td>
<td>RW</td>
</tr>
<tr>
<td>$031$</td>
<td>371</td>
<td>C34</td>
</tr>
<tr>
<td>$032$</td>
<td>369</td>
<td>C33</td>
</tr>
<tr>
<td>$033$</td>
<td>334</td>
<td>C32</td>
</tr>
<tr>
<td>$034$</td>
<td>332</td>
<td>C31</td>
</tr>
<tr>
<td>$035$</td>
<td>300</td>
<td>C30</td>
</tr>
<tr>
<td>$036$</td>
<td>326</td>
<td>C33</td>
</tr>
<tr>
<td>$037$</td>
<td>299</td>
<td>C24</td>
</tr>
<tr>
<td>$038$</td>
<td>256</td>
<td>C23</td>
</tr>
<tr>
<td>$039$</td>
<td>255</td>
<td>C22</td>
</tr>
<tr>
<td>$040$</td>
<td>211</td>
<td>C21</td>
</tr>
<tr>
<td>$041$</td>
<td>208</td>
<td>C20</td>
</tr>
<tr>
<td>$042$</td>
<td>245</td>
<td>C52</td>
</tr>
<tr>
<td>$043$</td>
<td>210</td>
<td>C14</td>
</tr>
<tr>
<td>$044$</td>
<td>175</td>
<td>C13</td>
</tr>
<tr>
<td>$045$</td>
<td>177</td>
<td>C12</td>
</tr>
<tr>
<td>$303</td>
<td>315</td>
<td>XXX20.XXX5.XXX2-D</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-------------------</td>
</tr>
<tr>
<td>$305</td>
<td>270</td>
<td>XXX20.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$307</td>
<td>274</td>
<td>XXX20.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$309</td>
<td>230</td>
<td>XXX20.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$311</td>
<td>231</td>
<td>XXX20.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$313</td>
<td>310</td>
<td>XXX24.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$315</td>
<td>269</td>
<td>XXX24.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$317</td>
<td>268</td>
<td>XXX24.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$319</td>
<td>225</td>
<td>XXX24.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$321</td>
<td>226</td>
<td>XXX24.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$323</td>
<td>309</td>
<td>XXX28.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$325</td>
<td>263</td>
<td>XXX28.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$327</td>
<td>267</td>
<td>XXX28.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$329</td>
<td>222</td>
<td>XXX28.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$331</td>
<td>223</td>
<td>XXX28.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$333</td>
<td>305</td>
<td>XXX32.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$335</td>
<td>258</td>
<td>XXX32.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$337</td>
<td>262</td>
<td>XXX32.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$339</td>
<td>217</td>
<td>XXX32.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$341</td>
<td>218</td>
<td>XXX32.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$343</td>
<td>301</td>
<td>XXX36.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$345</td>
<td>251</td>
<td>XXX36.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$347</td>
<td>257</td>
<td>XXX36.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$349</td>
<td>212</td>
<td>XXX36.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$351</td>
<td>213</td>
<td>XXX36.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$353</td>
<td>399</td>
<td>XXX39.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$355</td>
<td>400</td>
<td>XXX39.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$357</td>
<td>363</td>
<td>XXX39.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$359</td>
<td>364</td>
<td>XXX39.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$361</td>
<td>360</td>
<td>XXX39.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$363</td>
<td>394</td>
<td>XXX13.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$365</td>
<td>393</td>
<td>XXX13.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$367</td>
<td>360</td>
<td>XXX13.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$369</td>
<td>359</td>
<td>XXX13.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$371</td>
<td>325</td>
<td>XXX13.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$373</td>
<td>386</td>
<td>XXX17.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$375</td>
<td>385</td>
<td>XXX17.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$377</td>
<td>352</td>
<td>XXX17.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$379</td>
<td>351</td>
<td>XXX17.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$381</td>
<td>317</td>
<td>XXX17.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$383</td>
<td>388</td>
<td>XXX21.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$385</td>
<td>387</td>
<td>XXX21.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$387</td>
<td>350</td>
<td>XXX21.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$389</td>
<td>349</td>
<td>XXX21.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$391</td>
<td>312</td>
<td>XXX21.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$393</td>
<td>382</td>
<td>XXX25.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$395</td>
<td>381</td>
<td>XXX25.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$397</td>
<td>346</td>
<td>XXX25.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$399</td>
<td>345</td>
<td>XXX25.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$401</td>
<td>311</td>
<td>XXX25.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$403</td>
<td>304</td>
<td>XXX29.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$405</td>
<td>377</td>
<td>XXX29.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$407</td>
<td>344</td>
<td>XXX29.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$409</td>
<td>343</td>
<td>XXX29.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$411</td>
<td>306</td>
<td>XXX29.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$413</td>
<td>376</td>
<td>XXX33.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$415</td>
<td>373</td>
<td>XXX33.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$417</td>
<td>340</td>
<td>XXX33.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$419</td>
<td>339</td>
<td>XXX33.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$421</td>
<td>302</td>
<td>XXX33.XXX1.XXX2-D</td>
</tr>
<tr>
<td>$423</td>
<td>372</td>
<td>XXX37.XXX5.XXX2-D</td>
</tr>
<tr>
<td>$425</td>
<td>367</td>
<td>XXX37.XXX4.XXX2-D</td>
</tr>
<tr>
<td>$427</td>
<td>336</td>
<td>XXX37.XXX3.XXX2-D</td>
</tr>
<tr>
<td>$429</td>
<td>335</td>
<td>XXX37.XXX2.XXX2-D</td>
</tr>
<tr>
<td>$431</td>
<td>296</td>
<td>XXX37.XXX1.XXX2-D</td>
</tr>
</tbody>
</table>

