UNIVERSITY OF ADELAIDE LIBRARY
12.10.88

Design of Arithmetic Systems in VLSI

Grzegorz B. Zyner, B. E. (Hons.)

Being a thesis submitted

for the

Degree of Doctor of Philosophy

in the

Department of Electrical and Electronic Engineering

The University of Adelaide

Adelaide, South Australia.

October 1988.

awarded 24.8.89

Contents

1 Introduction. 1

2 A Methodology for Custom VLSI Design and the Associated Software Tools 5

2.1 Introduction 5

2.2 Significant Issues in VLSI Design 6

2.2.1 Unstructured Medium 6

2.2.2 Functional Concurrency 6

2.2.3 Communication on the Silicon Surface 7

2.2.4 Topological Constraints 7

2.2.5 Technological Changes 7

2.2.6 Design Trade-offs 8

2.2.7 Circuit Testability 8

2.2.8 Fabrication Yield 8

2.3 Major Design Approaches 8

2.3.1 Highly Partitioned Systems 9

2.3.2 Functionally Partitioned Systems 10

2.4 Structured Design Methodology 11

2.4.1 Hierarchy 11

2.4.2 Modularity 12

2.4.3 Regularity 12

2.4.4 Description Domains 13

2.4.5 Restricted Constructs 13

2.4.6 Design Procedures 14

2.5 The Design Process 15

2.5.1 Design at the TOP Level 17
2.5.2 Design at the CP Level 19
2.5.3 Design at the MCM Level 24
2.5.4 Design at the FM Level 25
2.5.5 Composition 27
2.6 The Software Tools 27
2.6.1 Floorplanning 27
2.6.2 The Functional Simulator USA 28
2.6.3 Symbolic Design Method 32
2.6.4 Implementation Tools 33
2.7 Summary 38

3 A Comparative Study of Analytical MOS Transistor Models 40
3.1 Introduction 40
3.2 Large Geometry MOSFET Models 41
3.2.1 Pao – Sah Model 41
3.2.2 A Charge Sheet Model 45
3.2.3 Dang's Model 46
3.2.4 Inhantola–Moll Model 47
3.3 Small Geometry MOSFET Models 49
3.3.1 Short Channel Effect 50
3.3.2 Narrow Width Effect 78
3.3.3 Small Geometry Effect 81
3.3.4 High Field Effects 84
3.3.5 Junction Resistances 91
3.3.6 Linear Region of Operation 92
3.3.7 Saturation Voltage 99
3.3.8 Finite Output Conductance in Saturation 102
3.4 Summary 107

4 MOSFET Analytical Model for Circuit Simulation 109
4.1 Introduction 109

4.2 Proposed Model Equations 112
4.2.1 Threshold Voltage 112
4.2.2 Carrier Mobility 116
4.2.3 Linear Region of Operation 117
4.2.4 Saturation Voltage 119
4.2.5 Saturation Region of Operation 120
4.3 Parameter Extraction 121
4.4 Comparison with Experiment 122
4.5 Modification of FACTS 128
4.6 The Target Fabrication Process 132
4.7 Summary 136

5 Numerical Modelling of MOS Transistors **139**
5.1 Introduction 139
5.2 The Mathematical Model 140
5.2.1 Basic Semiconductor Equations 140
5.2.2 Equation Domain and Boundary Conditions 142
5.2.3 Scaling 144
5.3 Solution of Semiconductor Equations 145
5.3.1 Discretization in the Space Domain 145
5.3.2 Discretization in the Time Domain 150
5.3.3 Solution Methods of the Nonlinear System 153
5.4 Solution of Systems of Linear Algebraic Equations 158
5.4.1 Basic Iterative Methods 159
5.4.2 Stopping Criterion 167
5.4.3 Convergence Acceleration of Iterative Methods 168
5.4.4 Systems of Linear Equations with a Nonsymmetric Matrix A . . 171
5.5 Summary 172

6 The TIME Machine Architecture **174**
6.1 Introduction 174
6.2 The Two-Dimensional Array Architecture 175

6.2.1 Two-Dimensional Problem 177
6.2.2 Evaluation of a Vector Inner Product 181
6.3 The TIME Machine Architecture 184
6.3.1 Three-Dimensional Problem 187
6.4 The Mathematical Processor (mP) 198
6.4.1 Design Overview 198
6.4.2 Logical Architecture 200
6.4.3 Transformation to Physical Architecture 205
6.4.4 Floating Point Arithmetic 209
6.5 Summary 211

7 The Floating Point Adder **213**

7.1 Introduction 213
7.2 Free Form Specification 214
7.3 Floating Point Addition Algorithm 215
7.3.1 The Conventional Form 215
7.3.2 An Analysis of Normalized Floating Point Addition 217
7.3.3 The Modified Form 220
7.4 Functional Specification 222
7.5 System Partitioning 222
7.6 Partitioning Validation 227
7.7 Physical Architecture 228
7.7.1 Sign Unit (SignUnit) 233
7.7.2 Exponent Input Stage (ExpInp) 234
7.7.3 Mantissa Switch (ManSw) 239
7.7.4 Alignment Shifter (Shifter) 241
7.7.5 Mantissa Complementer (ManComp) 244
7.7.6 Mantissa Adder (ManAdder) 244
7.7.7 Most Significant Bit Position Finder (MSBPF) 251
7.7.8 Carry Propagate Signal Generator (Propagater) 254
7.7.9 Shift Distance Encoder (PLA2) 257

7.7.10 Mantissa Normalizer (Normalizer) 257
7.7.11 Special Case Detector (ManFlags) 258
7.7.12 Mantissa Rounder-Complementer (ManInc) 265
7.7.13 Exponent Output Stage (ExpOtp) 267
7.8 Design Evaluation 270
7.8.1 Area Requirements 270
7.8.2 Speed Performance 272
7.9 Summary 285

8 The Floating Point Multiplier **287**
8.1 Introduction 287
8.2 A Comparative Study of Parallel Integer Multiplier Designs 288
8.2.1 Array of Ripple Carry Adders (ARCA) 290
8.2.2 A Binary Tree of Ripple Carry Adders (BRCA) 292
8.2.3 Array of Carry Save Adders (ACSA) 297
8.2.4 Wallace Tree Multiplier (WT) 301
8.2.5 The Modified Booth Algorithm Multiplier (MBA) 302
8.2.6 Multipliers for Signed Binary Numbers 306
8.3 New Parallel Integer Multiplier Designs 306
8.3.1 Binary Tree of Carry Save Adders (BCSA) 307
8.3.2 "Double Stream" Multiplier (DS) 314
8.3.3 Discussion 316
8.4 Free Form Specification 318
8.5 Floating Point Multiplication Algorithm 318
8.5.1 The Conventional Form 319
8.5.2 An Analysis of Normalized Floating Point Multiplication 320
8.5.3 The Modified Form 320
8.6 System Partitioning 321
8.7 Physical Architecture 322
8.7.1 Mantissa 323
8.7.2 Mantissa Multiplier 324

8.7.3 Sticky Bit . . . 333
8.7.4 Round Condition Detector (RndCndDet) . . . 333
8.7.5 Mantissa Rounder (ManRndA and ManRndB) . . . 334
8.7.6 Mantissa Overflow Detector (ManOvf) . . . 335
8.7.7 Mantissa Selector (ManSel) . . . 336
8.7.8 Exponent . . . 337
8.7.9 Input Stage . . . 339
8.7.10 Exponent Adder . . . 340
8.7.11 Bias Subtracter . . . 340
8.7.12 Exponent Incrementer . . . 341
8.7.13 Flags Generator . . . 341
8.7.14 Output Stage . . . 343
8.7.15 Sign . . . 343
8.8 Final Floorplan . . . 343
8.9 Design Evaluation . . . 344
8.9.1 Area Requirements . . . 345
8.9.2 Speed Performance . . . 348
8.10 Summary . . . 351

9 Conclusion **353**

A The Transform and Filter Brick **357**

B Model Parameters Extraction **358**
B.1 Determination of V_{th}, N_A and V_{FB} . . . 358
B.2 Determination of y_{S0}, σ, α and δ . . . 359
B.3 Determination of μ^o, θ, L and W . . . 360
B.4 Determination of η . . . 361

C ISSL Code Examples **363**
C.1 Functional Module Specification . . . 363
C.2 Composition Module Description . . . 366

Abstract

Accurate transistor simulation, a task that is essential for time efficient and cost effective design and evaluation of smaller and novel transistors, can only be achieved with a numerical solution of the basic semiconductor equations. The solution process of such equations is characterized by the need to perform a large number of floating point arithmetic operations. The computations involved are also highly regular and may be performed, with a suitable design of the algorithm, with a high degree of parallelism.

The application of VLSI (very large scale integration) technology offers a possibility of greatly increasing the speed with which the above numerical problem and, in general, engineering and scientific problems having substantial computational requirements can be solved; firstly by allowing efficient implementation of the time consuming floating point arithmetic operations in hardware, and secondly by providing a cost effective means of performing the computations with a large degree of concurrency.

The thesis is largely concerned with the design of a parallel computer for an efficient solution of partial differential equations arising from the numerical simulation of MOS transistors. A complete design of such a computer and even of its major part, such as the individual processing element, are tasks well beyond the scope of this work. Instead the following approach which results in novel designs of two arithmetic circuits is taken. The specific problem is analysed, its relevant features are identified and, at the system level, the proposed architecture of the parallel computer, the TIME Machine, is described. The proposed computer is composed of a modified two-dimensional array of single chip processing elements, the Mathematical Processors. The original modification proposed in this thesis, consisting of introducing a number of additional processing elements, results in a parallel computer that combines the characteristics of a two-dimensional array and of a binary tree of processing elements. At the chip level the main functional blocks are identified and the hardware architecture of the Mathematical Processor is specified in sufficient detail for the design and implementation work of the individual modules to be carried out.

The complete digital circuit designs incorporating novel architectural features of two most complex functional blocks of the Mathematical Processor, the Floating Point Adder and the Floating Point Multiplier, are described in detail in this thesis. The test results, obtained from fabricating a substantial section of the Floating Point Adder using a 2-micrometer p-well CMOS process, together with the improved small geometry MOS transistor analytical models, which have been developed in the course of this work and incorporated into the circuit analysis program FACTS, are used to predict the performance of these two arithmetic systems, assuming a hypothetical 1-micrometer p-well

CMOS process, and to demonstrate the speed advantages which have been achieved compared to the corresponding conventional hardware designs of such systems.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any University, and to the best of the author's knowledge and belief contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

The author consents to the thesis being made available for photocopying and loan, if accepted for the award of the degree.

Grzegorz B. Zyner.

Acknowledgements

I would like to thank my supervisor Dr. Peter Cole for introducing me to the field of VLSI design, and for his help and advice during the course of this research.

I am indebted to Dr. Kamran Eshraghian for instigating this project and for making the VIVID software system available in the University of Adelaide. Without this software this project would have not been possible.

My thanks also go to the staff of Integrated Silicon Design Pty. Ltd., and especially to Mr. Alf Grasso, for providing the knowledge and experience needed in maintaining and "debugging" the VIVID software system.

I would like to thank Mr. D. Fensom, Dr. A. Papliński and Mr. J. Ziukelis for discussions on the subject of design of arithmetic systems which have been very helpful, especially in the early stages of this project.

It would have been difficult for me to carry out the experimental part of this work without the help of the technical staff of the Electrical Engineering Departments in the University of Adelaide and the South Australian Institute of Technology. My special thanks go to Mr. J. Bannigan for helping me with the task of collecting experimental data from test transistors.

I am grateful to Alex Dickinson and Michael Pope for their efforts in educating me in the matters of software engineering and especially for their help in modifying the circuit simulation program FACTS, and to John Rockliff for his checking and criticism of substantial parts of this thesis. I would also like to thank all my other colleges in the Department for the experiences and insights gained while working with them.

Finally, I would like to thank the persons to whom I dedicate this thesis – my parents for their continued support and encouragement, and my wife, Leanne, for her tremendous contribution, although of a nontechnical nature, to this thesis.

G. B. Z.

Publications

1. K. E. Eshraghian, R. C. Bryant, A. G. Dickinson, D. S. Fensom, P. D. Franzon, M. T. Pope, J. E. Rockliff, and G. B. Zyner, "A new CMOS architecture for signal processing," *Pacific/Asian Regional Conference on VLSI*, Melbourne, Australia, May 1984.

2. K. E. Eshraghian, R. C. Bryant, A. G. Dickinson, D. S. Fensom, P. D. Franzon, M. T. Pope, J. E. Rockliff, and G. B. Zyner, "The Transform and Filter Brick - a new architecture for signal processing," *Proceedings of VLSI 85*, Tokyo, August 1985.

3. G. Zyner and K. E. Eshraghian, "Multiplier/divider designs for signal processing systems," *Conference Proceedings, IREECON International Convention*, Melbourne, Australia, October 1985.

4. G. Zyner and P. H. Cole, "SPEEDEY - A special purpose VLSI architecture for the solution of partial differential quations," *Conference Proceedings, IREECON International Convention*, Sydney Australia, September 1987.

5. G. Zyner and P. H. Cole, "A small geometry MOSFET model for circuit simulation," *Conference Proceedings, IREECON International Convention*, Sydney Australia, September 1987.

UNIVERSITY OF ADELAIDE LIBRARY

Chapter 1

Introduction.

Large scale scientific and engineering computing is dominated by "crunching of numbers". Fast Fourier transform and convolution [RaGo75] are two examples of data processing involved in image and speech processing problems. Aerodynamic problems, such as modelling the flow of air around an aerofoil, or wind simulation for weather prediction [FoOt84], [ChGu86] are further examples of computationally intensive tasks requiring millions of floating point multiplications and additions. A wide class of such problems involves a numerical solution of one, or a simultaneous solution of a number of coupled partial differential equations. In particular, rigorous simulation of semiconductor devices [Selb84], [Engl86], [COCM86] requires a numerical solution of three coupled partial differential equations (commonly known as the basic semiconductor equations), namely the Poisson's equation and the current density continuity equations for holes and electrons.

In recent years the MOSFET (metal-oxide-silicon field effect transistor) has become one of the most important components in VLSI (very large scale integration) circuits. With the advent of MOS transistors with submicrometer channel lengths and widths numerical device simulations, especially those which consider *all* space dimensions, have become essential [HuCh82], [KYYS82], [YKTH82], [TMAS85], [ShDa85]. Computer simulations have become invaluable for time efficient and cost effective design and evaluation of new MOS transistors. However, device simulation programs, such as DEVICE [FNPP85] and CADDETH [TMAS85], which solve the basic semiconductor equations in three dimensions require the computational power of large supercomputers, precluding them from being widely available within the VLSI community.

The computations involved in the solution process of the basic semiconductor equations are highly regular and may be performed, with a suitable design of the algorithm,

with a high degree of parallelism, making them well suited for VLSI implementation. Current progress in MOS transistor miniaturization permits circuits with over 10^5 transistors (highly regular structures, such as memories, contain over 10^6 transistors) to be fabricated on a single chip. This high device density achievable with VLSI technology creates a potential for the implementation of a microelectronic system with a rich instruction set, containing elements with data processing capability, memory and input-output interface on a single die. It is possible for such circuits to exhibit a degree of concurrency which was previously not economically possible. In particular, VLSI can be considered to be a new medium for the realization of high performance systems required to perform complex or application specific computations and algorithms. The application of VLSI technology offers the possibility of greatly increasing the speed with which problems having substantial computational requirements can be solved: firstly by allowing efficient implementation of the time consuming floating point arithmetic operations in hardware, and secondly by providing a cost effective means of performing the computations with a large degree of concurrency. The design of such systems is one of the major concerns of this thesis.

Design of integrated circuits may be regarded as a process of transforming the desired functional behaviour into a circuit to be fabricated on a silicon chip. There are a number of important limitations imposed on the design of VLSI systems. In particular, the physical limits which determine the switching speed of individual transistors dictate that the communication be localized, if large time delays or power dissipation costs are to be avoided. In addition, complexity limitations dictate that the design be highly regular and composed of as many identical elements as possible, both to reduce the design and verification time, and to provide scope for design automation. In general, a successful design of an integrated circuit at the VLSI level of complexity, relies on the application of a suitable design methodology and on the use of computer software for all aspects of the circuit design process. Chapter 2 of this thesis addresses the main design concerns and discusses the design methodology applied in this work to successfully produce circuits of substantial complexity. In addition, a description of the design process followed in this work and the software tools that were available are also described in that chapter.

A software tool of particular importance in the design of integrated circuits is the transistor level circuit analysis program. It provides a time efficient and cost effective method of estimating the performance and verifying the correctness of a given circuit prior to that circuit being fabricated. For a circuit analysis program to simulate accurately circuit performance, an accurate transistor model is necessary. The main requirement of the transistor model in a circuit analysis program is to provide a means of efficiently

and accurately computing the drain current and capacitance, in the various nodes of a transistor, for a given bias voltage. Another important consideration is the response speed and the capability of simulating circuits with ever increasing number of transistors. These two main objectives, ie accuracy and speed, cannot be met simultaneously; higher simulator speed has to be traded-off for lower accuracy.

The circuit analysis program FACTS, which was used in this work, is sufficiently fast to permit simulation of circuits of substantial complexity ($\approx$ 10000 transistors) to be carried out in an economical manner. However, circuit simulation results were initially of limited value because of the poor accuracy of the MOS transistor analytical models, that were incorporated into the simulator. While it was recognized that the available software tools were not entirely suitable for the most time and cost efficient design of microelectronic systems at the chip level, and that the effort required for their development and improvement was outside the scope of this work, two significant advantages in directing some effort into improving the accuracy of FACTS were identified. Firstly, an incorporation of accurate MOS transistor current-voltage models would result in a software tool that could reliably predict the performance of digital circuits having substantial complexity and secondly, much knowledge and understanding would be gained from this process. The study of such models is the subject of Chapter 3. The transistor analytical model with geometry dependent parameters which was developed as a result of this study, and the method used to incorporate that model into the circuit simulator are described in Chapter 4. The accuracy of the model is demonstrated by comparing it with experimental results also in Chapter 4.

The thesis is largely concerned with the design of a parallel computer for an efficient solution of partial differential equations arising from the numerical simulation of MOS transistors. A complete design of such a computer and even of its major part, such as the individual processing element, constitute tasks well beyond the scope of this work. Instead the following approach which results in novel designs of two arithmetic circuits is taken. The specific problem is analysed and its relevant features are identified in Chapter 5, and at the system level a VLSI architecture of the parallel computer, described in Chapter 6, is proposed. The proposed computer, the TIME Machine, is composed of a two-dimensional array of single chip processing elements, the Mathematical Processors. At the chip level, the main modules are identified and the logical architecture is specified in sufficient detail for the design and implementation work of the modules to be carried out.

The culmination of this thesis is the complete digital circuit design, incorporating novel architectural features, of two most complex functional blocks of the Mathematical Pro-

cessor; the Floating Point Adder which is described in Chapter 7 and the Floating Point Multiplier which is described in Chapter 8. The test results, presented in Chapter 4, obtained from fabricating a substantial section of the Floating Point Adder using a 2-micrometer p-well CMOS process, together with the improved small geometry analytical models developed in the course of this work and incorporated into the circuit analysis program FACTS, are used both to estimate the performance of these two arithmetic systems assuming a hypothetical 1-micrometer p-well CMOS process, and to demonstrate the speed advantages which have been achieved compared to the corresponding conventional hardware designs.

Chapter 9 summarizes the conclusions obtained in the many subjects addressed in this thesis and presents some directions for further research.

The group project in which the author of this thesis was involved in the early stages of this research program, in order to gain knowledge and experience in the area of VLSI system and circuit design, is briefly described in Appendix A of this thesis.

Chapter 2

A Methodology for Custom VLSI Design and the Associated Software Tools

2.1 Introduction

The largely unstructured property of the silicon surface gives the designer great freedom and flexibility in the placement of individual transistors and other circuit primitives to realize the desired function and to develop novel structures. This property of the silicon surface, however, is also one of the main causes of complexity in VLSI system design [Séqu83], [Snyd84]. Other factors that contribute to VLSI design complexity include: the large number of transistors to be managed, and the communication difficulties on the silicon surface caused by a relatively small number of conductive layers (with some layers exhibiting limiting characteristics of high resistance or capacitance) which can be used for providing interconnections between the individual transistors. This complexity must be controlled, if the system which is being designed is to make a full use of the available fabrication process.

This chapter begins by describing the possibilities presented to the IC designer by present day VLSI technologies and the significant constraints that contribute to the complexity of VLSI system design. The techniques and software tools, used in this work to manage that design complexity and successfully produce circuits of substantial size and complexity, are subsequently discussed.

2.2 Significant Issues in VLSI Design

The major design issues that need to be addressed in the design process and that contribute to the high level of complexity associated with the task of designing a large microelectronic system, include the following: unstructured nature of the medium, problems deriving from functional concurrency, communication characteristics of the silicon surface, constraints deriving from the two-dimensional nature of the surface, rate of technological evolution, normal design trade-offs, circuit testability and fabrication yield. Each of these matters is discussed in some details in the subsections that follow.

2.2.1 Unstructured Medium

The physical domain of a single silicon chip is unstructured in the sense that it does not impose any particular partitioning or subdivision of the circuit which is being implemented. This freedom on the one hand gives design flexibility which can be potentially exploited for significant performance gains by designing novel circuit configurations, but on the other hand it may result in a situation in which the complexity within this unstructured medium "simply overwhelms the designer" [Séqu83].

2.2.2 Functional Concurrency

As pointed out in Chapter 1, VLSI is a medium in which it is possible to achieve high performance implementations of algorithms by designing systems having high degree of functional concurrency. In such systems many useful operations are performed by various parts of the silicon chip at the same time. However, this large scope for concurrent operation in VLSI adds greatly to the design complexity mainly because concurrent execution gives rise to a number of possible errors not present in sequential systems. Some of these errors as identified in [Séqu83] include: unequal rates of data production-usage (buffering), reading data before it has settled (timing) and cyclic data dependencies (deadlock). In addition, it is more difficult to understand fully the behaviour of circuits designed for high level of concurrency, and this may present problems of defining suitable strategies for design validation and circuit testability [WiPa83].

2.2.3 Communication on the Silicon Surface

Circuit interconnection is the primary complexity problem in VLSI design [TRLG81], [DaGw82], [Séqu83], [Ferr85]. Interconnect length and complexity provide fundamental limits in circuit integration [Keye81]. There are four major costs associated with integrated circuit interconnection:

1. *Area* – longer wires occupy larger silicon area leaving less space for the placement of active primitives;
2. *Speed* – longer wires result in greater RC time delays for a fixed driver size; and
3. *Power* – larger drivers required to drive the capacitive loads associated with long wires dissipate more power.

2.2.4 Topological Constraints

The active circuit elements are constrained to share a single two-dimensional plane of the chip surface with the wires used for carrying signals to various parts of the chip. This two-dimensional nature of integrated circuits imposes a set of topological restrictions on implementation, ie it is not possible to jump to a subroutine as it is in software [Séqu83]. This set of restrictions is very often the cause of the difficulty in finding a suitable silicon mapping for the intended system function.

An additional serious drawback of VLSI technology is the limited number of pins available on each chip [RaMa81], [Snyd84]. As a result, communication between separate VLSI chips also becomes a difficult design problem, and it becomes necessary to design VLSI chips in such a way that the benefits of density improvements can be utilized without a requirement for additional pins.

2.2.5 Technological Changes

The length of time needed for the development of a new product often exceeds the time taken to bring about improvements in a fabrication process. As a consequence, new designs are often aimed at a fabrication process which is still in its experimental phase with the hope that the fabrication process will reach a sufficient level of maturity when the design is complete [Keye81], [Séqu83]. This fact necessitates the use of such circuit design domains and computer aided design (CAD) tools that permit the chip mask

layout to be economically updated to new design rules of a more advanced fabrication process.

2.2.6 Design Trade-offs

The main objectives in producing a physical implementation of a microelectronic system are to minimize the silicon area and maximize the circuit speed. These objectives must be traded-off against design time and cost. There are many other design trade-offs specific to the VLSI medium which need to be addressed. Two additional examples of particular trade-offs, that in some cases need to be considered, are the problem of how to divide the realization of a given function between hardware and software implementation and the question of how much extra hardware to include to simplify the testability of the system being designed. One of the main design tasks is to evaluate and decide upon all the design trade-offs relevant to the particular design problem [RaMa81], [Snyd84].

2.2.7 Circuit Testability

In a fabricated chip only signals that are brought out to the chip's terminals are readily observable, and in general it is very tedious and costly to gain access to the inner parts of a circuit for testing and debugging purposes. This fact makes it necessary to treat the task of devising methods of testing the circuit as an integral part of the design process. In general special circuits, such as a scan path [WiPa83], [WeEs85] or provision of internal access probe pads to incorporate a sufficient level of observability and controllability to make the circuit testable, must be included on the chip.

2.2.8 Fabrication Yield

One of the major factors determining the final fabrication yield is the area of the chip, with the yield decreasing dramatically as the chip area is increased [MeCo80], [WeEs85]. Thus, the fabrication yield puts an upper limit to the maximum silicon area that is available to realize a particular function.

2.3 Major Design Approaches

The main objective of every designer is to design in as short a design time as practicable, reliably working, space efficient and cost effective designs. In order to meet this

objective, it is necessary to follow a design approach that recognizes the issues of VLSI design, discussed in the previous section, and provides means for managing the design complexity.

The general approach used to reduce design complexity is to impose different structuring schemes on the silicon surface. The two major classes of structuring VLSI designs, highly partitioned and functionally partitioned [Ferr85], are discussed in the next two subsections. Each design approach can be considered as a trade-off between the need to reduce both the length of interconnect and the silicon area, and the need to simplify and automate the design process, thus increasing design productivity.

2.3.1 Highly Partitioned Systems

It is possible to impose a structure onto the unstructured silicon surface by the use of restricted configurations, such as gate arrays or standard cells [Brod81], [Hurs85]. Such a design style is characterized by the placement of standard cells (small functional modules) and separate logic gates, and routing them to realize the intended function, such that the area and length of the interconnect are minimized.

In a gate array design the positions of the gates are already fixed on the surface and channels are provided for their interconnection. The design process is reduced to the task of determining a suitable gate interconnection; one that will ensure that the necessary wires fit into the provided channels and the critical timing circuit paths are optimized.

In the standard cell design style the designer is restricted, in the implementation of the required functional behaviour, to the use of parameterized fundamental functional blocks called standard cells. Each standard cell implements a function usually of the complexity of a few logic gates. The layout, the port locations and the heights of the bounding boxes of the cells are fixed, but their positions in the overall design are not. The cells are typically placed in a number of rows with sufficient space between the rows being reserved for the interconnecting wires.

The gate array design style and the standard cell design style are termed *highly partitioned* because the designs are structured into a large number of small partitions. The main advantages of this design approach are the short time of implementation and high design automation which can be achieved. The main disadvantage of this design style is an inefficient use of silicon area which is caused primarily by the large area occupied by the wires running between the active components. This prevents the circuits designed using this approach from achieving transistor densities which are possible with the level

of integration offered by VLSI technology.

Continual progress in VLSI technology reduces the cost associated with communication on the silicon surface. For example, more interconnect layers become available [Orbi87] which can be placed on top of the active devices and new conductive substances are being developed with more attractive electrical properties [CrZi79]. The most powerful technique, however, to improve the utilization of silicon surface is the application of a design style which considers the costs of interconnections at a very early stage of the design process, and attempts to minimize the silicon area allocated to function interconnect.

2.3.2 Functionally Partitioned Systems

In order to utilize the available silicon area more efficiently it is necessary to resort to design styles based around *functional partitioning*. Functional partitioning consists of a recursive decomposition of the system into smaller and simpler parts (modules), such that the combined operation of the properly interconnected parts performs the intended function of the system. This partitioning process consciously focuses on the functional aspects of the design, but the functional and geometrical hierarchies must eventually match because the interface between modules is along the geometrical interfaces.

In general, functionally partitioned circuits are implemented using *full custom* design style. In this way, not only is the system functionally partitioned but the design of the layout takes into account the geometrical interactions between the interconnecting functional blocks. The increased function densities achievable with this form of partitioning may be attributed to an associated reduction in interconnect area [Ferr85].

The use of full custom design based on functional partitioning results in much higher design complexity compared with the highly partitioned design techniques. This stems from the fact that module interfaces must be carefully designed to decrease interconnect cost and the higher transistor densities which are possible increase the number of components in the design.

Crucial to a successful and efficient construction of a large and complex system is a suitable design methodology, and a good set of software tools that support it. In addition, since the complexities of present day microelectronic systems are much too high to be effectively managed by a single designer, the design methodology, applied to VLSI system design, must not only offer means of reducing the complexity but must also be suitable for an efficient team operation.

All designs presented in this thesis were produced using the full custom design style based on functional partitioning. The design methodology which was applied in this work and which incorporates a number of design complexity management techniques, has been adapted from the structured design methodology presented in a number of sources [MeCo80], [LBBR81], [TRLG81], [DaGw82], [Lipp83] and [BrKu83], and is described in the following section.

2.4 Structured Design Methodology

The *structured design methodology* introduced into VLSI design by Mead and Conway [MeCo80] is one of the most powerful techniques available to designers to control the complexity of VLSI design. The most important parts of this methodology are the use of hierarchy, in describing system structure, and the recognition of regularity, if present, in system structure. These and other major aspects of this design methodology for custom design of VLSI circuits are described in some detail in the remainder of this section.

2.4.1 Hierarchy

If the system to be designed is too large to be comprehended in its entirety it must be partitioned (modularized) into subsystems; smaller, more manageable parts. Each subsystem, properly defined in terms of its functional behaviour and interface, can be viewed as a separate, simpler system and it, in turn, may be capable of further decomposition. Such a recursive application of partitioning results in a system hierarchy, the lowest level of which is composed of well understood circuit elements (functional modules) having complexities at a level consistent with available design capability and designer understanding, such as registers, adders and multiplexers. Combinations of functional modules in accordance with the system hierarchy are called composition modules. Combinations of composition modules give rise to composition modules at higher levels of hierarchy.

The use of hierarchy also contributes to the reduction in design complexity by only emphasizing the details of lower level modules which are necessary to the composition task and suppressing other details. For example, at one level of system hierarchy the representation of a module may include information about port positions, names, layers, the module size and the aspect ratio of the bounding box, while at the lowest level of system hierarchy such a module is represented by its full mask representation. Such an

information hiding technique enables modules to be used as components in a composition prior to their implementation.

The fact that hierarchical modules contain submodules (ie simpler modules), whose behaviour and external interface are well specified and understood, not only makes the design simpler but it also simplifies their testing, ie the submodules may be tested separately followed by their interconnection.

2.4.2 Modularity

The first level of system decomposition results in large, functionally independent modules (subsystems). Once each subsystem has been specified in sufficient detail, the design can proceed in parallel with different designers being responsible for the implementation of different subsystems. Subdividing the system into modules leads to the formation of a module library into which finished, verified and properly documented modules are placed, to be used by other design teams in other parts of the system, or in later projects, resulting in a reduction of design time. Modularity also achieves computational gains in the area of verification; only the modules that have been changed need to be tested by, for example, performing circuit simulations or design rule checking.

2.4.3 Regularity

Introduction of regularity into the system at any level of decomposition results in a reduction of complexity in system description. By a suitable application of partitioning, a composition module may sometimes be decomposed into functional modules having identical functional specifications. This gives rise to functional regularity. Design and computation time gains are achieved since only a single functional module needs to be designed and verified, and then simply replicated (instanced) the required number of times.

Regularity of interconnection is achieved if the interface of each module is also identical. Figure 2.1 depicts an example of functional regularity, and how it may or may not be accompanied by regularity of interconnection.

Both functional and interconnection regularities are often a property of modules at the low levels of system hierarchies and occur within such elements as n-bit parallel adders, registers or multiplexers. The description of the system at high levels of hierarchy, on the other hand, is in general in terms of nonidentical modules having different functional behaviours. At these levels, however, regularity of interconnection may still be

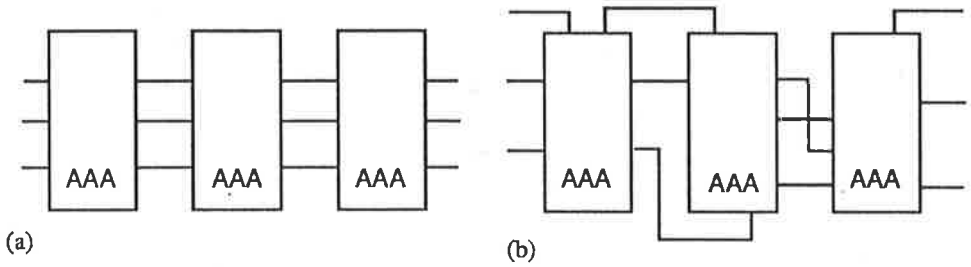


Figure 2.1: Examples of functional regularity: (a) including regularity of interconnection, (b) not including regularity of interconnection.

introduced to reduce communication complexity.

2.4.4 Description Domains

An additional technique for a reduction of system complexity, by the suppression of irrelevant detail, is the use of various domains to simplify the description of the system. The process of chip design proceeds through a series of levels of abstractions starting from the most abstract description of the system, addressing the system behaviour, and ending at the most detailed description, ie the mask layout. There are three domains of system description, as identified in the Caltech [TRLG81] design methodology:

1. *Behavioural* – the description of the intended system function without necessarily specifying any structure;

2. *Structural* – description of the system in terms of its components, with each component realizing a subset of the overall behaviour and components being interconnected in such a way as to realize the function of the system; and

3. *Physical* – description of the system in terms of a set of interconnected physical components, eg transistors, wires, capacitors, each having a direct mask representation.

2.4.5 Restricted Constructs

A number of restrictions can be introduced into the design process, in order both to reduce the possibility of designer's error and to simplify the implementation of the software tools. For example, in the standard cell design style modules that have been proven to work correctly are used only, and the processes of placement and interconnection are

generally automated, to reduce the level of human intervention which usually results in a significant number of errors.

In full custom design style, the structured methodology may impose some or all of the following restrictions.

1. When producing the geometrical layout, the designers are only permitted the use of Manhattan geometry in which all mask edges meet at right angles. This makes the implementation of software tools, such as a design rule checker, substantially simpler and faster and reduces the margin for error by limiting, for example, the number of transistor configurations possible. An example of a functional module implemented in this way is shown in Figure 8.24.

2. The use of hierarchy for system decomposition is limited to *restricted hierarchy.* This is a concept in the Caltech design methodology, where only two kinds of modules are allowed: composition modules and leaf modules (also referred to as functional modules). A composition module contains only instantiations of lower level modules and no circuit primitives, ie wires, contacts, transistors, are allowed. A functional module is defined only in terms of circuit primitives and it does not contain instances of any other modules.

3. The leaf modules may only have rectangular boundaries which must not overlap (apart from sharing a common bus on the adjacent boundary of a neighbouring leaf module) during composition. In addition, module interconnection may only occur at adjacent edges via predefined ports – this simplifies module composition and validation.

2.4.6 Design Procedures

There are two main design procedures possible: the "top-down" and the "bottom-up". The process of system decomposition is carried out predominantly in the top-down manner. The primary advantages of the top-down design is that the problem of communication on the silicon surface may be addressed at an early stage. Additional advantages are identified in [Lipp83]:

1. It generates stable interfaces between functional parts of the system;

2. A substantial part of the design is process independent;

3. Design may be initiated before the target fabrication process is stabilized; and

4. It allows the design of various parts of the system to be carried out at different times or by different design groups.

In practice a pure top-down design style is unrealistic and an element of the bottom-up design must be introduced. This introduces the necessary information about the characteristics of the medium, which may not be visible at the top, and allows for the top-down design to be guided properly. Examples of how the top-down design was combined with the bottom-up design in this work are given in the next section.

The design process that was followed in this work is described in the following section. It incorporates the features of the structured design methodology, discussed in this section, and was influenced by the particular set of available software tools, which tools are described in Section 2.6.

2.5 The Design Process

The design process of a digital system can be subdivided into two phases: a conceptual or planning phase and an implementation or construction phase [BrTF83], [DaSh83]. The planning phase involves the generation of functional and structural specifications of a particular design solution which will be later implemented, in the construction phase, in the physical domain.

During the planning phase the principles of hierarchical decomposition are applied and the complete design is subdivided and specified as a set of interconnected functional blocks, each of which presents a simpler design task. Function is typically used as the main criterion to produce system partitioning with the other main design issues, such as silicon or printed circuit board area, cost and performance treated as design constraints, rather than making them principal goals. In this way a sequence of design tasks of sufficient simplicity is formulated to be carried out manually by groups of designers (with different groups concentrating on different design tasks) or automatically by software tools. The result of decomposing a module into a number of lower level modules is validated to ensure that the particular partitioning step did not violate any specifications of the higher level module. Typically a number of competing design solutions are generated and one is accepted on the basis of informally specified cost criteria, such as ease of implementation, silicon area requirements, complexity, speed etc. The final product of the planning phase includes complete functional and interface descriptions for all the functional blocks identified in the decomposition process. Such information is normally presented by a combination of means including the use of a high

level language, a module floorplan or a schematic diagram.

The implementation phase deals with the problem of realizing the system as a semiconductor structure, as one or a number of different or identical chips, that has the composed function which is the same as the intended function of the digital system being designed. Typically, physical layout is generated for each leaf module using the leaf module schematic diagrams and floorplans. The leaf module layout is followed by the assembling of the leaf modules into the system layout, according to the floorplans of the intermediate levels of hierarchy. The result of the implementation phase is validated at each level of hierarchy, to ensure that the system functional specifications are met. A degree of optimization (adjustment of transistor sizes in some leaf modules) may also be required, to ensure that timing and power dissipation specifications are met.

The number of levels used to hierarchically decompose the complete system depends on a number of factors, such as the designer's experience, the software tools available and the complexity of the undertaken design. In a general case four major levels are identified in this work. At the highest level, referred to as the TOP level of hierarchy, the system may be viewed as a black box containing a set of labelled ports, which enable values to be communicated to and from the external environment. Associated with the black box is a behaviour which performs a mapping of given inputs to specific outputs, depending on the function being invoked. The second major level, referred to as the ChiP (CP) level, is identified as that level of the hierarchy at which the design is described in terms of individual integrated circuits (ICs). At the third major level, referred to as the Major Composition Module (MCM) level, a particular IC is described in terms of composition modules having substantial complexities, such as arithmetic logic units, random access memories, input-output ports, control sections, etc. The final major level of the hierarchy is the Functional Module (FM) level, at which all composition modules, identified at the MCM level, are described in terms of functional modules only.

In this work, the term *system* is defined as a digital circuit which is completely defined in terms of its functional behaviour and external interface. This term is used as a reference point to denote the part of the complete design on which the discussion is focused. Thus, the system may constitute a 1-bit adder module, a RAM module or a complete chip, depending on the particular part of the design being considered.

It follows from the above discussion that the number of major levels present in the hierarchical decomposition of a particular system will depend on the nature of the system, however, every system will have at least the TOP and the FM levels as part of its hierarchical decomposition (although these two levels may coincide in some trivial

cases which are not of interest in this discussion).

In the particular designs undertaken and reported in this thesis there are four systems that can be identified, namely the TIME Machine, the Mathematical Processor, both described in Chapter 6, the Floating Point Adder, described in Chapter 7, and the Floating Point Multiplier, described in Chapter 8. The highest level of the hierarchical decomposition of the digital system being designed in this work is the TIME Machine. The Mathematical Processor system belongs to the CP level of the decomposition of the TIME Machine system, and both the Floating Point Adder and the Floating Point Multiplier systems belong to the MCM levels of the hierarchical decompositions of both the TIME Machine and the Mathematical Processor system. A diagram showing the hierarchical decomposition of the Floating Point Adder system is given in Figure 7.9.

2.5.1 Design at the TOP Level

The initial circuit specification is usually in *free form* description and it focuses primarily on circuit's functionality. A free form description of the system to be designed is made by means of English language, diagrams, tables and mathematical expressions, perhaps supported by some functional model description. It usually consists of three parts:

1. A description of the intended functional behaviour and the interface of the system with the external environment;

2. A specification of the constraints to be satisfied by an acceptable implementation, eg response time, maximum silicon area allocated, reliability, power dissipation, system throughput; and

3. A specification of the relevant characteristics of the target environment of the system which may influence the specific implementation chosen, eg topological characteristics, ie stand alone operation or as a part of parallel processor, functional characteristics, ie the pattern and frequency of various functions which are invoked.

At the initial stages of the design process the free form specification of the system is analysed and if possible the desired function of the system is described using formal means most suitable to the particular problem, eg a high level language. The specification forms a controlling document for design review, design validation, chip test and chip qualification activities. For example, the free form specification of the TIME Machine system is analysed in Chapter 6 and a functional description for the TIME

Machine system is developed in terms of mathematical and data transfer operations. Another example is the Floating Point Adder system described in Chapter 7. The free form specification of the Floating Point Adder system is formalized by describing its functional behaviour in terms of shifts, additions and logical operations using a high level language ISSL [ISD88].

Very often the free form specification contains the necessary information regarding the *competence* aspect [Alle83] of the algorithm, ie the information specifying what the algorithm does, to be performed by the system being designed. The *performance* aspect [Alle83], ie how the algorithm is to be performed, of the algorithm is usually not given. In such cases, and in the event of the performance aspect of the algorithm being supplied and found not suitable for economical hardware realization, a new algorithm needs to be developed or an existing one modified. This process was carried out for both the Floating Point Adder and the Floating Point Multiplier systems where existing algorithms were modified to improve their performance aspects, by increasing the level of parallelism initially present.

Once a suitable algorithm is developed, or identified, the process of system decomposition can be finalized. In general the decomposition process is carried out in parallel, to some extent, with the process of algorithm analysis. For example, it has been identified in [Unge58], [BBKK68], [Scho75], [HKSH83], [FoOt84], [Snyd84], [Zakh84] and [HBFH85] that for the problem considered in this work, ie solution of partial differential equations, the most suitable architecture is that of a parallel computer composed of a two-dimensional array of identical Processing Elements (PEs). Each PE in such a computer is connected to its four nearest neighbours. A study of this architecture and the numerical algorithms used for solving such equations led to the identification of computations, required by the algorithms, which are not carried out efficiently on the two-dimensional array processor due to the limited communication links between the individual PEs. This led to the subsequent development, described in Chapter 6, of the TIME Machine which is a particular topology arrangement (TIME topology) of identical PEs, termed here the Mathematical Processors (mPs).

One of the first tasks, in defining the initial system partitioning, is to decide whether the technology available to the designer will enable the complete design to be implemented as a single IC or as a number of different ICs, and whether ready made components are in existence or a full custom chip design needs to be undertaken. (It is assumed that at this stage of the design process the broad decisions regarding the type of architecture being designed eg von Neuman, Harvard or a parallel processor have been made.) In reference to the designs reported in this thesis, after identifying the major architectural

features of the TIME Machine, it was then necessary to establish the main functional features of the individual mPs and to decide whether a number of such mPs could be fabricated on a single chip or a number of separate chips would be required to realize a single mP.

Having decomposed the system down to the CP level, and having validated (the subject of validation of design partitioning is discussed in Section 2.6.2.1) and evaluated the partitioning against the specific cost criteria, the designer can commence the design of each individual IC. The design process continues at the next major level of hierarchical decomposition, eg at the CP level, if at the TOP level it has been decided that a number of individual ICs need to be designed, or at the MCM level, if the TOP level of the system being considered corresponds to a single IC.

The decisions made at the TOP level of the TIME Machine system led to the functional and interface specifications of functional blocks, namely mPs, to be implemented as single ICs (with a possible additional data memory chip). The design of the Mathematical Processor was then carried out in sufficient level of detail to enable a complete design and implementation, in the symbolic domain, of the two most complex composition modules identified at the CP level, namely the Floating Point Adder and the Floating Point Multiplier systems.