**IALOGICV HAS COMPLETED NORMALLY**
APPENDIX E

STEPS TO AID IN CONVERGENCE OF SPICE2G.6
APPENDIX E

STEPS TO AID IN CONVERGENCE OF SPICE2G.6

A.1 D.C. NON-CONVERGENCE

Convergence errors in d.c. analysis are usually due to an error in specifying the input circuit. However circuits with positive feedback may exhibit d.c. convergence problems. The steps below are given to help d.c. convergence problems.

STEP 1 Use the LIST and NODE parameters on the .OPTION card to verify the circuit is correct.

STEP 2 Use the NODESET card to improve the initial guesses for each node. SPICE2G.6 assumes initial node voltages of zero, if NODESET is not used. Source stepping may be used to obtain reasonable node voltages (see section 2.6.1).

STEP 3 If positive feedback is present the OFF option should be used for all devices in the feedback path.

STEP 4 Re-labeling of the source and drain may aid in convergence. The drain current is used in the check for convergence whereas the source current is not.

STEP 5\(^1\) Include the UIC option on the .TRAN card making sure to use the .IC card for any sources. This step circumvents d.c. analysis.

NOTE: The remaining steps must be used with care. These steps modify the parameters used by the program to determine convergence.

STEP 6\(^1\) Reset the relative tolerance of the program. This is accomplished by changing the RELTOL = parameter on the .OPTION card. (default 0.1 percent)

\(^1\) When using these steps the user should pay close attention to the simulation results for validity.
STEP 7\textsuperscript{1} Reset the absolute current error tolerance by including the \texttt{ABSTOL=} parameter on the \texttt{OPTION} card. (default 1 picoamp)

STEP 8\textsuperscript{1} Reset the absolute voltage error tolerance by including the \texttt{UNTOL=} parameter on the \texttt{OPTION} card. (default 1 microvolt)

A.1 TRANSIENT NON-CONVERGENCE

Convergence errors in transient analysis usually result in the error message:

\texttt{INTERNAL TIMESTEP TOO SMALL IN TRANSIENT ANALYSIS}

This error is usually caused by nodes in the circuit under simulation changing rapidly from timestep \( t \) to \( t + \Delta t \). The steps below are given to help transient convergence problems.

STEP 1 Include the \texttt{UIC} option on the \texttt{TRAN} card, specifying sources only on a \texttt{IC} card. This lets the circuit gradually reach steady-state as if it had just been powered up.

STEP 2 Add 1 G\( \Omega \) shunt resistors and 1 \( \Omega \) series resistors to each node.

STEP 3 Change the MOS model level. Level two is known to have a discontinuity in its I-V characteristics which may cause transient analysis non-convergence.

STEP 4 Change the integration method from the default trapezoidal to gear by including the \texttt{METHOD=} parameter on the \texttt{OPTION} card. The use will have to pay the price of increased CPU time for this change.

STEP 5 Re-labeling of the source and drain may aid in convergence. The drain current is used in the check for convergence whereas the source current is not.
STEP 6  Reset the transient analysis iteration limit by using the 
**ITL5** parameter on the **OPTION** card. (default 5000)

**NOTE:** The remaining steps must be used with care. These steps modify the
parameters used by the program to determine convergence.

STEP 7\(^2\)  If **LULTIM=2** (default) SPICE uses truncation-error timestep
control and the user must vary **TRTOL** on the **OPTION**
card to change the conditions under which convergence
occurs. The default for TRTOL is 7 increasing this number
will aid convergence.

STEP 8\(^2\)  If **LULTIM=1** SPICE uses truncation-error timestep control
and the user must vary **ITL4** on the **OPTION** card to
change the conditions under which convergence occurs. The
default for ITL4 is 10 increasing this number will aid
convergence with increased CPU time required.

---

2. When using these steps the user should pay close attention to the simulation results for
validity.
Vita Auctoris

Bruce Erickson was born on April 30, 1962 in Winnipeg, Manitoba. He completed his high school education at Aldershot High School in Burlington, Ontario. He graduated from the University of Windsor in 1987 with a Bachelor of Applied Science in Electrical Engineering. In April 1989 he finished his Masters of Applied Science in Electrical Engineering also at the University of Windsor. His research interests include circuit simulation and embedded control applications.