2.5.2 Design at the CP Level

The principles of hierarchical decomposition are applied at the chip level (in designing a custom IC) to partition the particular chip into a number of relatively complex composition modules, such as arithmetic logic units, random access memories, input-output ports, control logic etc. In many cases such subcircuits emerge naturally from the functional specification of the chip, ie floating point adder or multiplier, or data memory. The process of decomposition, at this level of hierarchy, is guided by the specifications developed at the TOP level of the design. Such specifications are formalized by deciding on such architectural aspects as the timing strategy, eg a two or a three phase clock, the control strategy, eg microcode or random logic or other scheme, instruction set etc.

Following the partitioning process down to the MCM level, functional (how it contributes to the overall function) and interface (how it fits in with the rest of the system) descriptions for each major composition module are developed. The descriptions are translated into a high level language and a functional simulator is used to evaluate and validate the particular partitioning. In addition estimates are also made of the amount of logic necessary for each subcircuit to obtain a measure of the total chip silicon area

required and the power dissipation at this early stage of design. The size estimates of the various composition modules are used to produce a preliminary layout (floorplan) of the complete chip. This ensures that the two-dimensional nature of the medium is given early attention and allows for the physical problem of global communication overhead to be minimized. For example, major composition modules interacting with high frequency are placed in physical proximity since the silicon area and time delay penalties of long high bandwidth communication channels are severe.

An example of a system (the Mathematical Processor chip) floorplan at the MCM level is shown in Figure 6.16. It is a geometrical plan that describes the position, shape, size and interconnect of the various parts of the system, typically traversing one level of hierarchical decomposition. The floorplan can be regarded as being the *environment specification* of a module. It represents a basic technique available for the design of custom circuits that allows for the refinement of functional description and the corresponding refinement of the topological description to be carried out in parallel, in the top-down design.

At this point, a decision, based on the results of the functional simulation and the floorplan, regarding the suitability of the partitioning can be made. If the preliminary layout does not conform to the size restrictions of the chip or if the circuit validation process identifies signal paths that are longer than desired, the partitioning at this level must be modified, otherwise the design of the various functional blocks identified at this level can be initiated. Before the discussion of the design process is continued, the major floorplanning techniques are described.

In this thesis the basic structured floorplan style which incorporates many of the features of the Caltech Structured Design Style is adapted from that described in [MeCo80] and [TRLG81]. The following section provides an overview of structured floorplan and techniques applied for its generation.

Floorplanning

In generating a floorplan at a given level of hierarchical decomposition each module is described, as illustrated in Figure 2.2(a), in terms of:

1. A rectangle, drawn to a given dimensional scale, that represents the outer boundary of the module and indicates the area occupied by that module. The name of the module is given inside the box. All of the components of the given module are confined to the allocated area.

2. A set of input and output ports that are located on the periphery of the bounding box. These provide the only points of interconnecting the module's internal geometry to the outside environment.

At a lower level of decomposition, a composition module may be described in terms of its external interface, just discussed, and its internal components and their interconnections, Figure 2.2(b). The internal implementation of a leaf module is specified in terms of circuit primitives, ie the actual physical layout. The task of producing a physical layout of leaf modules is carried out in the implementation phase of the design and does not enter into the floorplanning process. Hence such modules are treated in floorplans as having external interface only.

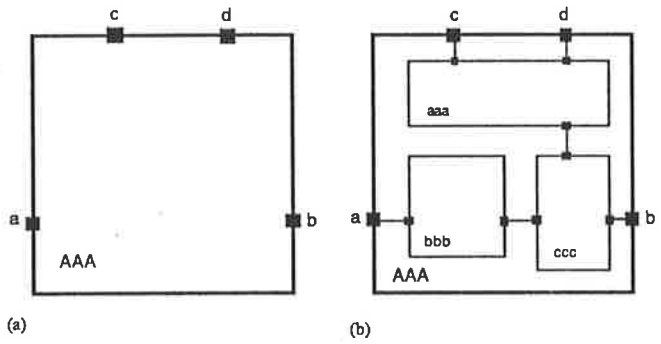


Figure 2.2: Module AAA: (a) TOP level floorplan, (b) floorplan at the first level of decomposition.

Restricting the bounding boxes to rectangles simplifies the task of composition and placement, since rectangles pack better than irregular polygons. The provision of ports as the only allowed points of access to the module contents simplifies the design of interconnect. A further simplification in interconnect complexity is achieved by careful design of module external interface. In particular, in creating a floorplan for a given composition module the components are restricted to interconnection by abutment where possible, Figure 2.3(a). In other cases the interconnect is restricted to run between the boundaries of adjacent modules. This leads to short routing runs which can be implemented as simple river routes (or channel routes) as shown in Figure 2.3(b).

There are two primary techniques used to impose structure on floorplan designs: the creation of regular instances and regular interconnect. An example of a floorplan exhibiting both of these features is a composition module defined as a linear array of a number of instances of its components. Linear arrays have two main advantages: only one module being instanced at the lower level of hierarchy needs to be created (this achieves significant gains if it is a leaf module), and the instances can be easily de-

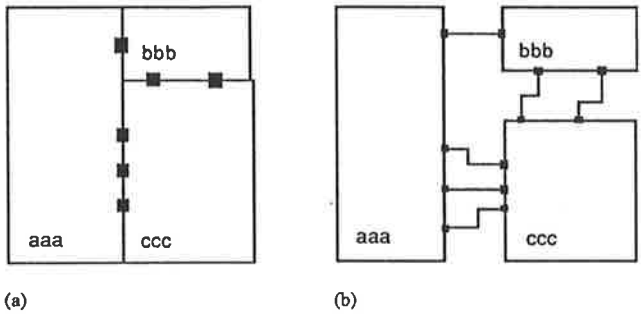


Figure 2.3: Module interconnection: (a) by abutment, (b) using river routing.

fined to interconnect by abutment. An example of a floorplan including a 4-bit ripple carry adder realized as a linear array of instances of 1-bit adder leaf modules is given in Figure 7.23.

In cases where two different modules are to connect to each other a greater interconnect regularity can be achieved by proper module pitch matching, as depicted in Figure 2.4(b).

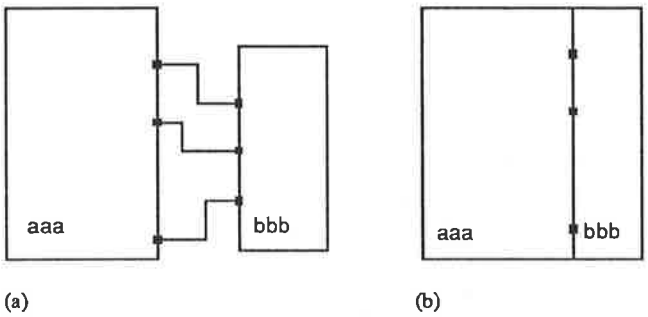


Figure 2.4: Module interconnection: (a) using river routing (b) by pitch matching, following a suitable modification of module bbb.

A common method of interconnecting a number of different modules is via a common bus. The presence of such buses is often a central constraining feature in a floorplan. A regular pattern of interconnection is achieved in such situations by *embedding* the bus (global interconnection) inside the modules, Figure 2.5(a), rather than adopting an implementation in which the bus exists as a separate floorplan entity, Figure 2.5(b). This approach is often possible since the presently available CMOS technology makes typically two layers of metal and one layer of polysilicon available. By suitable allocation of interconnection layers the bus may be implemented using the second metal layer for example, in which case it can be simply placed *on top* of active geometry inside the modules being connected together.

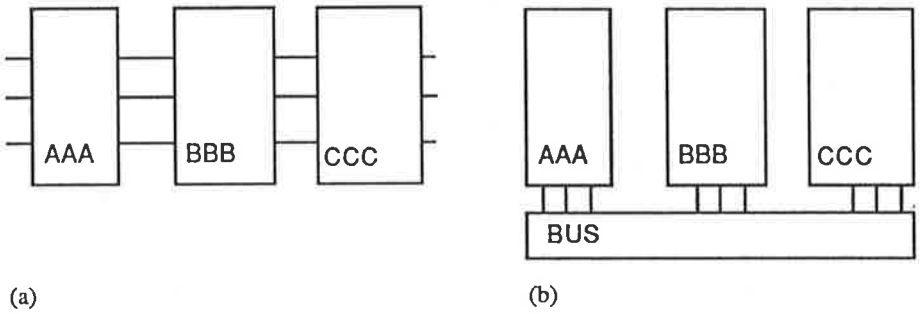


Figure 2.5: Module interconnection: (a) by an embedded bus, (b) by an external bus.

It is also desirable, but not strictly necessary, to achieve a global subsystem interconnection pattern without interconnections implemented on the same routing layer having to cross one another, a condition termed *planarity fault*. Planarity faults are costly because:

1. Contacts or vias are necessary to change routing layers consuming area and increasing resistance, capacitance and fabrication fault probability; and
2. The available cross-over layer may exhibit less desirable electrical characteristics than the routing one, eg polysilicon cross-over compared to a metal routing.

Thus, one of the objectives of generating a floorplan is to minimize the number of planarity faults especially at the first level of decomposition where the associated costs are particularly severe (eg two 32-bit buses crossing each other). In situations where a planarity fault cannot be avoided, the expense incurred should be minimized. This can be achieved, for example, by crossing over a narrow rather than a wide routing run, Figure 2.6.

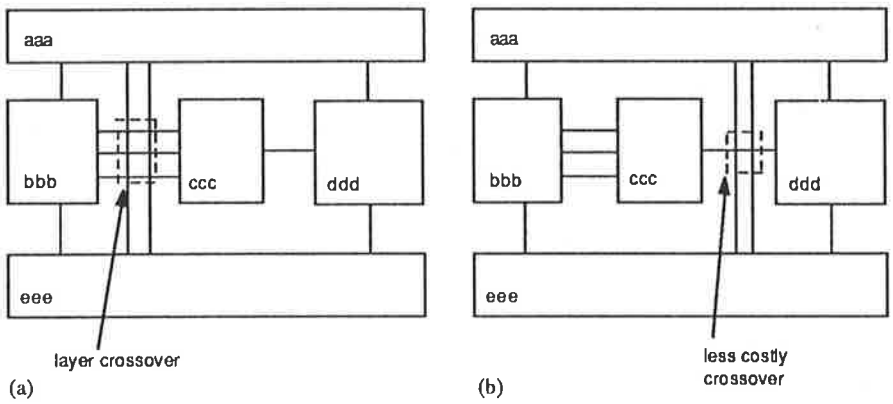


Figure 2.6: (a) More expensive and (b) less expensive cross-overs.

2.5.3 Design at the MCM Level

The complexity of a given composition module will generally dictate the approach taken in its implementation. If the particular module is a memory block then it is likely that it will already be available in the module library or, because of its inherent regularity, an automatic means for its generation will be available. The same is true if the required module can be implemented as a programmable logic array (PLA). In many cases, however, the complexity of the module is such that it can be considered to constitute an independent system, at this level of hierarchical partitioning, and it becomes possible to directly apply techniques of structured design methodology for its design. The Floating Point Adder and the Floating Point Multiplier systems identified at the CP level of the Mathematical Processor system belong to this category of composition modules. In general the design of major composition modules proceeds by producing an hierarchical decomposition of sufficient depth, such that the functional modules at the lowest level of this hierarchy are simple enough to be managed by the designer or automatically by design tools. This process is guided by the area, interface and performance constraints which have been imposed on the given module at the CP level of design decomposition.

The module is examined in all description domains; behavioural, structural and physical, to evaluate the possibility of its being implemented in the particular technology subject to the particular constraints. The restrictions of area, power dissipation or time delay may dictate the use of one circuit technique over another, ie fully combinatorial, dynamic or pseudo-nMOS, and may also determine whether a standard structural implementation can be used or whether a novel one needs to be developed.

The complexity of the composition module is reduced by introducing regularity into its description, as discussed in Section 2.4.3. Very frequently, functional modules with the same functional behaviour but with different port placement are used in a number of places in the subsystem. This regularity of function has been utilized in this work by developing a library of *semi-customized* functional modules. One particular module consisted of a 1-bit incrementer. Rather than producing a number of implementations characterised by different port placements or different bounding box aspect ratios, only one implementation of this module was realised in order to reduce module library complexity. The implementation of this module was such that it was a trivial matter to place ports in a restricted number of places around its boundaries, making it suitable for instantiations in various positions of the system.

In any design the circuit performance is as important as its functional correctness. Particular attention needs to be paid to any subcircuit containing a potential *critical*

signal path that will ultimately determine the performance of the complete system, such as placing a lower limit on the system clock period. The design and partitioning decisions that may have to be taken, at the MCM level of design, in order to optimize such a critical path must also be considered at the CP level, as they may have global repercussions. For example, a good starting point to increasing the speed of propagation of the carry bit across an n-bit parallel adder is to make the pitch of the individual bit slices as small as possible. All other modules which form parts of the system, as the adder, need to have the same pitch for abutment to be possible. Thus, the aspect ratio of the bounding box of the adder module determines the aspect ratios of the bounding boxes of many other modules, affecting a large portion of the system at the CP level. This is an example of how the top-down design style is advantageously combined with the bottom-up design style.

The decision regarding what part of a composition module is to constitute a functional module is a trade-off, matching a desire to produce a self contained functional entity with a desire to limit design difficulty. If a functional module is made too small then it will not be complex enough to perform a specific function, and if it is made too large then it may be too difficult to understand all its aspects, and thus to design or modify it. In order to produce an efficient implementation of a composition module from the silicon area point of view, the designer uses experience gained in previous projects and the making of some decisions in a bottom-up fashion to decide at what level of hierarchical decomposition to stop the partitioning. Decisions made in this way can lead to space efficiencies which are not possible if too zealous a pursuit of decomposition is undertaken. For example, a strict top-down decomposition of a parallel integer multiplier system may result in the system hierarchy as shown in Figure 2.7(a). By terminating the decomposition at the BMC module level, however, rather than decomposing it further into a CSA (carry save adder) and AND (and gate) functional modules, the same functionality may be achieved at a smaller expense in silicon area, Figure 2.7(b).

The final product of this design stage is a floorplan of the particular major composition module being considered, (or a number of floorplans) showing its composition in terms of its functional modules, together with the functional and interface specifications of all functional modules. Examples of such floorplans are shown in Figures 7.10 and 8.37.

2.5.4 Design at the FM Level

The behaviour of each functional module identified in the decomposition of a particular major composition module is specified, most commonly, in the form of a schematic

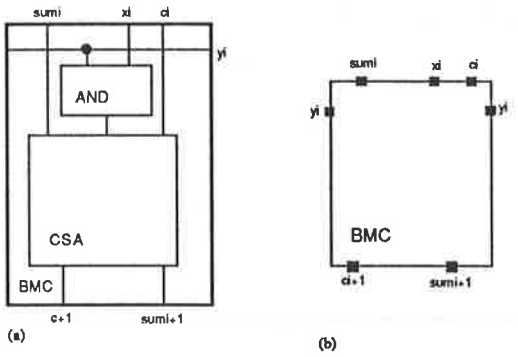


Figure 2.7: Partitioning process: (a) down to more fundamental functional module level, (b) terminated at a less fundamental functional module level.

diagram. This logic representation may be simulated using a logic level simulation program or alternately it may be modelled in a high level language and simulated using a functional simulator, in order to validate its correctness and to facilitate the preparation of test patterns. In the next stage, the logic design is reduced to transistor level, although in practice logic and transistor level design my not be clearly distinguished.

The functional module designs are committed to a geometric format which may be in mask or symbolic representation. Circuit extraction is then carried out on the geometric representation for verification, performance improvement and circuit parameter extraction purposes. The process of verification and performance improvement of the functional modules is most commonly carried out with the aid of a transistor level circuit simulation program. An accurate circuit description obtained from the circuit extraction process and accurate transistor model are essential for these design tasks. The delay time along any critical signal paths is examined by performing circuit simulations and transistor sizes adjusted accordingly, to reduce the maximum delay to an acceptable level. Typically, a number of iterations through logic, transistor level design, geometric layout and cell verification will occur before a particular physical implementation is finally accepted.

Provided sufficient computing power is available a complete major composition module may be extracted and simulated at the transistor level to accurately predict its performance. Such a process was carried out for the Floating Point Adder system. In many cases, however, this is not possible. In such instances only a part of the complete module, such as its critical signal path, may be extracted and simulated, as was done in the case of the Floating Point Multiplier system.

The physical implementation of functional modules will be discussed in more detail in Section 2.6.4.1, in conjunction with the discussion of the software tools used for this

task.

2.5.5 Composition

With each major composition module of a particular chip designed in the manner just described the floorplan defined at the MCM level of system partitioning is used for the process of *chip assembly*. This is in general a fully automatic step with all aspects of signal and power routing and pad placement handled by the software [AKNS82]. If the design is composed of a number of individual chips, the custom designed chips and any commercially available chips used are then assembled on printed circuit boards to complete the composition process. A more detailed discussion on this subject may be found in [LBBR81] and [DaGw82].

As pointed out in the introduction of this thesis and this chapter a suitable set of software tools is indispensible for an efficient and error free design of integrated circuits. The next section describes the software tools which were available in the course of this work, in an attempt to demonstrate how these tools influenced the scope of the VLSI system design undertaken in this research.

2.6 The Software Tools

Software tools, in general, may be subdivided into two groups: those used in the planning phase of design and those used in the implementation phase of design. Software tools that are of most value in the planning phase of design include automatic or manual floorplanners and functional simulators. Software tools that are commonly used in the implementation phase of design include functional module layout tools, composition module tools, structure generators, design rule checkers, logic and transistor level simulation programs and critical path optimizing programs.

The software tools available in the course of this work and the way they were employed are discussed in some details in the subsections to follow.

2.6.1 Floorplanning

As discussed earlier, floorplanning is an essential design task for design partitioning and composition. No floorplanning software tools were available in the course of this work, and all aspects of floorplanning were carried out by manual means.

2.6.2 The Functional Simulator USA

Functional simulators permit testing, evaluation and experimentation with different architectural realizations of the VLSI system by providing means of conveniently defining (in a high level language) models of circuit structure and behaviour [BrTF83], [DaSh83], [Barr84], [Coel84], [Disn88]. The advantages of using a functional simulator in the planning phase of the design are many. Some are given below.

1. It is possible to describe the system at all levels of abstraction in the behavioural and structural domains. This allows the user to eliminate all but the essential implementation details and thus make it possible for exercising particular features of the proposed design with greater ease than is obtainable with logic or transistor level simulation techniques.

2. New system implementation ideas may be efficiently tested and evaluated, since a modification to a design models involves just an editing session of the circuit description, given in a hardware description language, and subsequent compilation of the code.

3. It provides effective means for validating the structural and behavioural descriptions of the system at all levels of decomposition. Some of the functional simulators attempt to perform formal proofs of design correctness [Barr84].

4. It provides means for documenting the system in the structural and behavioural domains.

An experimental version of a functional simulator, The Universal Systems Analyser (USA), became available in the later part of this work and some use was made of it in the design of the Floating Point Adder system. The full documentation and implementation issues can be found in [Disn88] (USA was formally called Pink) and [ISD88], and only an outline of the features that were found most useful will be presented here. The methodology of VLSI design based on the use of a functional simulator is also described.

The USA is a general purpose software system for the description and modelling of complex systems in many disciplines, eg biology, medicine, engineering, but the characteristics of its description language, ISSL, and the simulator itself are particularly apt in the architectural development phase of VLSI systems. The key principle, which gives USA much of its power and flexibility, is the provision of suitable abstractions in system description. The system description language provides constructs suitable for hierarchical system description and for an efficient way of expressing the regularity of interconnection and function. In addition, it enforces the use of separated hierarchy, ie

each module is described in terms of its input and output ports and its functional behaviour (functional module) or in terms of its input and output ports and instantiations and interconnections of lower level modules (composition module).

As the first stage of the planning phase of the design work in which a functional simulator is employed, the translation of the system function from the free form specification (see Section 2.5.1) to one of the two formal descriptions in ISSL (functional or topological) is necessary. Such translation is required so that the various tasks for which the simulator is required can be carried out. In general the TOP level behavioural description of the system is produced before any partitioning is attempted. Arguments for proceeding in this way are:

1. The assistance in checking the adequacy of the initial system specification which is provided by the translation process is available at the earliest time;
2. The TOP level system description is later used for the validation task of a proposed system partitioning (Section 2.6.2.1);
3. The designer, who will be searching for an optimum design, achieves an initial formal system description unburdened by a commitment to a particular partitioning;
4. Construction of both TOP level description and a formal decomposed description increases understanding of the system behaviour and reduces the chance of error in the translation; and
5. A significant part of the resulting description may be reused in the production of the one or more partitioned descriptions which will be required later.

The use of the USA functional simulator was limited in this work to two basic modes: interactive and module based simulations, explained below. In the interactive or browsing simulation the simulation is specified, initialized and subsequently controlled in an interactive manner via the use of the simulator command interface. The user generates (on line) system stimulations and observes outputs and signals at various points of the system by applying appropriate simulator commands. This type of simulation is useful for gaining an introductory understanding of the system, for searching for the causes of system irregularities or for *informally* validating the system behaviour and partitioning. The simulator may also be used for *module-based* simulations. In this case, input is automatically generated, as needed, by stimulator modules described in ISSL and connected as signal generators to the input ports of the system being simulated. The output of the system may be assessed either via the built-in display features of the simulator (as done in the browsing simulations) or by an assessment module also defined in ISSL

and connected to the system output ports, to check for particular output conditions. The most important application of this style of simulation is to perform validation of a particular design partitioning being studied. This design task is discussed in more detail in the following subsection.

2.6.2.1 Validation

All translations and partitionings need to be validated as described below. Validation of translation, ie the process of describing the TOP level system behaviour in ISSL, involves the comparison of outputs calculated from the free form system description with the outputs calculated from the formal system description, for a sufficiently large set of input vectors. Validation of partitioning involves a similar comparison of outputs calculated on the one hand from the formal functional description of a module before it is decomposed, and on the other hand from the topological description of the module after it has been decomposed.

The process of validation just described is seen to be arithmetical rather than algebraic, ie it involves the calculation of system or module responses to a set of particular inputs or test vectors. The fact that validation is performed in this manner derives from the fact that algebraic techniques, to prove the equivalence of disparate system descriptions, do not yet exist for either free form system descriptions or descriptions written in languages, such as ISSL, of applicability sufficiently wide to be of practical use. As discussed in [Barr84] attempts are, however, being made to produce software tools that will formally prove in an algebraic manner the correctness of a particular system partitioning.

It will be clear that the extent to which the validation process does indeed assure the equivalence of two descriptions depends on the extent to which the test vectors fully exercise the system. In simple systems a complete set of test vectors covering all possible inputs can be assembled and the necessary simulations performed, while in more complex systems this is just not practicable. In such a case the designer must wisely choose the inputs to be used and also be satisfied, at least in the early stages of design, with an incomplete assurance of equivalence.

Figure 2.8 shows how a validation of system partitioning task can be accomplished by means of a module-based simulation involving the original system description and additional generator and assessment modules. This is an important part of the top-down, correct-by-construction circuit design. The correctness of the partitioning is judged on the equality of the outputs produced by the left module (a functional module representing the TOP level system behavioural description) and the right hand module (a

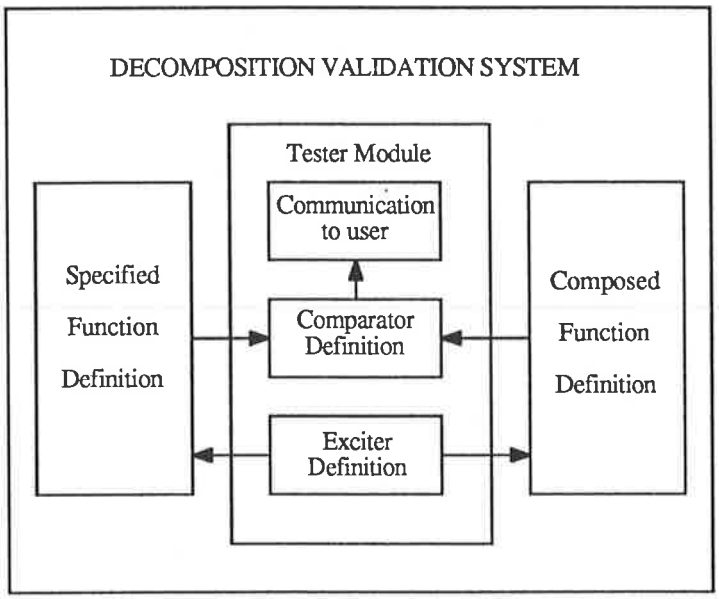


Figure 2.8: Module based validation.

topological module describing the TOP level of hierarchy description of the system) when stimulated by a sufficient set of test vectors. The function of the Exciter module is to generate suitable test vectors which might be random or systematic. The function of the Comparator module is that of collecting and examining the equality of the decomposed and the undecomposed system outputs, and the function of the Communication module is that of reporting to the user when the two outputs are not equivalent. To perform the validation of system partitioning task, the above augmented system is set up and set running for the chosen number of inputs. If no disagreement in outputs is reported, the partitioning is regarded as valid to a certain level of confidence. The methods for choosing a suitable set of test vectors is a field of active research beyond the scope of this work [WiPa83]. The approach taken in this work was to apply the exhaustive set of inputs to the complete or a reduced version of the system being considered.

The validations to be performed include not only initial translations and partitionings but also the validation of the code for each functional block at the lowest level of decomposition. This need arises because decompositions involving lowest level functional modules are created by the designer against informally expressed or unexpressed beliefs of the functions of those modules, and there is a need for the designer to gain assurance that the formal expressions of the functions of those modules are in accord with those beliefs. These validations can normally be done informally, using interactive simulations, as the modules are usually simple enough for the designer to comprehend every aspect of their function and the number of inputs is usually small, thus the probability

of human error is low.

2.6.3 Symbolic Design Method

The simplified lambda based design rules were introduced by Mead and Conway [MeCo80] to help the designer focus on the higher level aspects of the layout rather than be burdened with the much more detailed and complex industrial set of rules. The symbolic layout method [West80] was introduced to further simplify the layout process. This method provides a simplified means for manually generating a circuit layout using symbols to represent circuit primitives, such as wires, transistors and contacts, thus relieving the user of the necessity of satisfying process design rules in producing the circuit layout. In order to facilitate easy design capture and simplify the necessary tools the circuit elements are snapped to a "virtual grid". The adjustment of the device dimensions and positions to satisfy process design rules then becomes a function of an automatic post-editing processing program called a *compactor*. An example of a 1-bit adder functional module described in the symbolic domain is shown in Figure 8.24 and the result of compaction, according to the Orbit 2-micrometer CMOS p-well process design rules [Orbi87], is shown in Figure 8.25.

The result of adopting symbolic domain design are that fewer designer errors are made, the functional description of actual circuit produced by the design process is more easily extracted from the symbolic domain description, and the post-editing compaction to satisfy process design rules can be made error free. Thus the essential step of design validation occurs both earlier and less often in the design cycle, so that designer productivity is greatly increased. The designer should, however, become familiar with the characteristics of the compactor, as he is able, through using particular circuit primitives placement strategies, to exert an indirect influence on the final mask representation, and thus optimize the designs emerging from the compaction process.

The high layout flexibility possible with the full custom design style is still maintained with symbolic domain design, and the incurred cost is in the form of a reduction in transistor densities achievable. A large degree of fabrication process independence is also achieved with the use of what are called *technology files*. The compactor uses firstly the symbolic description of a given module and secondly the technology related information contained in the appropriate file, to produce a mask domain representation of that module which satisfies all geometric design rules of the required process. The conversion of a given design to a form suitable for a different fabrication process involves generating a text file containing the geometric design rules characterizing the new fab-

rication process. It is not possible, in general, to convert a design initially produced for an nMOS process to a CMOS process because of a number of important differences, such as the need for substrate contacts in a CMOS design or the need for depletion mode transistors in an nMOS design between the two.

The use of the design rule checker is limited in such a design method to the task of verifying that a given technology file has been correctly constructed. Once this has been established, all mask descriptions generated by the compactor will be guaranteed to be free of any geometric design rule errors.

2.6.4 Implementation Tools

The implementation phase of the design work carried out during the course of this research was performed to a large extent with the aid of the VIVID[1] [VIVI85] software system and to a smaller extent with the Integrated Circuit Design's Pty. Ltd., Phase 2 [ISD87] and Phase 1 [ISD86] Software Suites.

The VIVID system has been developed at the Microelectronics Center of North Carolina and it is a symbolic layout system based on the "virtual grid" concept [West80], [WeEs85] providing a highly interactive design environment. At its present degree of development (version v1.3 running on VAX[2]780/Unix[3]) it provides sufficient capability for the design of modules containing approximately 10000 to 15000 transistors.

The VIVID software system and Phase 2 Software Suite allow the interactive editing, layout compaction, circuit connectivity extraction and circuit simulation at the transistor level within the symbolic domain. The main parts of the VIVID system include (The corresponding tools of the Phase 2 Software Suite are given in parenthesis. It should be noted, however, that the correspondence between the two software environments is not complete. Some of the main differences will be discussed in the later part of this section.):

1. ABCD – a circuit description language which makes it convenient for module layout modifications to be carried out using a text editor.
2. ICE (SYMEDIT) – a symbolic colour graphics editor for the layout of functional and composition modules. Composition based on the separated hierarchy principle is encouraged. An automatic router of limited capability is also included.

[1] VIVID is a trademark of the Microelectronics Center of North Carolina.

[2] VAX is a trademark of Digital Equipment Corporation.

[3] Unix is a trademark of AT & T.

3. Abstract (SYMNET) – symbolic circuit net generator suitable for logic and circuit simulations.

4. Compact (SYMPACK) – an automatic compactor for converting the symbolic circuit description to the mask circuit description.

5. FACTS (PROBE) – an interactive transistor level simulator. It is based on the timing simulator MOTIS described in [ChGK75].

6. ATOLL (S2C, C2P, P2S) – utility conversion program.

The strongest point of the VIVID software system is the circuit analysis program, FACTS, which allowed the simulations of composition modules of around 10000 transistors to be performed. A detailed discussion on the principles of timing and circuit simulation techniques can be found in [JLLR73], [ChGK75], [DeMa79] and [HaSa81]. The accuracy of the results produced by FACTS in its original form, however, was very poor, due to the simple MOS transistor analytical models (expressions (4.38)) contained in the program. It was decided to conduct a study of existing analytical MOS transistor models, firstly to arrive at a more acceptable model to be incorporated into FACTS, and secondly to gain knowledge and sound understanding of the operation of such devices which can be effectively used in circuit design work. The research [ZyCo87] that led to the development of accurate transistor models with geometry dependent parameters which were incorporated into FACTS will be discussed in Chapters 3 and 4. The result was a transistor level simulator capable of simulating large digital circuits with sufficient accuracy to provide useful results regarding circuit performance.

The Phase 2 Software Suite solved some of the limitations of the VIVID system. For example, the Phase 2 graphics editor, SYMEDIT, contains a nonhierarchical command menu (the command menu in ICE has a number of levels of hierarchy) and also allows commands to be invoked from the keyboard (not possible in ICE), rather than solely with a pointing device, such as "mouse". This has resulted in minimum number of actions needed to invoke the most frequently used commands and makes the manipulations on the screen more efficient than it is possible using the ICE graphics editor. An additional advantage of the editor SYMEDIT over the editor ICE is that the compactor, SYMPACK, is integrated with SYMEDIT in such a manner that an interactive compaction of a module, or part thereof, can be performed and the result displayed on the screen during the current editing session, allowing for a quick assessment of the layout to be carried out. The greatest use was, however, made of SYMPACK, the compactor of the Phase 2 system, which has a far superior performance, in terms of the cpu time taken, than the compactor of the VIVID system.

The following two sections describe in some detail the method in which the available software tools were applied to produce functional and composition modules.

2.6.4.1 Module Implementation in the Physical Domain

The completion of the chosen leaf module floorplan (interface specification) and schematic circuit description (functional specification) constitutes the point in the design cycle, at which the designer starts making use of the implementation software tools viz the VIVID or the Phase 2 Software Suite. The steps followed from this point in a leaf module implementation process are as follows:

1. The symbolic representation of the chosen module is made with the use of ICE (or SYMEDIT if the Phase 2 Software Suite is used). This representation contains an implicit specification of the schematic circuit as well as a specification of the topological placement of transistors. A more detailed discussion of this process is given in the following subsection.

2. Network list description of the leaf module is then produced from the symbolic description with the use of the symbolic circuit connectivity extractor Abstract, (or SYMNET). Behavioural aspects of the module are then examined with the circuit simulation program FACTS (or PROBE) which provides detailed display of the voltage waveforms at any node of the circuit.

3. Functional simulations of the specified geometry thus performed with FACTS are compared with the specified behaviour and any necessary modifications to the layout at the symbolic level are made using ICE until the desired function has been achieved.

4. At this point, the leaf module implementation process enters a second stage. The symbolic description of the leaf module is now optimised, using the graphics editor and the compactor in an iterative fashion, to produce, after compaction, a mask specification which meets, as far as practicable, the desired area and the aspect ratio criteria set for the bounding box of the module. A further symbolic circuit extraction with Abstract (or SYMNET) and simulation with FACTS (or PROBE) is then performed to verify that any adjustments to the symbolic layout, that may have been made in the process of area optimization, have not altered the functional behaviour of the leaf module.

The process described above is followed in the implementation of all the functional modules required within the system and not available from other sources. The assembly

of the full system then proceeds at the major composition module level, again using ICE (or SYMEDIT) through all the levels of the module hierarchy. At each level of the hierarchical decomposition circuit extraction and simulation at the symbolic level are performed to confirm that the desired function and performance has been achieved.

For the final compaction of the complete system, the tool Compact (or SYMPACK) are typically used off-line. The circuit extractor NET (part of the Phase 1 Software Suite) may be used, following a compaction, to extract the network list description of the entire design from the mask domain description to calculate more accurately parasitic capacitances and transistor sizes which are used by the simulator. If at this stage of the design cycle the timing specifications are not met, the design of the leaf module causing performance degradation needs to be re-examined.

Finally the conversion of the mask abstraction detail to a standard fabrication interface is performed. The output from Compact (or SYMPACK) is translated into the Caltech Intermediate Form (CIF) format, for example, using the utility program ATOLL (or S2C). This step also provides an interface to the Phase 1 Software Suite. The hierarchical Phase 1 mask editor, PLAN, provides a means for the assembly of the completed design with other cells, such as propriety fabrication house pads available only at the mask level. This task was not supported with the available versions of the VIVID (version v1.3) and Phase 2 (version v.1) systems.

It was found in this work that the task of system composition at the CP, or even at the MCM level, is not efficiently supported with either of the software systems. This is mainly caused by the fact that all composition tasks need to be performed manually using the graphics screen editor, and by the limited field of view of the monitor screen. For example, when a particular composition module gets to the size where it is not possible to display it in its entirety on the screen at an acceptable resolution, the composition task becomes very tedious, as a result of the many windowing and panning operations that need to be performed. It was also found that although the task of implementing a functional module can be carried out, given sufficient experience, much more efficiently in the symbolic domain than in the mask domain, the task of "debugging" the implementation using the layout-extract-simulate cycle, described above, can also be time consuming in many instances. New software tools, such as automatic chip assemblers, floorplanners and functional module generators are currently being developed at Integrated Silicon Design Pty. Ltd., to complement and enhance the Phase 2 symbolic domain software system.

2.6.4.2 A Symbolic Layout Style

The layout of a functional module, assuming CMOS technology, is carried out using an approach adapted from the layout techniques described in [Wein67], [LoLa80], [AnRe82]. The process may be summarized as follows.

First the vss and vdd power rails are placed, using the first metal layer, to define the overall bounding box of the module. An imaginary demarkation line is placed in parallel with the power supply rails and dividing the bounding box in halves. This demarkation line is used to separate the p-channel transistors, which are placed between the demarkation line and the vdd power rail, from the n-channel transistors which are placed between the demarkation line and the vss power rail. This layout style enables an area efficient placement, performed by the compactor, of the p-well and the p-plus (assuming a p-well CMOS process) masks.

The devices are mostly placed in horizontal orientation with as many devices of the same type abutted together as possible to minimize the interconnect capacitance and resistance. In some cases, however, a vertically oriented device provides a better area efficiency and this possibility is also explored.

To guard against latch-up no more than four horizontal levels of devices are placed between a power rail and the demarkation line. As a further precaution against latch-up, at least one vss contact is placed for every group of four n-channel devices and a vdd contact is places for every four p-channel devices [WeEs85].

To effect transistor interconnection polysilicon is normally allowed to run vertically, connecting as many gates as possible, but in some instances polysilicon is run horizontally to provide additional connections. The first metal layer is run in both vertical and horizontal directions; this layer of interconnect is primarily used to provide connections over large distances, such as control or data lines spanning a number of functional modules, and for the purposes of power distribution within leaf modules. Diffusion, as a means of interconnect, is only used for short distances and mainly to effect connections to the power rail.

Due to the relatively large pitch of the second metal layer and the relatively large area requirements of the vias, this layer of interconnect is mainly used for providing inter-module interconnections. In particular, since the module power rails run, in general, horizontally in first layer metal, the second layer of metal may be used for routing signals between modules in the vertical direction where the use of the polysilicon layer would result in performance degradation. An example of a module layout produced using this style is shown in Figure 8.24.

2.7 Summary

The VLSI medium presents the designers of microelectronic systems with the possibility of producing circuits with functional densities and concurrencies previously not economically possible. There are several features of this medium that make the design task particularly prone to complexity problems: the largely unstructured nature of the VLSI medium, problems deriving from functional concurrency, communication characteristics of the silicon surface, constraints deriving from the two-dimensional nature of the surface, rate of technological evolution, normal design trade-offs, and the issues of circuit testability and fabrication yield.

The highly partitioned design style attempts to contain the complexity of VLSI design by imposing a rigid structure onto the surface of the silicon chip and automating most of the design tasks. In the resulting circuit implementations the interconnect dominates and prohibits the best utilization of the medium to be achieved. The full custom design style, applied to functionally partitioned designs, offers the designer full freedom of expressing his ideas, by not restricting him in any way regarding the placement and interconnection of transistors. Circuits implemented in this way have a greater scope for design innovation and for making the best use of the available silicon area. These gains, however, are achieved at the expense of higher design complexity.

In order to design successfully a VLSI system, it is necessary to employ a design methodology that offers techniques for managing the complexity associated with the design of large systems. The two most important techniques which form the basis of the structured design methodology are the use of hierarchy in describing system structure and the recognition of regularity in system structure. Structured floorplans, which incorporate these complexity management techniques allow for the physical limitation of the silicon surface to be considered from the earliest stage of the design. In addition, floorplans provide a basis for the evaluation of a particular system partitioning in the physical domain, during the planning phase of design process, and are used as a guide in the task of chip assembly, during the implementation phase of design.

The VIVID system and the Phase 2 Software Suite, both of which are based on the description of circuits in the symbolic domain, formed the basic software tool sets used in the design work undertaken in this research. The symbolic style of design provides a convenient means of functional module implementation, by eliminating the need to consider process design rules. Both sets of tools, however, were found to be inefficient in the implementation of circuits at the CP and MCM levels of system decomposition. While it was recognized that the effort required for the improvement of the complete

sets of the software tools was outside the scope of this work, the need for an accurate circuit simulator capable of analysing large digital circuits resulted in some effort being directed towards improving the accuracy of the circuit analysis program FACTS. This work is described in the next two chapters.

Chapter 3

A Comparative Study of Analytical MOS Transistor Models

3.1 Introduction

This chapter examines and compares a number of analytical models for the current-voltage characteristics of MOS transistors. Such models were first derived for large geometry transistors. With a reduction in transistor physical dimensions, however, these characteristics cannot be accurately predicted by the large geometry models. This failure of the large geometry transistor models is the result of a number of phenomena, commonly known as *short channel*, *narrow width* and *small geometry* effects becoming significant.

A considerably large number of publications has appeared describing techniques developed for incorporating these effects into large geometry MOS transistor analytical models, thus, extending their range of application to small geometry MOS transistors. This chapter presents a comprehensive review of this large number of works, with an emphasis on the understanding of the device analytical models, on elucidating their major differences and similarities (not always obvious), and on assessing their practicality to circuit simulation. The requirement for such a study arose, as pointed out in the previous chapter, from the need to improving the accuracy of the circuit simulator, FACTS, used in evaluating the circuits designed in this work. Only the models for the linear and saturation modes of operation are of main interest because mainly static digital circuit design techniques, for which the subthreshold current can be neglected, are used in this work.

The chapter opens with the derivation of the common models for the current-voltage

characteristics of large geometry MOS transistors. Next, the various effects which become significant with decreasing transistor dimensions are discussed, and the different approaches to model them are studied and compared. Finally, the most commonly employed techniques for incorporating these effects into expressions for current-voltage characteristics are examined and compared. The comparisons are mainly carried out on the basis of model complexity and the required number of empirical parameters. The accuracy of the studied models has also been assessed by comparison with experimental results and some discussion on this topic is given in the following chapter.

3.2 Large Geometry MOSFET Models

This section examines the classical equations and the underlying physical assumptions which form the basis for practically all analytical transistor models used in today's circuit analysis programs.

The notation used in this chapter follows, as closely as practicable, the commonly accepted use of symbols for various physical quantities. However, to avoid any misinterpretation, all symbols are defined as they are introduced.

3.2.1 Pao – Sah Model

The most rigorous analytical model for large geometry MOS transistors was derived by Pao and Sah [PaSa66a], [PaSa66b]. The major assumptions made in the analysis of [PaSa66a] are summarized below. An n-channel, enhancement mode device whose simplified geometry is given in Figure 3.1 is considered in this chapter.

1. The *gradual channel approximation* applies. This idea can be mathematically expressed as $\frac{d\psi}{dy} \ll \frac{d\psi}{dx}$ at any point along the channel (where ψ is the electrostatic potential at a point in the substrate referenced to the source potential), ie a long channel is implied;

2. The hole current is small compared to the electron current in the channel;

3. The channel depth is negligibly small and it can be assumed that $\frac{\partial \phi_n}{\partial x} = 0$ across the channel, where ϕ_n is the electron quasi-Fermi level referenced to the bulk Fermi level (ϕ_F);

4. The junction leakage current in the drain junction is negligible;

5. The variation of the width (W) of the channel along its length is negligible;
6. The electron mobility (μ) in the channel is constant;
7. Einstein's relationship is valid;
8. The substrate doping (N_A) is uniform;
9. The substrate is nondegenerate; and
10. The drain current beyond saturation is constant.

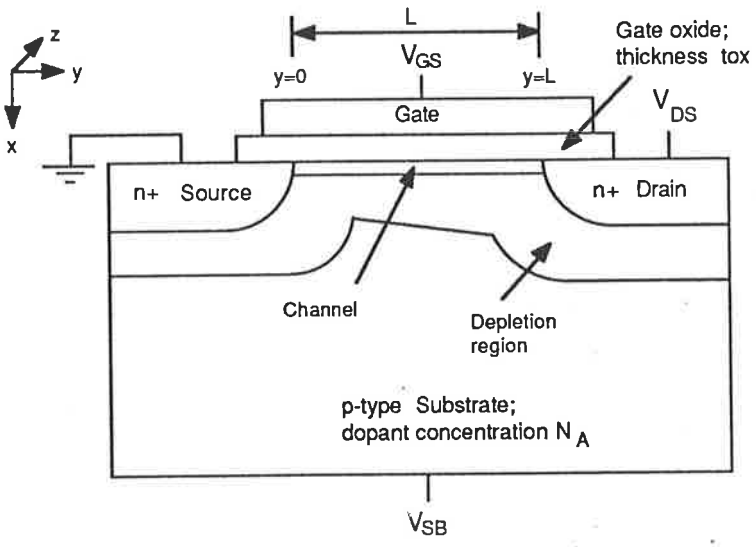


Figure 3.1: The basic geometry of a MOS transistor.

In light of the above assumptions the electron current density (J_n) in the y direction is given by:

$$J_n = -q\left(\mu n\frac{\partial\psi}{\partial y} - D_n\frac{\partial n}{\partial y}\right), \tag{3.1}$$

where n is the free electron density in the channel, q is the electronic charge and D_n is the electron diffusion coefficient. For nondegenerate materials (assumption (9)), the electron density is given by [PaSa66a], [Sze81], [Pier83]:

$$n = n_i e^{-(V+\phi_F-\psi)\frac{q}{kT}}, \tag{3.2}$$

where $V = \phi_n - \phi_F$ is the difference between the quasi-Fermi level for electrons (ϕ_n) and the bulk Fermi level (ϕ_F) (the quantity V is often referred to as the "channel voltage" [PaSa66b]), n_i is the intrinsic free electron concentration and kT is the thermal energy.

Using expression (3.2) together with Einstein's relation $D_n = \frac{kT}{q}\mu$ [Sze81], [Pier83], expression (3.1) becomes:

$$J_n = -q\,\mu\,n\,\frac{\partial V}{\partial y}. \tag{3.3}$$

The corresponding drain current (I_{DS}) is found by integrating (3.3) over the active cross section area of the channel and using assumption (3), ie:

$$\begin{aligned} I_{DS} &= -\int_0^{x_i}\int_0^W J_n dx dz, \\ &= \left(-W\frac{dV}{dy}\right)\left(-q\int_0^{x_i}\mu\, n\, dx\right), \\ &= -\mu W Q_n \frac{dV}{dy}, \end{aligned} \tag{3.4}$$

where x_i is the channel thickness, ie a point in the bulk beyond which the electron density is negligible, usually taken as the point where $\psi = \phi_F$ [DiGr66], [PaSa66a], [Das68], [Sze81]. Separating the variables and integrating over y from $y = 0$ to $y = L$ and over V from $V = 0$ at the source end of the channel to $V = V_{DS}$ at the drain end of the channel (where $V_{pq} = V_p - V_q$ is defined as the voltage at point p with respect the voltage at point q) gives:

$$I_{DS} = -\mu\frac{W}{L}\int_0^{V_{DS}} Q_n\, dV, \tag{3.5}$$

where Q_n is the charge per unit area in the yz plane due to the minority carriers within the inversion layer:

$$\begin{aligned} Q_n &= -q\int_0^{x_i} n\, dx \\ &= -q\int_{\psi_S}^{\phi_F} \frac{n(\psi)d\psi}{d\psi/dx}, \end{aligned} \tag{3.6}$$

and where ψ_S is the channel surface potential taken with respect to the source potential. The expression for $d\psi/dx$ (where the electric field E is given by definition by $E = -d\psi/dx$) can be found by solving the one-dimensional Poisson equation for the surface space charge region at a point y along the channel (see appendix A of [PaSa66a]):

$$-\frac{d\psi}{dx} = E = \frac{kT}{q}\frac{1}{L_{D_i}}F(\psi, V, V_{SB}), \tag{3.7}$$

$$\begin{aligned} F(\psi, V, V_{SB}) &= \left\{ e^{(\psi - V - \phi_F)\frac{q}{kT}} + e^{(\phi_F - \psi - V_{SB})\frac{q}{kT}} + \left[\frac{q}{kT}(\psi + V_{SB}) - 1\right] e^{\phi_F \frac{q}{kT}} \right. \\ &- \left. \left[\frac{q}{kT}(\psi + V_{SB}) + e^{(-V - V_{SB})\frac{q}{kT}}\right] e^{(-\phi_F)\frac{q}{kT}} \right\}^{1/2}. \end{aligned} \tag{3.8}$$

Expression (3.8) is the field function and L_{D_i} is the intrinsic Debye length defined as:

$$L_{D_i} = \sqrt{\frac{kT\epsilon_s}{q^2 2n_i}}. \tag{3.9}$$

In some treatments [Sze81] the constant 2 in the denominator is omitted and appears elsewhere in the analysis.

The current expression given by (3.5) and (3.6) can be written in its complete form:

$$I_{DS} = \mu\frac{W}{L}\frac{\epsilon_s}{2L_{D_i}}\int_0^{V_{DS}}\int_{\phi_F}^{\psi_S} \frac{e^{(\psi - V - \phi_F)\frac{q}{kT}}}{F(\psi, V, V_{SB})} d\psi dV. \tag{3.10}$$

If the channel width is uniform (assumption (5)), the flux lines are all perpendicular to the silicon-silicon dioxide interface. This is known as the gradual channel approximation, defined in assumption (1), according to which the Gauss theorem relating the various charges in the semiconductor, defined below, reduces to a one-dimensional form. The charges, all per unit area in the yz plane, entering the Gauss theorem are:

1. The charge induced in the semiconductor (Q_S), (also referred to as the surface charge per unit area);
2. The charge induced on the gate electrode and given by $C_{ox}(V_{GS} - \psi_S - \phi_{MS})$, where Φ_{MS} is the electron work function difference of the gate material and the semiconductor, and $C_{ox} = \epsilon_s/t_{ox}$ is the gate capacitance per unit area in the yz plane; and
3. The interface charge (Q_{ss}).

The one-dimensional Gauss theorem can then be written in the following form:

$$C_{ox}(V_{GS} - \psi_S - \Phi_{MS}) + Q_{ss} + Q_S = 0. \tag{3.11}$$

Rearranging the above equation results in an expression relating the gate to source voltage (V_{GS}) and the surface potential (ψ_S):

$$V_{GS} = V_{FB} + \psi_S - \frac{Q_S}{C_{ox}}, \tag{3.12}$$

where $V_{FB} = \Phi_{MS} - Q_{ss}/C_{ox}$, the flat band voltage, accounts for the effects due to surface states and the work function difference. The quantity Q_S under the conditions of strong inversion is given by:

$$Q_S = -\epsilon_s E_s = -\frac{\epsilon_s}{L_{D_i}}\frac{kT}{q}F(\psi_S, V, V_{SB}). \tag{3.13}$$

The complete model is specified by the set of equations (3.10), (3.12), (3.13) and (3.8). It is very accurate [PaSa66a], [Brew78], [PiSh83], [Pier83] and it is valid under all bias conditions, ie in the subthreshold, linear and saturation regions of operation. This model takes into account the diffusion current, the variation of the applied potential from the source to the drain and from the depletion region to the oxide interface, and the two-dimensional nature of the channel. However, despite the many simplifying assumptions made in this analysis, the resulting expressions are very complex and cumbersome to use.

This model became a standard against which simpler closed form expressions derived from it by other authors, [PaSa66b], [Dang77b], [Brew78], [WhWL80], [Sten83], [PiSh83], [TuMa84], have been evaluated, however, despite its high accuracy it is never used in circuit simulation programs because of its high computational requirements.

3.2.2 A Charge Sheet Model

Another model which is valid under all bias conditions is the charge sheet model developed by Brews [Brew78]. The model is in excellent numerical agreement for large geometry MOS transistors with the exact Pao-Sah model, even though a number of further simplifying assumptions are made. Starting with expression (3.1):

$$J_n = -q\left(\mu n\frac{\partial\psi}{\partial y} - D_n\frac{\partial n}{\partial y}\right) \tag{3.14}$$

rearranging, and making use of the Einstein's relation yields:

$$\begin{aligned}\frac{\partial n}{\partial y} &= \frac{nq}{kT}\frac{\partial\psi}{\partial y} + \frac{J_n}{kT\mu},\\ &= \frac{nq}{kT}\frac{\partial\psi}{\partial y} - \frac{dI_{DS}}{dx}\frac{1}{\mu kTW}.\end{aligned} \tag{3.15}$$

Using expression (3.12) and writing Q_S, the surface charge per unit area in the yz plane, as a sum of the mobile channel charge (Q_n), and the bulk charge (Q_B), both per unit area in the yz plane, the following result is obtained:

$$C_{ox}(V_{GS} - V_{FB} - \psi_S) = -(Q_n + Q_B). \tag{3.16}$$

Differentiating (3.16) and using (3.6) and (3.15), and the assumption that Q_B, ψ and ψ_S are functions of y only gives:

$$-C_{ox}\frac{d\psi_S}{dy} = q\int_0^{x_i}\left(\frac{nq}{kT}\frac{d\psi}{dy} - \frac{dI_{DS}}{dx}\frac{1}{\mu kTW}\right)dx - \frac{dQ_B}{dy}. \tag{3.17}$$

At this point in the analysis the *charge sheet approximation*, which states that Q_n is enclosed within a channel of infinitesimal thickness, is introduced. This assumption is mathematically expressed as $\psi = \psi_S$, ie the variation of the potential within the channel (ψ) from the surface potential (ψ_S) can be neglected. Introducing the charge sheet approximation and using expressions (3.6) and (3.16) to eliminate the term $q\int_0^{x_i} n dx$ gives:

$$I_{DS}dy = \mu WC_{ox}\left(V_{GS} - V_{FB} - \psi_S + \frac{kT}{q} + \frac{Q_B}{C_{ox}}\right)d\psi_S - \frac{kT}{q}dQ_B. \tag{3.18}$$

The bulk charge (Q_B), per unit area in the yz plane, is found from the depletion approximation given in [Sze81]:

$$Q_B = -\sqrt{2\epsilon_s qN_A(\psi_S + V_{SB} - kT/q)}. \tag{3.19}$$

Using (3.19) in (3.18) and integrating from source to drain gives the following expression for the drain current [Brew78]:

$$I_{DS} = \mu\frac{W}{L}C_{ox}\{(V_{GS} - V_{FB} + kT/q)(\psi_{SL} - \psi_{S0}) - 1/2(\psi_{SL}^2 - \psi_{S0}^2)$$

$$- \frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}[(\psi_{SL} + V_{SB} - kT/q)^{3/2} - (\psi_{S0} + V_{SB} - kT/q)^{3/2}]\}$$
$$+ \frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}\frac{kT}{q}[(\psi_{SL} + V_{SB} - kT/q)^{1/2} - (\psi_{S0} + V_{SB} - kT/q)^{1/2}]. \quad (3.20)$$

The values of the surface potential at the source (ψ_{S0}) and at the drain (ψ_{SL}) ends of the channel are determined from equation (3.12) and the accompanying expression (3.13) derived in the Pao-Sah model, or by a simpler result derived in [Brew78] which reads:

$$V_{GS} - V_{FB} = \psi_S + \frac{\epsilon_s}{C_{ox}L_{D_i}}\sqrt{\frac{q}{kT}(\psi_S + V_{SB}) - 1 + e^{(\psi_S - V - 2\phi_F)\frac{q}{kT}}}. \quad (3.21)$$

This is a general model valid in all regions of operations but it is substantially simpler than the Pao-Sah model. Although the value for the drain current is given in a closed form by (3.20), expression (3.21) must be solved numerically, once with $V = 0$ to determine $\psi_{S0} = \psi_S(V = 0)$ and once with $V = V_{DS}$ to determine $\psi_{SL} = \psi_S(V = V_{DS})$ for a given gate to source voltage (V_{GS}). This complexity must be accepted to make the generality of this model possible.

3.2.3 Dang's Model

Using the following classical approximations for the values of the surface potential at the source and drain ends of the channel:

$$\psi_{S0} = 2\phi_F, \quad (3.22)$$
$$\psi_{SL} = V_{DS} + 2\phi_F \quad (3.23)$$

in expression (3.20), and simplifying, produces the expression given by Dang [Dang77b], [Dang79]:

$$I_{DS} = \mu\frac{W}{L}C_{ox}\{(V_{GS} - V_{FB} - 2\phi_F + kT/q - V_{DS}/2)V_{DS}^2$$
$$- \frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}[(V_{DS} + 2\phi_F + V_{SB} - kT/q)^{3/2} - (2\phi_F + V_{SB} - kT/q)^{3/2}] \quad (3.24)$$
$$+ \frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}\frac{kT}{q}[(V_{DS} + 2\phi_F + V_{SB} - kT/q)^{1/2} - (2\phi_F + V_{SB} - kT/q)^{1/2}]\}.$$

The above expression, although it retains the contribution from the diffusion current, is valid in the linear region of operation only, because the surface potential at both ends of the channel is not solved for exactly from expression (3.12).

Beyond a certain point the potential at the drain end of the channel does not increase with an increase in the drain voltage and remains constant, ie the approximation (3.23)

is not valid. This value of the drain voltage is termed the *saturation voltage* (V_{sat}). The classical approach taken in the large geometry MOS transistor theory to determine the saturation voltage is given by [InMo64], [Dang77b], [Sze81]:

$$\frac{dI_{DS}}{dV_{DS}} = 0 \tag{3.25}$$

with $V_{DS} = V_{sat}$. Using (3.24) in (3.25) and solving for V_{DS} yields a rather complicated expression for the saturation voltage [Dang77b] ($V_{SB} = 0$):

$$\begin{aligned} V_{sat} &= \frac{4}{3}\cos^2\left(\theta_{sat} + \frac{2}{3}\pi\right)\left[V_{GS} + \frac{kT}{q} + \frac{1}{3C_{ox}^2}(2\epsilon_s q N_A)\right] \\ &+ \frac{4}{3\sqrt{3}}\cos\left(\theta_{sat} + \frac{2}{3}\pi\right)\frac{1}{C_{ox}}(2\epsilon q N_A)^{1/2}\left[V_{GS} + \frac{kT}{q} + \frac{1}{3C_{ox}^2}(2\epsilon_s q N_A)\right]^{1/2} \\ &+ \frac{1}{9C_{ox}^2}(2\epsilon_s q N_A), \end{aligned} \tag{3.26}$$

where θ_{sat} is found from:

$$\cos(3\theta_{sat}) = \frac{1}{6\sqrt{3}}\frac{1}{C_{ox}}(2\epsilon_s q N_A)^{1/2}\frac{9V_{GS} + \frac{2}{C_{ox}^2}(2\epsilon_s q N_A) - \frac{9}{2}\frac{kT}{q}}{\left[V_{GS} + \frac{kT}{q} + \frac{1}{3C_{ox}^2}(2\epsilon_s q N_A)\right]^{3/2}}. \tag{3.27}$$

The drain current beyond saturation is constant and given by (3.24) with $V_{DS} = V_{sat}$.

3.2.4 Inhantola–Moll Model

Neglecting the terms due to the diffusion current, ie the last line of (3.24) and the term kT/q, gives the standard bulk-charge model first derived by Inhantola-Moll [InMo64]:

$$\begin{aligned} I_{DS} &= \mu\frac{W}{L}C_{ox}\{(V_{GS} - V_{FB} - 2\phi_F - V_{DS}/2)V_{DS} - \\ &\frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}[(V_{DS} + 2\phi_F + V_{SB})^{3/2} - (2\phi_F + V_{SB})^{3/2}]\}, \end{aligned} \tag{3.28}$$

and often referred to as *the text-book* model. It will be now shown how expression (3.28) is derived in a different and more common way [InMo64], [Cobb66], [PaSa66a], [Grov67], [Sze81].

Generally, four principal charge components control the behaviour of the MOS transistor: gate charge (Q_G), channel charge (Q_n), substrate charge due to the immobile ionized dopant (Q_B) and the interface charge (Q_{ss}), all per unit area in the yz plane. These charges are related through the following charge neutrality condition:

$$Q_G + Q_{ss} + Q_B + Q_n = 0. \tag{3.29}$$

Assuming that the electric field in the insulator is normal to the surface (gradual channel approximation) it is possible to apply Gauss theorem across the gate oxide and obtain:

$$Q_G = C_{ox}(V_{GS} - \Phi_{MS} - \psi_S). \tag{3.30}$$

Substituting (3.29) into (3.30) gives an expression relating the gate voltage (V_{GS}) and the channel surface potential (ψ_S):

$$\begin{aligned} V_{GS} &= (\Phi_{MS} - Q_{ss}/C_{ox}) + \psi_S - (Q_B + Q_n)/C_{ox}, \\ &= V_{FB} + \psi_S - \frac{Q_S}{C_{ox}}, \end{aligned} \tag{3.31}$$

which is the same as (3.12).

The large geometry transistor threshold voltage (V_{TL}) is defined as the gate potential required to produce $\psi_S = 2\phi_F$ (ie the strong inversion condition defined by Brown [Brow53]), but Q_n is still neglected compared to all the other charges (both source and the substrate potentials are zero):

$$V_{TL} = V_{FB} + 2\phi_F - Q_B/C_{ox}. \tag{3.32}$$

The depletion region charge, per unit area in the yz plane (Q_B), is given by expression (3.19) with the small contribution from the thermal potential normally neglected:

$$Q_B = -\sqrt{2\epsilon_s q N_A(2\psi_S + V_{SB})}. \tag{3.33}$$

Using expressions (3.31) and (3.33), and approximating the surface potential at inversion by the usual expression, ie $\psi_S = 2\phi_F + V$ (where V is defined, as before, as the reverse bias between point y along the channel and the source electrode), gives the following expression for the mobile channel charge:

$$Q_n = -C_{ox}(V_{GS} - V_{FB} - 2\phi_F - V) + \sqrt{2\epsilon_s q N_A(2\phi_F + V_{SB} + V)}. \tag{3.34}$$

Substituting this into (3.5) and performing the integration yields:

$$\begin{aligned} I_{DS} &= \mu\frac{W}{L}C_{ox}\left\{(V_{GS} - V_{FB} - 2\phi_F)V_{DS} - {V_{DS}}^2/2\right. \\ &- \left.\frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}\left[(V_{DS} + 2\phi_F + V_{SB})^{3/2} - (2\phi_F + V_{SB})^{3/2}\right]\right\}. \end{aligned} \tag{3.35}$$

The saturation voltage V_{sat} is defined, in this case, as the drain voltage that causes the channel charge Q_n to become zero at $y = L$. This is the so called *pinch-off* point beyond which the drain current becomes saturated. The value of the saturation voltage may be obtained from (3.34) by setting $Q_n = 0$ and solving for $V = V_{sat}$:

$$V_{sat} = V_{GS} - V_{FB} - 2\phi_F + \frac{\epsilon_s q N_A}{C_{ox}^2}\left[1 - \sqrt{1 + \frac{2(V_{GS} - V_{FB} + V_{SB})C_{ox}^2}{\epsilon_s q N_A}}\right]. \tag{3.36}$$

Note that the values of the saturation or threshold voltages are not required explicitly in either the Pao-Sah or Brews models because the surface potential along the channel is determined from the field function.

Summary

Of the four large geometry transistor current-voltage models examined in this section, the Pao-Sah model is the most general. This model is valid in all regions of operation: subthreshold, linear and saturation.

The charge sheet model is also valid for all biasing conditions and may be derived from the Pao-Sah model by assuming an approximate expression for Q_n. The value of the surface potential, at both ends of the channel, has to be solved for numerically from expression (3.21) for both models.

By taking an approximate expression for the surface potential at the source and drain ends of the channel, the charge sheet model reduces to that derived by Dang. Finally, the Inhantola-Moll model, which constitutes the text book model, has been rederived here from the Dang model by omitting terms associated with diffusion current. These last two models require additional expressions for the transistor threshold voltage and the saturation voltage and are valid only in the linear and saturation regions of operation.

Large geometry transistor models, such as the Dang and Inhantola-Moll models, are important, since in most cases they constitute a starting point in deriving analytical expressions for the current-voltage characteristics of small geometry MOS transistors.

3.3 Small Geometry MOSFET Models

According to the scaling principle described in [DGYR74] it is possible, in theory, to design a small geometry MOS transistor which exhibits current-voltage characteristics consistent with the large geometry MOSFET theory. The reason is that precise scaling leaves the physics inside the transistor unchanged, although the physical dimensions are scaled down. Precise scaling requires that operation potentials also need to be scaled. Considerations of the noise margin, the necessity for voltage level compatability with other technologies, and the nonscalability of the drain and source junctions built-in potentials, however, are some of the factors that require the supply voltage to be reduced less than precise scaling would indicate [HoMe72], [MaNK79], [Buss82], [Mans82], [PfSM85], [TaJA85]. As a result, with decreasing transistor dimensions many

effects considered to be of second order importance, in developing large geometry device models, must be included in small geometry device models.

Many attempts have been made to go beyond the gradual channel approximation and to develop analytical models which would incorporate the most important physical limitations affecting the behaviour of small devices, and which would be suitable for circuit simulation. This section reviews and compares some of the more commonly used models for the following phenomena, which affect the characteristics of small geometry devices:

1. *Short channel* effect – a decrease of the threshold voltage with a decrease in the channel length;
2. *Narrow width* effect – an increase of the threshold voltage with a decrease in the width of the channel;
3. *Small geometry* effect – the two above effects combined together;
4. *Drain induced barrier lowering* effect – a decrease of the threshold voltage with an increase in the drain bias;
5. *High field effects* – the velocity saturation of the carriers at large parallel electric fields and the carrier mobility degradation effect due to the influence of the normal electric field;
6. *Junction resistances* of the source and drain contacts which negatively affect the transistor characteristics both in the linear region and in saturation; and
7. *Finite output conductance in saturation* for short channel transistors.

3.3.1 Short Channel Effect

Consider the space charge region of a short channel device shown in Figure 3.2. An MOS device is considered to have a short channel when its channel length (L) is of the same order of magnitude as the source and drain depletion region depths, denoted by W_S and W_D, respectively.

The total depletion region charge which contributes to the determination of the threshold voltage (denoted herein the *effective* depletion region charge) is this part of the total space charge which is controlled by the gate. The treatment of a long channel transistor is based on the assumption, that the effective depletion region charge is contained within the uniform volume of depth W_C, width W and length L. All electric field lines

originating on the gate electrode are terminated on the charges in that region. The cross-section of that volume, in the xy plane, is given by the rectangle ADEH in Figure 3.2. In the long channel transistor theory it is implied that the effects of the built-in potentials produced by both the source and drain depletion regions can be neglected.

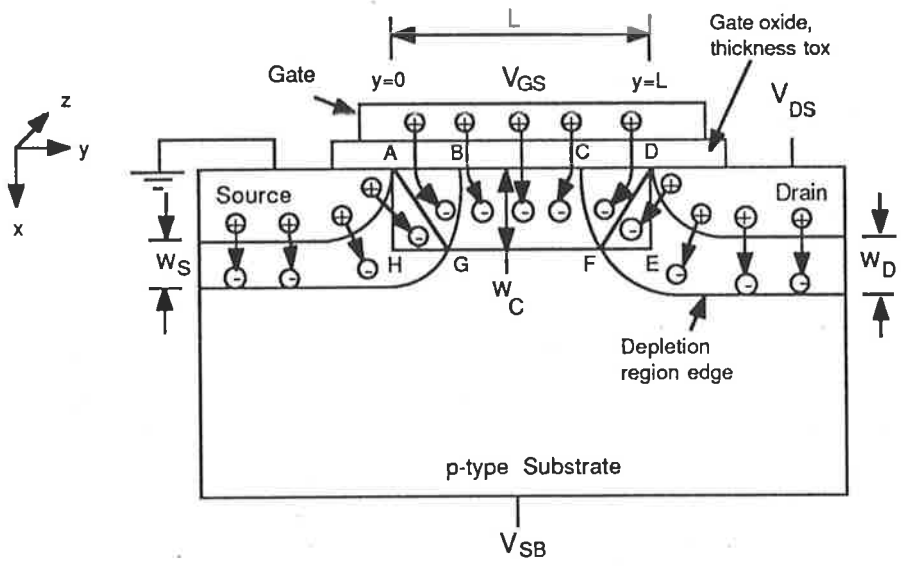


Figure 3.2: Short channel effect geometry model [Yau74].

When the channel length becomes comparable to W_S and W_D, the assumption that the induced charge is a function of the voltage differential between the gate and the substrate only, is no longer valid near the source and drain depletion regions. Because of the interpenetrating source and drain depletion regions, a significantly smaller amount of charge is supported by the gate potential than predicted by the long channel threshold voltage expression, the remainder being supported by the source and the drain potentials, ie the effective depletion region charge for a short channel transistor is smaller than that predicted by the long channel transistor theory. The three most widely used approaches for the modelling of the short channel effect are:

1. The a priori charge sharing assumption;
2. The a priori surface potential distribution; and
3. Empirical techniques.

Both the first two methods assume that charge sharing is taking place and seek to obtain an expression for the portion of the depletion region charge supported by the gate charge. The charge conservation principle, applied to an appropriately drawn Gaussian surface including the gate electrode, is used to obtain a closed form expression for the threshold

voltage. The approaches differ in that in the first method a simple assumption about a uniform potential at inversion is made, and this assumption is supported by a plausible geometric assumption about the portion of the depletion region charge supported by each of the source, gate and drain electrodes. The second method recognizes charge sharing but does not make an a priori assumption about its magnitude. Instead it seeks to obtain a more accurate approximation for the distribution of potential along the channel surface as a result of different doping concentrations in the source, channel and drain regions. The charge sharing is then derivable from this potential distribution.

The empirical models are derived to provide satisfactory representation of experimental or two-dimensional simulation results with the functional form of the model being sometimes guided by the analysis described above.

The models described in the remainder of this section are grouped into the three categories described above.

3.3.1.1 A priori Charge Sharing

The approach based on the a priori charge sharing assumption involves a calculation of the effective depletion charge (Q_{BS}), ie the total depletion region charge which is supported by the gate charge only. The diagram of Figure 3.2 gives a simple model developed by Poon *et al.* [PYJB73] and Yau [Yau74], and based on geometrical considerations of the depletion charge. According to this model, the effective depletion region charge is contained within a uniform volume (in the z direction) whose cross-section in the xy plane is given by the trapezium ADFG. The field lines originating from the charges inside this trapezoidal region are terminated on the charges on the gate electrode, while the field lines originating from the charges outside this region are terminated at the source and drain regions.

The trapezoidal bulk charge causes, however, a nonuniform surface potential along the channel. Therefore, a simplifying approximation is made, that the effect of the bulk charge is uniform at threshold, and the surface potential is constant along the channel. The effective depletion region charge is then obtained by subtracting the regions of charge, represented in the xy plane by triangles AGH and DEF, associated with the source and drain depletion regions, respectively:

$$\begin{aligned} Q_{BS} &= -qN_A\left(WLW_C - WA_S - WA_D\right), \\ &= -\left(1 - \frac{A_S + A_D}{W_C L}\right) qN_A W_C WL, \\ &= \gamma_S Q_B WL, \end{aligned} \tag{3.37}$$

where γ_S is the dimensionless geometry factor given by the term enclosed in brackets and representing the short channel effect, Q_B is the depletion charge per unit area in the yz plane given by (3.33), W_C is the depletion region depth under the gate and approximately given by:

$$W_C = \left[\frac{2\epsilon_s}{qN_A}(2\phi_F + V_{SB})\right]^{1/2}, \tag{3.38}$$

A_S is the area of triangle AGH and A_D is the area of triangle CDF.

The effective threshold voltage (V_{TS}) of a short channel transistor is defined, using the concept of the effective depletion region charge, by:

$$\begin{aligned} V_{TS} &= V_{FB} + 2\phi_F - \frac{Q_{BS}}{WLC_{ox}}, \\ &= V_{FB} + 2\phi_F - \gamma_S \frac{Q_B}{C_{ox}}, \\ &= V_{TL} + \Delta V_{TS}, \end{aligned} \tag{3.39}$$

where V_{TL} is the long channel transistor threshold voltage given by (3.32) and ΔV_{TS} is a negative quantity representing the short channel effect. The expressions developed by various workers using this approach basically differ in the formulations used for the estimation of the two regions of charge, A_S and A_D.

One of the first models, based on the a priori charge sharing assumption was developed by Varshney [Vars73]. The geometry used in that analysis is shown in Figure 3.3. The

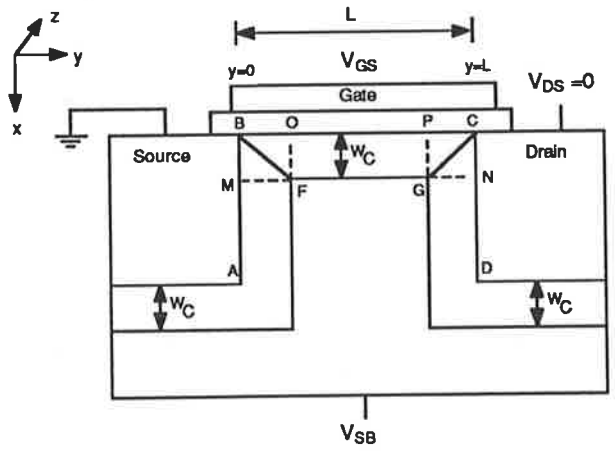


Figure 3.3: Cross-section view of the model used by Varshney [Vars74].

author assumes that, under the conditions of no source or drain bias, the charge in the region bounded by FMBO is equally supported by the source and the gate fields, while the charge contained within the area bounded by GPCN is equally supported by the gate and the drain fields. It is also assumed that the potential difference between

the points F and M is approximately equal to the potential difference between points F and O. This is a reasonable assumption since, at strong inversion, the source and drain junction built-in potential (V_{bi}) is to a first approximation equal to the commonly assumed value ($2\phi_F$) of the channel surface potential. Thus, it can be assumed that the source junction depletion region depth $W_S \approx W_C$. A similar analysis can be applied to the drain junction depletion region depth; $W_D \approx W_C$. The effective depletion region charge is bounded by the uniform volume in the z direction, whose cross-section in the xy planc is given by the trapezium BFGC:

$$\begin{aligned} Q_{BS} &= -qN_A\left(WLW_C - 2W\frac{W_C W_C}{2}\right), \\ &= -qN_A W_C\left(1-\frac{W_C}{L}\right)WL. \end{aligned} \tag{3.40}$$

The geometry factor according to [Vars73] is given by:

$$\gamma_S = 1 - \frac{W_C}{L} \tag{3.41}$$

and the short channel effect is given by:

$$\Delta V_{TS} = \frac{Q_B}{C_{ox}}\frac{W_C}{L}. \tag{3.42}$$

This model, though very simple, provides a physical insight into the charge sharing concept. In addition, it correctly predicts the departure, as a function of the channel length and reversed substrate bias, from the long channel transistor threshold voltage expression. However, it does not include the effects of the drain voltage and the drain and source junction depths.

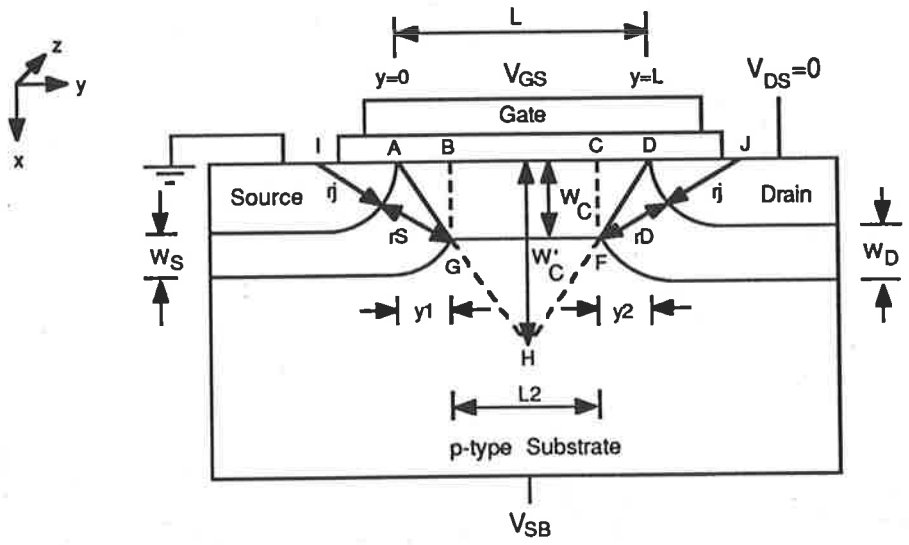


Figure 3.4: Cross-section view of the model used by Yau [Yau74].

A model based on Varshney's analysis but which is a more accurate representation of the short channel effect was developed by Yau [Yau74]. This model is important because

it became a basis for later models derived by other authors. The geometry used by Yau is depicted in Figure 3.4. The effective depletion region charge is enclosed in the xy plane by the trapezium ADFG, and is given by:

$$Q_{BS} = -qN_A W_C \left(\frac{L + L_2}{2L}\right) WL. \tag{3.43}$$

The length $L_2 = L - 2y_1$ (where it is assumed that $V_{DS} = 0$, from which it follows that $y_1 = y_2$) is expressed in terms of known distances by considering the triangle IBG (or CJF) in which it is assumed that $r_S = r_D = W_C$:

$$(y_1 + r_j)^2 + {W_C}^2 = (W_C + r_j)^2, \tag{3.44}$$

from which by straightforward trigonometric analysis:

$$\frac{L + L_2}{2L} = 1 - \frac{y_1}{L} = 1 - \frac{r_j}{L}\left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right]. \tag{3.45}$$

Employing (3.43) and (3.45) Yau derives the following expression for the short channel effect:

$$\Delta V_{TS} = \frac{Q_B}{C_{ox}}\frac{r_j}{L}\left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right], \tag{3.46}$$

with the short channel effect geometry factor given by:

$$\gamma_S = 1 - \frac{r_j}{L}\left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right]. \tag{3.47}$$

Equation (3.46) is modified for the case where the substrate bias voltage is sufficiently high ($V_{SB} > V'_{SB}$) to cause the source and drain depletion regions to meet. Under such conditions, it is assumed in [Yau74] that the effective depletion charge is enclosed, in the xy plane, by the triangle ADH. The substrate voltage causing this condition is obtained from (3.45):

$$W'_C = W_C(L_2 \to 0) = \frac{L}{2r_j}\left(r_j + \frac{L}{4}\right). \tag{3.48}$$

The short channel effect for $V_{SB} > V'_{SB}$ becomes:

$$\Delta V_{TS} = \frac{qN_A}{2C_{ox}}\left[W'_C - 2\frac{1}{4(1 - 2r_j/L)}(W_C - W'_C)\right]. \tag{3.49}$$

This model explicitly and correctly describes the dependence of the threshold voltage on the source and drain junction depths and the channel length. In the limit $W_C \ll r_j$, Yau's expression, (3.46), reduces to that of Varshney, (3.42). This analysis, however, assumes the depletion region to be symmetrical with respect to source and drain, and consequently the final expression does not incorporate the influence of the drain bias, ie the drain induced barrier lowering effect.

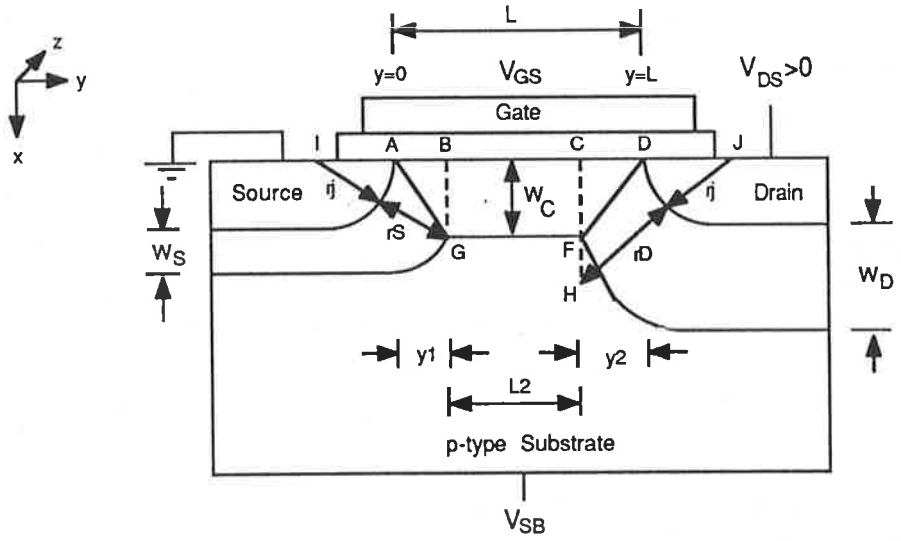


Figure 3.5: Cross-section view of the model used in [YaMo83].

Yamaguchi and Morimoto [YaMo83] extended Yau's analysis and proposed an expression that describes the effect of the drain voltage on the threshold voltage. The effect of the drain voltage is included by considering a general trapezoid AGHD in which $y_2 > y_1$, as shown in Figure 3.5. Using the abrupt plane pn junction approximation the various depletion region widths in Figure 3.5 are estimated as follows:

$$r_S = W_S = \left[\frac{2\epsilon_s}{qN_A}(2\phi_F + V_{SB})\right]^{1/2}, \tag{3.50}$$

$$r_D = W_D = \left[\frac{2\epsilon_s}{qN_A}(2\phi_F + V_{SB} + V_{DS})\right]^{1/2}. \tag{3.51}$$

The two lengths, y_1 and y_2, are found from triangles IBG and JHC, respectively (with the length CH= W_D):

$$y_1 = r_j\left[\left(1 + \frac{2W_S}{r_j}\right)^{1/2} - 1\right], \tag{3.52}$$

$$y_2 = r_j\left[\left(1 + \frac{2W_D}{r_j}\right)^{1/2} - 1\right]. \tag{3.53}$$

The effective depletion region charge is found by substituting $L_2 = L - y_1 - y_2$ into expression (3.43):

$$\begin{aligned} Q_{BS} &= -qN_A W_C\left(1 - \frac{y_1}{2L} - \frac{y_2}{2L}\right)WL, \\ &= Q_B\gamma_S WL. \end{aligned} \tag{3.54}$$

Using the above expression together with expressions (3.52) and (3.53) gives the short channel effect according to [YaMo83]:

$$\Delta V_{TS} = 0.5\frac{Q_B}{C_{ox}}\frac{r_j}{L}\left\{\left[\left(1 + \frac{2W_S}{r_j}\right)^{1/2} - 1\right] + \left[\left(1 + \frac{2W_D}{r_j}\right)^{1/2} - 1\right]\right\}, \tag{3.55}$$

with the short channel effect geometry factor given by:

$$\gamma_S = 1 - 0.5\frac{r_j}{L}\left\{\left[\left(1+\frac{2W_S}{r_j}\right)^{1/2} - 1\right] + \left[\left(1+\frac{2W_D}{r_j}\right)^{1/2} - 1\right]\right\}. \tag{3.56}$$

As expected, under the conditions of no drain to source voltage (W_D=W_S) the above expression reduces to Yau's model given by (3.46).

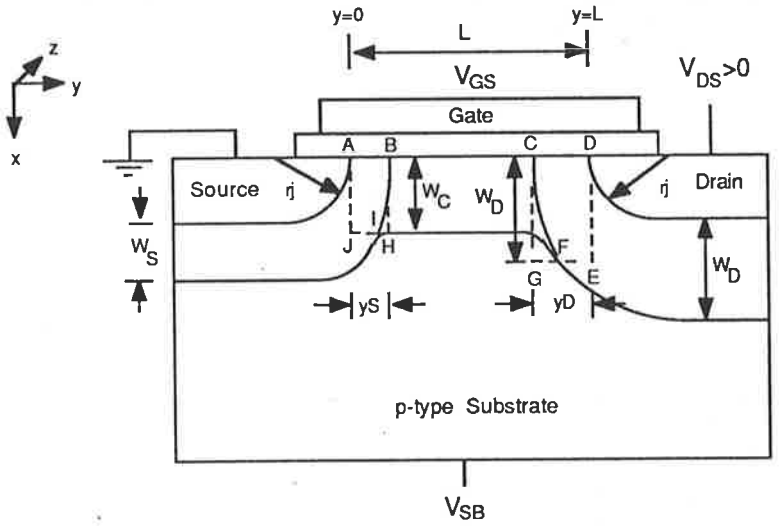


Figure 3.6: Cross-section view of the model used by Jantsch [Jant82].

In the short channel threshold voltage model proposed by Jantsch [Jant82] the source and drain depletion regions are approximated by parts of ellipses, as shown in Figure 3.6. The ellipse for the source depletion region boundary has a major axis given by $2(r_j+W_S)$ and a minor axis given by $2(r_j + y_S)$, and the ellipse for the drain depletion region boundary has a major axis given by $2(r_j + W_D)$ and a minor axis given by $2(r_j + y_D)$. The ratio of the areas of the source depletion region defined in this way and the rectangle ABHJ (or the ratio of the areas of the drain depletion region and the rectangle CDEG) is a function of r_j and N_A and ranges in value from 0.6 for $r_j = 0.1 \times 10^{-6}m$ and $N_A = 10^{21}m^{-3}$ to 0.8 for $r_j = 0.5 \times 10^{-6}m$ and $N_A = 10^{22}m^{-3}$, thus a value of 0.7 is proposed in [Jant82]. The total depletion region charge supported by the gate charge is then given by:

$$Q_{BS} = -0.7qN_AW(y_SW_S + y_DW_D), \tag{3.57}$$

where W_S and W_D are given by expressions (3.50) and (3.51), respectively, and where y_S is the source depletion region width at the surface of the channel and y_D is the drain depletion region width at the surface of the channel. In [Jant82] it is assumed that $y_S = y_1$ and $y_D = y_2$, with y_1 and y_2 given by (3.52) and (3.53), respectively. The short

channel effect according to Jantsch is then given by:

$$\Delta V_{TS} = 0.7\frac{Q_B}{C_{ox}}\frac{r_j}{L}\left\{\left[\left(1+\frac{2W_S}{r_j}\right)^{1/2}-1\right]+\frac{W_D}{W_S}\left[\left(1+\frac{2W_D}{r_j}\right)^{1/2}-1\right]\right\}. \qquad (3.58)$$

Comparing expressions (3.58) and (3.55), given by [YaMo83], it is found that under the conditions of no drain bias the two differ only by a constant multiplying factor $0.7/0.5 = 1.4$. In addition expression (3.58) predicts a larger decrease in the threshold voltage with an increase in the drain voltage than expression (3.55).

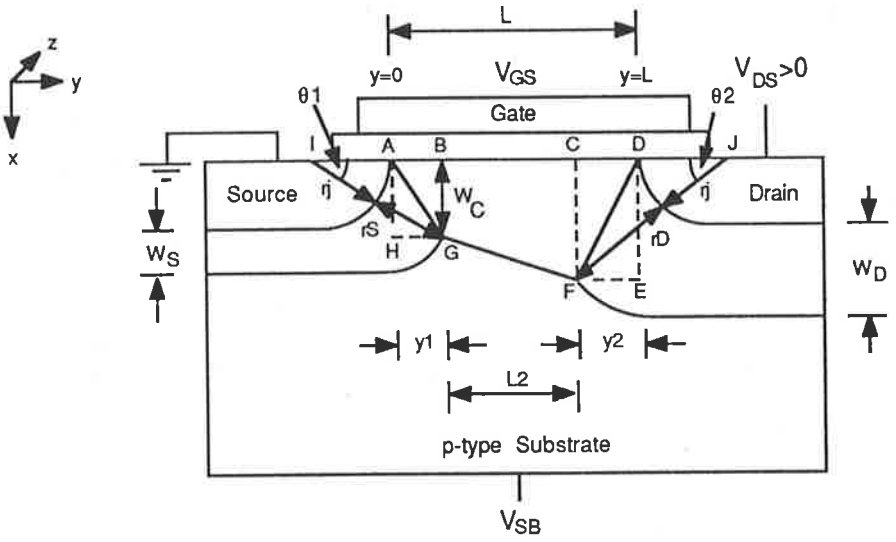


Figure 3.7: Cross-section view of the model used by Silburt *et al.* [SIFP84].

Silburt *et al.* [SiFP84] considered the geometry shown in Figure 3.7. The regions containing the depletion charge supported by the source and drain electric fields are approximated in the xy plane by triangles AGH and DEF, respectively. The following assumptions about the various distances are made: $r_S = BG = W_C = W_S$ and $r_D = FC = W_D$, with W_S and W_D given by (3.50) and (3.51), respectively. The resulting expression, (3.59) below, for the short channel effect is the same as (3.58) given by [Jant82], with the constant factor of 0.7 replaced by 0.5:

$$\Delta V_{TS} = -0.5\frac{qN_A}{C_{ox}}\frac{r_j}{L}\left\{W_S\left[\left(1+\frac{2W_S}{r_j}\right)^{1/2}-1\right]+W_D\left[\left(1+\frac{2W_D}{r_j}\right)^{1/2}-1\right]\right\}. \qquad (3.59)$$

Expression (3.59) was subsequently simplified to eliminate square root calculations by assuming that, over the bias range in which the device is used, the angles θ_1 and θ_2, defined in Figure 3.7, do not change significantly with drain and substrate voltages. Taking $\tan\theta_1$ and $\tan\theta_2$ given by:

$$\tan\theta_1 = \frac{1}{\frac{r_j}{W_S}\left[\left(1+\frac{2W_S}{r_j}\right)^{1/2}\right]}, \qquad \tan\theta_2 = \frac{1}{\frac{r_j}{W_D}\left[\left(1+\frac{2W_D}{r_j}\right)^{1/2}\right]}, \qquad (3.60)$$

to be constant and substituting these into expression (3.59) yields:

$$\Delta V_{TS} = -0.5\frac{qN_A}{C_{ox}}\frac{1}{L}\left(\frac{W_S^2}{\tan\theta_1} + \frac{W_D^2}{\tan\theta_2}\right). \tag{3.61}$$

Substituting for W_S and W_D from (3.50) and (3.51) gives:

$$\begin{aligned}\Delta V_{TS} &= -\frac{\epsilon_s}{C_{ox}L}\left(\frac{V_{SB}+2\phi_F}{\tan\theta_1} + \frac{V_{DS}+V_{SB}+2\phi_F}{\tan\theta_2}\right), \\ &= -\frac{K_S}{L}(2\phi_F + V_{SB}) - \frac{K_D}{L}V_{DS}, \end{aligned} \tag{3.62}$$

where,

$$K_S = \frac{\epsilon_s}{C_{ox}}\left(\frac{1}{\tan\theta_1} + \frac{1}{\tan\theta_2}\right), \qquad K_D = \frac{\epsilon_s}{C_{ox}}\left(\frac{1}{\tan\theta_2}\right). \tag{3.63}$$

In the resulting expression, the short channel effect is linearly dependent on the drain and substrate biases. No experimental data for the threshold voltage has been presented in [SiFP84] to demonstrate the validity of this analysis. The results obtained by Troutman and Fortino [TrFo77], Klassen and De Groot [KlGr80], and Ratnakumar and Meindl [RaMe82] indicate that these linear voltage dependencies are more accurate than the other models currently in use. The complete model is simple, incorporates all the important physical effects and is well suited for circuit simulation purposes. Furthermore, by treating K_N and K_D as empirical parameters expression (3.62) should, theoretically, represent the short channel effect over a limited range of bias voltages with sufficient accuracy for circuit simulation purposes.

Fichtner and Pötzl [FiPo79] suggested that the approximations (3.50) and (3.51) used to determine the lengths y_1 and y_2 in Figure 3.5 result in expressions (3.52) and (3.53) overestimating y_1 and y_2, respectively. The authors assume instead, that the source and drain depletion regions do not extend as far into the bulk at the surface of the channel as they extend into the bulk beneath the source and drain junctions. Consequently, it is proposed that W_S and W_D be approximated in expression (3.55) by the following expressions corresponding to the depletion region widths at the surface of the channel (Figure 3.6) and given by the abrupt plane junction approximation:

$$W_S = y_S = \left[\frac{2\epsilon_s}{qN_A}(V_{bi} - 2\phi_F)\right]^{1/2}, \tag{3.64}$$

$$W_D = y_D = \left[\frac{2\epsilon_s}{qN_A}(V_{bi} - 2\phi_F + V_{DS})\right]^{1/2}. \tag{3.65}$$

The quantity V_{bi} is the built-in potential of the source and drain junctions, and is normally treated as an empirical parameter. This model was found to accurately represent the threshold voltage results, at low drain voltage, presented in Chapter 4.

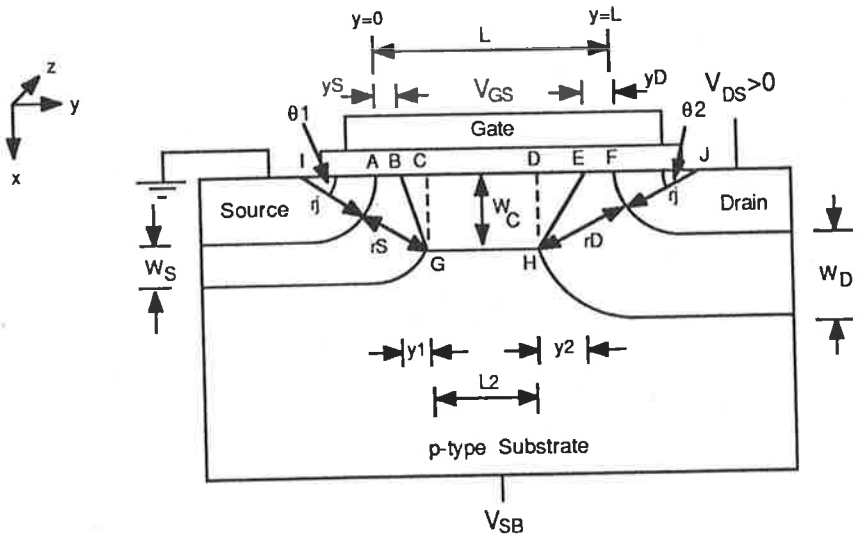


Figure 3.8: Cross-section view of the model used by Taylor [Tayl78].

Taylor [Tayl78] generalized Yau's approach, which assumed $y_S = y_D = 0$ (Figure 3.8). The analysis given in [Tayl78] and also later in [Tayl79] is rather sketchy and in the following it is attempted to elucidate the process leading to the final model. The various depletion region widths are also approximated by using the abrupt plane pn junction approximation:

$$r_S = W_S = \left[\frac{2\epsilon_s}{qN_A}(V_{bi} + V_{SB})\right]^{1/2}, \tag{3.66}$$

$$r_D = W_D = \left[\frac{2\epsilon_s}{qN_A}(V_{bi} + V_{SB} + V_{DS})\right]^{1/2} \tag{3.67}$$

and y_S and y_D are given by (3.64) and (3.65), respectively. The quantities y_1 and y_2 are calculated by geometric analysis, as in Yau's analysis. The length y_1 is given by considering triangle ICG:

$$y_1 = (r_j + r_S)\cos\theta_1 - r_j - y_S. \tag{3.68}$$

Using the trigonometric identity $\cos\theta_1 = \sqrt{1 - \sin^2\theta_1}$ and the corresponding expressions for $\sin\theta_1$, $\cos\theta_1$, r_S and y_S, expression (3.68) can be given in terms of r_j and the various bias voltages:

$$\begin{aligned}\cos\theta_1 &= \left\{1 - \left[\frac{\left[\frac{2\epsilon_s}{qN_A}(2\phi_F + V_{SB})\right]^{1/2}}{r_j + \left[\frac{2\epsilon_s}{qN_A}(V_{bi} + V_{SB})\right]^{1/2}}\right]^2\right\}^{1/2}, \\ &= \left\{\frac{\left[r_j + \left[\frac{2\epsilon_s}{qN_A}(V_{bi} + V_{SB})\right]^{1/2}\right]^2 - \frac{2\epsilon_s}{qN_A}(2\phi_F + V_{SB})}{\left[r_j + \left[\frac{2\epsilon_s}{qN_A}(V_{bi} + V_{SB})\right]^{1/2}\right]^2}\right\}^{1/2}.\end{aligned} \tag{3.69}$$

Substituting expression (3.69) into expression (3.68) gives:

$$y_1 = \left\{\frac{2\epsilon_s}{qN_A}(V_{bi}-2\phi_F)+2r_j\left[\frac{2\epsilon_s}{qN_A}(V_{bi}+V_{SB})\right]^{1/2}+{r_j}^2\right\}^{1/2} - r_j - \left[\frac{2\epsilon_s}{qN_A}(V_{bi}-2\phi_F)\right]^{1/2}. \quad (3.70)$$

A similar analysis based on triangle JHD yields y_2:

$$y_2 = \left\{\frac{2\epsilon_s}{qN_A}(V_{bi}+V_{DS}-2\phi_F)+2r_j\left[\frac{2\epsilon_s}{qN_A}(V_{bi}+V_{SB})\right]^{1/2}+{r_j}^2\right\}^{1/2} - r_j - \left[\frac{2\epsilon_s}{qN_A}(V_{bi}+V_{DS}-2\phi_F)\right]^{1/2}. \quad (3.71)$$

Taylor assumed that no gate induced depletion charge exists in the source and drain depletion regions and the effective depletion region charge is bounded, in the xy plane, by the trapezium BEGH and is calculated by:

$$Q_{BS} = -qN_A W_C \frac{2L_2 - y_1 - y_2}{2(L_2+2f_\delta)} WL, \quad (3.72)$$

where f_δ is introduced to account for the effects of fringing at the overlap of the source and drain and is defined by:

$$f_\delta = t_{ox} \ln\left(\frac{eL_2}{t_{ox}}\right), \quad (3.73)$$

where e is the base of the natural logarithm. The short channel effect according to Taylor is given by:

$$\Delta V_{TS} = \frac{Q_B}{C_{ox}}\left[\frac{2(L-y_S-y_D)-y_1-y_2}{2(L-y_S-y_D+2f_\delta)}\right]. \quad (3.74)$$

For an MOS device with geometry such that $r_j \ll \left[\frac{2\epsilon_s}{qN_A}(2\phi_F+V_{SB})\right]^{1/2}$ the above expression reduces to:

$$\Delta V_{TS} = \frac{Q_B}{2C_{ox}}\left[\frac{W_D-y_D+W_S-y_S}{L-y_D-y_S}\right]. \quad (3.75)$$

It can be shown, with a further algebraic manipulation, that (3.75) reduces to (3.46), ie the expression given by Yau [Yau74], under the conditions of no drain bias and if it is assumed that $V_{bi} = 2\phi_F$.

Dang [Dang79] proposed a model which can be related to the models given by Yau[Yau74] and Jantsch [Jant82]. Dang incorporated into the Yau's original theory the formula for unsymmetrical cylindrical junction and assumed, as was later also done by Jantsch, that the boundary of the space charge shared by source or drain takes the form of an ellipse. Figure 3.9 depicts the geometrical approximation used in Dang's analysis. The major axis of the ellipse approximating the boundary of the source depletion region

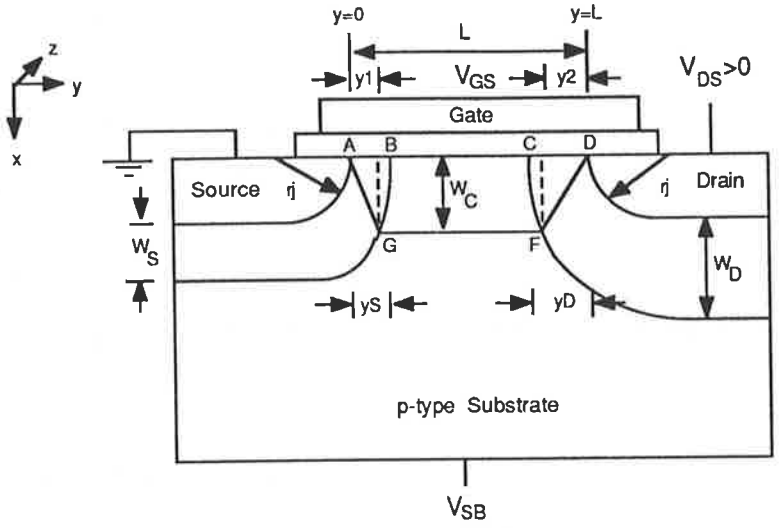


Figure 3.9: Cross-section view of the model used by Dang [Dang79].

is taken to be $2(r_j + W_S)$ and the minor axis of the ellipse is taken to be $2(0.7r_j + y_S)$, where y_S is given by the radial component of the *cylindrical* abrupt *pn* junction approximation, expression (3.76) below, and W_S is given by (3.66). Similarly, the major axis of the ellipse approximating the boundary of the drain depletion region is taken to be $2(r_j + W_D)$ and the minor axis of the ellipse is taken to be $2(0.7r_j + y_D)$, where y_D is given by the radial component of the *cylindrical* abrupt *pn* junction approximation, expression (3.77) below, and W_D is given by (3.67).

$$y_{S,p}/r_j = (1 + y_S/r_j)\left[\ln(1 + y_S/r_j) - 0.5 + 0.5(1 + y_S/r_j)^{-2}\right]^{1/2}, \quad (3.76)$$

$$y_{D,p}/r_j = (1 + y_D/r_j)\left[\ln(1 + y_D/r_j) - 0.5 + 0.5(1 + y_D/r_j)^{-2}\right]^{1/2}. \quad (3.77)$$

The above expressions must be solved for y_S and y_D numerically, where $y_{S,p}$ and $y_{D,p}$ are the corresponding quantities given from the *plane* abrupt *pn* junction approximation:

$$y_{S,p} = \left[\frac{2\epsilon_s}{qN_A}(2\phi_F + V_{SB})\right]^{1/2}, \quad (3.78)$$

$$y_{D,p} = \left[\frac{2\epsilon_s}{qN_A}(2\phi_F + V_{SB} + V_{DS})\right]^{1/2}. \quad (3.79)$$

Dang [Dang79] assumed, in line with Yau's model, that the effective bulk depletion charge is confined in the xy plane to the trapezoid ADFG, Figure 3.9. Then from geometry:

$$y_1 = (y_S + 0.7r_j)\left[1 - {W_C}^2/(W_S + r_j)^2\right]^{1/2} - 0.7r_j, \quad (3.80)$$

$$y_2 = (y_D + 0.7r_j)\left[1 - {W_C}^2/(W_D + r_j)^2\right]^{1/2} - 0.7r_j. \quad (3.81)$$

The short channel effect according to Dang is given by:

$$\Delta V_{TS} = -0.5\frac{qN_AW_C}{C_{ox}}\left(\frac{y_1+y_2}{L}\right), \tag{3.82}$$

$$= 0.5\frac{Q_B}{C_{ox}}\left(\frac{y_1+y_2}{L}\right). \tag{3.83}$$

The main disadvantage with this model is that the expressions (3.76) and (3.77) must be solved using numerical techniques. Dang [Dang79] proposed to approximately solve (4.8) using a second degree polynomial:

$$y_S \approx 0.063151 + 0.80113292\left(\frac{W_S}{r_j}\right) - 0.0111777\left(\frac{W_S}{r_j}\right)^2, \tag{3.84}$$

with an analogous expression for y_D. The error in using approximation (3.84) is less than 5% if $y_S/r_j > 0.3$, ie for transistors with shallow junctions or for cases where a large bias is applied across the junction. This approximation is used in the short channel effect model used in SPICE level 3 [Nage82] analysis.

Examination of expression (3.78) shows that the source depletion region width (y_S) at the channel surface is grossly overestimated, especially under the conditions of applied substrate bias. This is because the potential across the source-substrate junction is taken to be $2\phi_F + V_{SB}$. However, this junction potential is independent of the substrate bias under the condition of strong inversion. As suggested by Fichtner and Pötzl [FiPo79] and Taylor [Tayl78] the voltage drop, assuming strong inversion, across the source-substrate junction at the surface of the channel is better approximated by $V_{bi} - 2\phi_F$. A similar comment is valid for the drain-substrate junction where the corresponding voltage drop is better approximated by $V_{bi} - 2\phi_F + V_{DS}$ instead of $V_{DS} + 2\phi_F + V_{SB}$, which was used in [Dang79]. Thus, this model may be expected to overestimate the short channel effect, the drain induced barrier lowering effect and the effect of the substrate voltage on the threshold voltage. This is indeed the case for the results presented in Chapter 4.

3.3.1.2 A priori Surface Potential Distribution

The second main approach used in the modelling of the short channel effect is based on an approximation of the surface potential along the channel.

One of the first models to account for the variation of the surface potential along the channel was proposed by Lee [Lee73]. The starting point in Lee's analysis is an expression relating the applied gate voltage (V_{GS}), the surface potential (ψ_S) and the voltage across the oxide (V_{ox}) which is obtained with the use of the Fermi level diagram shown in Figure 3.10:

$$V_{ox}(y) = -\Phi_{MS} - \psi_S(y) + V_{SB} + V_{GS}, \tag{3.85}$$

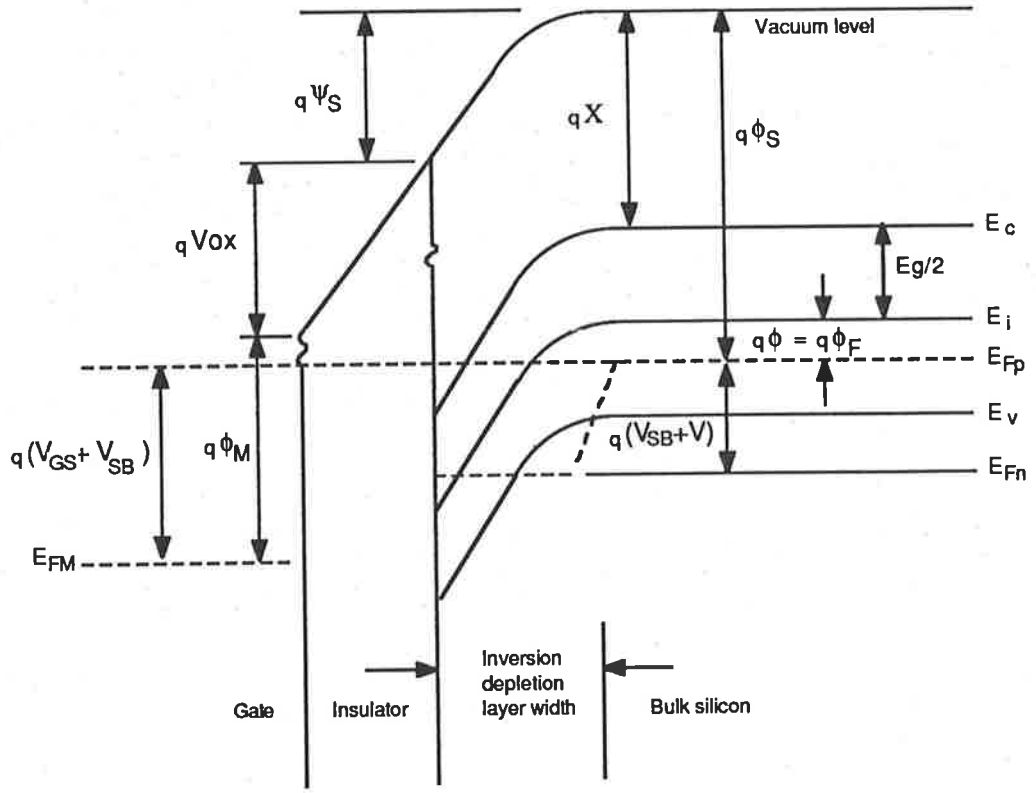


Figure 3.10: Cross-sectional energy band diagram for an n-channel MOS transistor (after Bandy and Kokalis [BaKo77]).

where

$$\Phi_{MS} = \Phi_M - (X + E_G/2 + \phi_F), \tag{3.86}$$

and

$$\psi_S(y) = V_{SB} + V(y) + B(y)\phi_F. \tag{3.87}$$

In Figure 3.10, E_i is the intrinsic Fermi level, E_c is the conduction band Fermi level, E_v is the valence band Fermi level, X is the semiconductor electron affinity, Φ_M is the work function of the gate material and E_G is the energy gap between the valence and conduction bands. The quantity $B(y)$, referred to as the Brown factor, designates the strength of inversion at a point y along the channel. At the onset of inversion the silicon surface is intrinsic and $B(y) = 1$, while for the condition of strongly inverted surface $B(y) = 2$. The quantity $V(y)$, as defined earlier, accounts for the split in the quasi-Fermi level due to the drain voltage, while it is assumed that the split in the quasi-Fermi level due to the substrate bias is uniform across the channel. In addition to (3.85)-(3.87) the charge conservation principle given by:

$$Q_{GT} + WLQ_{ss} + Q_{BS} = 0 \tag{3.88}$$

must also be satisfied, where Q_{GT} represents the total charge on the gate and it is assumed that at threshold the mobile channel charge is negligible compared to the

remaining charges. The total charge on the gate is found from the following expression:

$$Q_{GT} = WC_{ox}\int_0^L V_{ox}(y)dy, \tag{3.89}$$

and the effective depletion region charge is found from:

$$Q_{BS} = -W\int_0^L \int_{\psi_1(y)}^{\psi_2(y)} C_{si}(y)d\psi dy, \tag{3.90}$$

where $C_{si} = (dQ_{BS}/dy)/(WL)$ denotes the space charge differential capacitance [Lee73] per unit area in the yz plane at a point y along the channel. The quantity ψ_1 is the surface potential present in the device when no bias in applied and the quantity ψ_2 is the surface potential present in the device at threshold. Expression (3.88) can be written in the following form using expressions (3.89) and (3.90):

$$WC_{ox}\int_0^L V_{ox}(y)dy + WLQ_{ss} - W\int_0^L \int_{\psi_1(y)}^{\psi_2(y)} C_{si}(y)d\psi(y)dy = 0. \tag{3.91}$$

Substituting expression (3.85) into (3.91) and solving for the gate voltage yields the required equation relating V_{GS} and ψ_S:

$$V_{GS} = \Phi_{MS} - \frac{Q_{ss}}{C_{ox}} + \frac{1}{L}\int_0^L [B(y)\phi_F + V(y)]\,dy + \frac{1}{C_{ox}L}\int_0^L \int_{\psi_1(y)}^{\psi_2(y)} C_{si}(y)d\psi(y)dy. \tag{3.92}$$

Expression (3.92) conveys the fact that the quantities $V_{ox}(y)$, $B(y)$ and $V(y)$ must adjust among themselves such that the gate is maintained at an equipotential V_{GS} and the overall charge neutrality condition is satisfied.

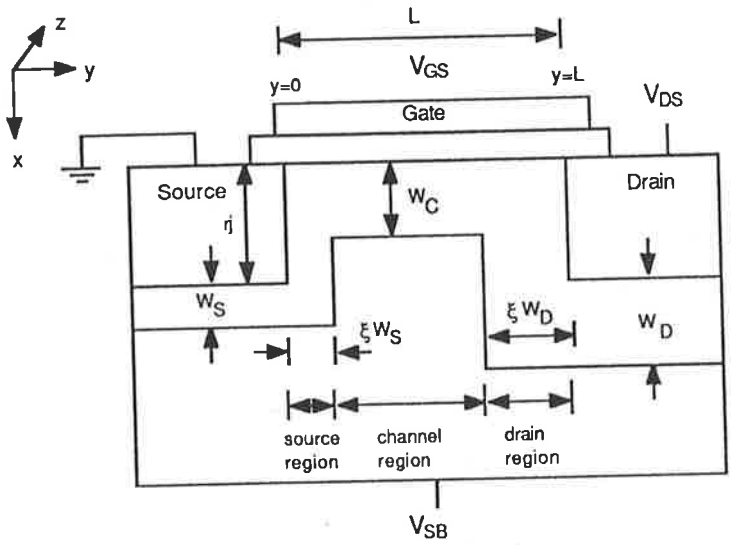


Figure 3.11: Model used by Lee [Lee73].

In order to evaluate the second integral in expression (3.92) a regional approximation of the source, the channel and the drain depletion charge regions is made as shown in

Figure 3.11. Using this approximation, the space charge differential capacitance per unit width is constant in each region and given by:

$$\int_0^L C_{si}(y)dy = \begin{cases} \xi W_S \frac{\epsilon_s}{W_S+r_j} & \text{source region,} \\ [L-\xi(W_D+W_S)]\frac{\epsilon_s}{W_C} & \text{channel region,} \\ \xi W_D \frac{\epsilon_s}{W_D+r_j} & \text{drain region,} \end{cases} \tag{3.93}$$

where the various depletion region dimensions are given from the abrupt plane pn junction theory:

$$W_C = \left[\frac{2\epsilon_s}{qN_A}(B\phi_F + V_{SB})\right]^{1/2}, \tag{3.94}$$

$$W_S = \left[\frac{2\epsilon_s}{qN_A}(V_{bi} + V_{SB})\right]^{1/2}, \tag{3.95}$$

$$W_D = \left[\frac{2\epsilon_s}{qN_A}(V_{bi} + V_{SB} + \xi_1 V_{DS})\right]^{1/2}. \tag{3.96}$$

The constant ξ is a weighing factor that determines the average net effect of the source and drain depletion regions extensions into the gate region and the constant ξ_1 represents the net average effect of the drain voltage on the depth of the drain depletion region. Both factors have to be determined from experimental data.

According to Lee's model the potentials ψ_1 and ψ_2 in each region are constant and given by:

source region	channel region	drain region
$\psi_1 = V_{bi}$	$\psi_1 = 0$	$\psi_1 = \xi_1 V_{DS} + V_{bi}$
$\psi_2 = V_{bi} + V_{SB}$	$\psi_2 = B(y)\phi_F + V_{SB}$	$\psi_2 = \xi_1 V_{DS} + V_{bi} + V_{SB}$

(3.97)

Employing the above approximations the second integral of expression (3.92) can be evaluated over the various regions, as follows:

$$\begin{aligned} \frac{1}{LC_{ox}}\int_o^L \int_{\psi_1(y)}^{\psi_2(y)} C_{si}(y)d\psi(y)dy &= \frac{1}{LC_{ox}}\left\{\int_{V_{bi}}^{V_{bi}+V_{SB}} \xi W_S \frac{\epsilon_s}{W_S+r_j}d\psi \right. \\ &+ \int_0^{B(y)\phi_F+V_{SB}} [L-\xi(W_D+W_S)]\frac{\epsilon_s}{W_C}d\psi \\ &+ \left.\int_{V_{bi}+\xi_1 V_{DS}}^{V_{bi}+V_{SB}+\xi_1 V_{DS}} \xi W_D \frac{\epsilon_s}{W_D+r_j}d\psi\right\}. \end{aligned} \tag{3.98}$$

The first integral in (3.92) is evaluated with the aid of two simplifying assumptions. The first assumption is that the quantity $V(y)+B(y)\phi_F$ remains constant in the subthreshold region as discussed in [Barr72], and the second assumption is that the condition $B(y)\phi_F > (\xi W_D V_{DS}/L)$ applies. The first integral in expression (3.92) then reduces to:

$$\frac{1}{L}\int_0^L [B(y)\phi_F + V(y)]\,dy = B\phi_F. \tag{3.99}$$

The final expression for the short channel effect is complex and reads:

$$\Delta V_{TS} = -\frac{\xi}{LC_{ox}}\left(A_0 + A_1 r_j + A_2 r_j^2\right), \tag{3.100}$$

where,

$$\begin{aligned}
A_0 &= \epsilon_{si}\left\{(V_{SB}+B\phi_F)^{1/2}(V_{SB}+\xi_1 V_{DS}+V_{bi})^{1/2} - (V_{SB}-B\phi_F)\right. \\
&\quad \left. +(\xi_1 V_{DS}+V_{bi}-B\phi_F)\ln\left[\frac{(V_{SB}+B\phi_F)^{1/2}+(V_{SB}+\xi_1 V_{DS}+V_{bi})^{1/2}}{(\xi_1 V_{DS}+V_{bi}-B\phi_F)^{1/2}}\right]\right., \\
A_1 &= (2\epsilon_{si}qN_A)^{1/2}\left[(V_{SB}+V_{bi})^{1/2}+(V_{SB}+\xi_1 V_{DS}+V_{bi})^{1/2}-(V_{bi})^{1/2}\right. \\
&\quad \left. -(\xi_1 V_{DS}+V_{bi})^{1/2}\right], \\
A_2 &= -qN_A \ln\left\{\frac{(r_j+W_D)(r_j+W_S)}{[r_j+W_D(V_{SB}=0)][r_j+W_S(V_{SB}=0)]}\right\}.
\end{aligned} \tag{3.101}$$

The first contribution (A_0) is independent of the depths (r_j) of the source and drain junctions. The second and third terms (A_1 and A_2) vanish for $V_{SB} = 0$. The final expression also includes two empirical parameters ξ and ξ_1 which are found to be approximately equal to 0.5 in [Lee73].

The starting point in Lee's analysis is very rigorous from a theoretical point of view, but because of the complexity of the problem a number of simplifying assumptions have been made. However, in spite of all the assumptions introduced into the analysis, the final expression is very complicated and involved, and not very suitable for circuit simulation programs. In addition, although expression (3.100) includes the essential influences of V_{DS}, V_{SB} and r_j, it is difficult to recognise, without a detailed study, a clear physical image of the short channel effect from this analytical model.

The same approach, as just described, was also used by Bandy and Kokalis [BaKo77]. However, a continuous function describing the potential distribution along the channel surface was defined in contrast to Lee's assumed, arguably nonphysical, constant surface potential in each region of the channel. The model is depicted in Figure 3.12. In order to simplify the analysis it is assumed that the abrupt *pn* junction theory applies at the source and the drain junctions and the surface potential is approximated by a piecewise linear function in the drain and source depletion regions. In the entire channel region the surface potential (ψ_S), as a function of y, is given by (3.87), with $B\phi_F$ replaced by ψ_{SG} which is the surface potential induced by the gate voltage, and given by:

$$\psi_S(y) = V_{SB} + V(y) + \psi_{SG}(y). \tag{3.102}$$

Bandy and Kokalis [BaKo77] also assume linearized distributions for the gate induced surface potential, $\psi_{SG}(y)$, for the bulk potential, $\phi(y)$, and for the channel voltage, $V(y)$,

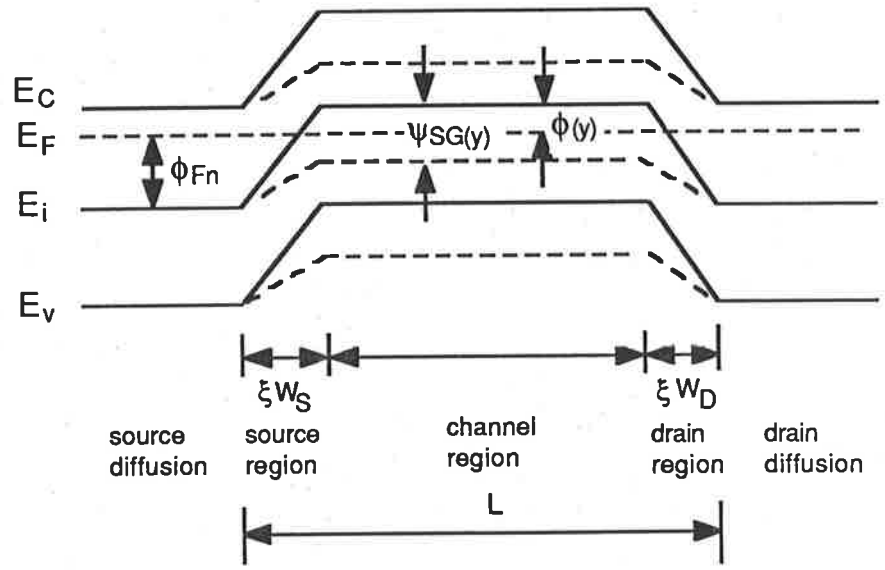


Figure 3.12: Energy band diagram for an n-channel device which considers the source and the drain depletion regions [BaKo77], with: (a) no biases applied (- - -) and (b) strong inversion condition (—).

to describe the overall potential distribution throughout the device. The gate induced surface potential, ie the behaviour of the band bending along the channel, is given by (see Figure 3.12):

$$\psi_{SG}(y) = \begin{cases} 2\phi_F\left(\frac{y}{\xi W_S}\right) & \text{source region,} \\ 2\phi_F & \text{channel region,} \\ 2\phi_F\left(\frac{L-y}{\xi W_D}\right) & \text{drain region,} \end{cases} \tag{3.103}$$

where W_S, W_D and ξ have the same meanings as in Lee's analysis. The bulk potential is approximated by:

$$\phi(y) = \begin{cases} \phi_{Fn} - (\phi_{Fn} + \phi_F)\left(\frac{y}{\xi W_S}\right) & \text{source region,} \\ -\phi_F, & \text{channel region,} \\ \phi_{Fn} - (\phi_{Fn} + \phi_F)\left(\frac{L-y}{\xi W_D}\right) & \text{drain region.} \end{cases} \tag{3.104}$$

The channel voltage is approximated for low drain voltages by:

$$V(y) = V_{DS}\left(\frac{y}{L}\right). \tag{3.105}$$

The quantity $V_{ox}(y)$ can now be defined in the source, channel and drain regions of the channel using expressions (3.103), (3.104), (3.105) in expression (3.85):

$$V_{ox}(y) = \begin{cases} V_{GS} - \Phi_{MS} - V_{bi} - \left(\frac{V_{DS}}{L} + \frac{2\phi_F - V_{bi}}{\xi W_S}\right) y & \text{source region,} \\ V_{GS} - \Phi_{MS} - 2\phi_F - \left(\frac{V_{DS}}{L}\right) y & \text{channel region,} \\ V_{GS} - \Phi_{MS} - V_{bi} + (V_{bi} - 2\phi_F)\left(\frac{L}{\xi W_D}\right) & \\ \times \left(\frac{V_{DS}}{L} + \frac{V_{bi} - 2\phi_F}{\xi W_D}\right) y & \text{drain region,} \end{cases} \tag{3.106}$$

where the built-in potential is given by $V_{bi} = \phi_{Fn} + \phi_F$. By integrating $V_{ox}(y)$ over the corresponding regions, the total charge per unit width on the gate can be obtained, ie the first integral in expression (3.92) can be evaluated. In order to evaluate the second integral in expression (3.92), the space charge differential capacitance $C_{si}(y)$ per unit area is approximated by the following continuous function:

$$C_{si}(y) = \frac{\epsilon_s}{\left[\frac{2\epsilon_s}{qN_A}\psi(y)\right]^{1/2}}, \tag{3.107}$$

in contrast to Lee's regional approximation. The quantities $\psi_1(y)$ and $\psi_2(y) = \psi_1(y) + \psi_S(y)$ in expression (3.92) have the same meanings as in Lee's analysis. Instead of using a geometrical representation of the charge sharing between the source-channel and the channel-drain regions, the effect of charge sharing is included in the model for $\psi_1(y)$ which is the zero bias potential present in the device. This quantity is given by:

$$\psi_1(y) = \begin{cases} V_{bi}\left(1 - \frac{y}{\xi W_S}\right) & \text{source region,} \\ 0 & \text{channel region,} \\ V_{bi}\left(1 - \frac{L-y}{\xi W_S}\right) & \text{drain region.} \end{cases} \tag{3.108}$$

Finally, with the various quantities so defined, integration of expression (3.92) results in the following closed form analytical expression for the short channel effect:

$$\begin{aligned} \Delta V_{TS} &= \frac{W_S}{L}\{V_{bi} - 2\phi_F \\ &- \frac{2}{3}\frac{qN_A}{C_{ox}}\left[\frac{W_C - W_S}{W_C + W_S}(W_C + 2W_S) + 2W_S(V_{SB} = 0)\right]\Big\}. \end{aligned} \tag{3.109}$$

The final expression does not depend on any empirical parameters, since it was found in [BaKo77] that $\xi \approx 1$. It also excludes the influence of the junction depth (r_j). This is due to the assumed linearized potential distribution which neglects the influence of the junction depth on the extension of the source and drain regions into the channel region.

The continuous potential distribution approach was also used by Omura and Ohwada [OmOh78]. In this derivation the surface potential along the channel under the flat band condition ($\psi_{S,fb}(y)$) is approximated by the following quadratic expressions (Figure 3.13(a)), in contrast to the linear expressions used in [BaKo77]:

$$\psi_{S,fb}(y) = \begin{cases} (V_{bi} + V_{SB})\left(1 - \frac{y}{y_{S0}}\right)^2 & \text{source region,} \\ 0 & \text{channel region,} \\ (V_{bi} + V_{DS} + V_{SB})\left(1 - \frac{L-y}{y_{D0}}\right)^2 & \text{drain region,} \end{cases} \tag{3.110}$$

The corresponding space charge region induced in the semiconductor is shown in Figure 3.13(b). The lateral lengths of the source and drain depletion regions (y_{S0} and y_{D0}) are determined from the fringing electric field due to the drain to gate voltage

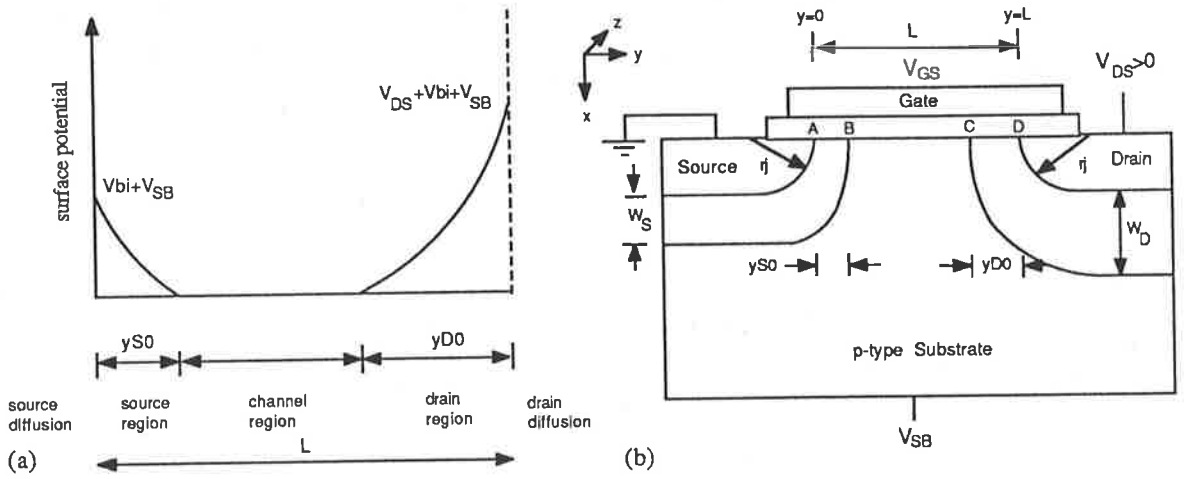


Figure 3.13: (a) A model of the surface potential distribution under the flat band condition, (b) the corresponding space charge region model, after Omura and Ohwada [OmOh79].

drop and the source to gate voltage drop, respectively. The required expressions are derived by solving a one-dimensional cylindrical Poisson equation and making a series of approximations. The results are:

$$y_{D0} = \frac{\left(\frac{2\epsilon_s}{qN_A}|V_{DS}+V_{bi}-V_{FB}|-\frac{r_j^2}{2}\right)^{1/2}}{\left[\ln\left(\frac{2\epsilon_s}{er_j^2qN_A}|V_{DS}+V_{bi}-V_{FB}|\right)^{1/2}\right]^{1/2}}-r_j \tag{3.111}$$

$$y_{S0} = \frac{\left(\frac{2\epsilon_s}{qN_A}|V_{bi}|-\frac{r_j^2}{2}\right)^{1/2}}{\left[\ln\left(\frac{2\epsilon_s}{er_j^2qN_A}|V_{bi}|\right)^{1/2}\right]^{1/2}}-r_j. \tag{3.112}$$

It has been found in the work reported in this thesis that these expressions do not always give sensible results and the use of the exact expressions, analogous to (3.76) and (3.77), are recommended.

At the onset of strong inversion the surface potential distribution, as assumed in [OmOh79], is shown in Figure 3.14(a). It is further assumed, in parallel with Taylor's analysis [Tayl78], that no gate induced surface potential exists between the pinch-off point K and the source junction, and between the pinch-off point L and the drain junction, ie in the channel regions where $\psi_S > (2\phi_F + V_{SB})$. The distance from the pinch-off point L to the drain and the distance from the pinch-off point K to the source are obtained by rearranging (3.110) and are given by:

$$y_D = y_{D0}\left\{1-[(2\phi_F+V_{SB})/(V_{DS}+V_{bi}+V_{SB})]^{1/2}\right\}, \tag{3.113}$$

$$y_S = y_{S0}\left\{1-[(2\phi_F+V_{SB})/(V_{bi}+V_{SB})]^{1/2}\right\}. \tag{3.114}$$

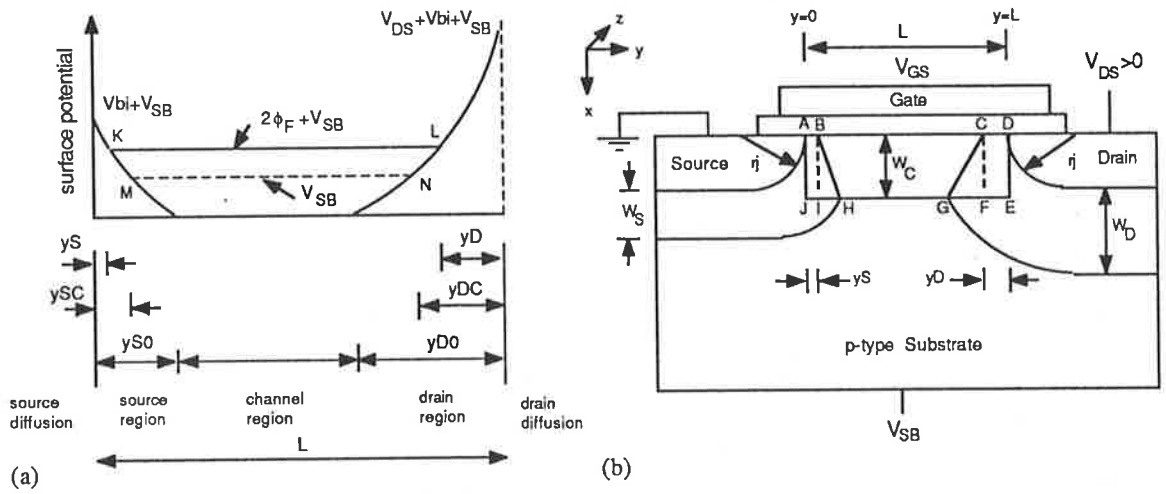


Figure 3.14: (a) A model of the surface potential distribution under the condition of strong inversion, (b) the corresponding space charge region, after Omura and Ohwada [OmOh79]

The value of the surface potential in the channel region, ie between points K and L, is given by $2\phi_F + V_{SB}$. It is also assumed that the surface potential between the pinch-off point K and a point M along the channel a distance y_{SC} from the source (Figure 3.14(a)) is induced by the gate and the source to substrate built-in potentials, and the surface potential between the pinch-off point L and a point N along the channel a distance y_{DC} from the drain is induced by the gate and the drain to substrate built-in potentials. The lengths y_{SC} and y_{DC} are obtained by substituting V_{SB} for $\psi_{S,fb}$ in (3.110) and rearranging:

$$y_{DC} = y_{D0}\left[1 - \left(\frac{V_{SB}}{V_{bi} + V_{SB} + V_{DS}}\right)^{1/2}\right], \tag{3.115}$$

$$y_{SC} = y_{S0}\left[1 - \left(\frac{V_{SB}}{V_{bi} + V_{SB}}\right)^{1/2}\right]. \tag{3.116}$$

Thus, according to this model, only between points M and N is the surface potential solely induced by the gate potential. The gate induced surface potential (ψ_{SG}) can then be expressed as:

$$\psi_{SG}(y) = (2\phi_F + V_{SB}) - \min(\psi_{S,fb} + V_{SB}), \quad y_S \le y \le L - y_D. \tag{3.117}$$

The corresponding space charge region, which is supported by the gate charge, is approximated in the xy plane by the trapezium BCGH, as shown in part (b) of the figure (the length JH= y_{S0}, and the length GE= y_{D0}).

The threshold voltage equation is obtained by evaluating the average value of the gate

induced surface potential and the effective contribution from the space charge:

$$V_{TS} = V_{FB} + \psi_S - \frac{Q_{BS}}{C_{ox}}, \tag{3.118}$$

$$\psi_S = \left(\int_{y_S}^{L-y_D} [(2\phi_F + V_{SB}) - \min(\phi_{S,fb} + V_{SB})] dy \right) / (L - y_S - y_D), \tag{3.119}$$

$$Q_{BS} = -\left(\int_{y_S}^{L-y_D} Q_B dy \right) / (L - y_S - y_D). \tag{3.120}$$

Evaluating the various integrals, the complete threshold voltage expression becomes:

$$\begin{aligned} V_{TS} &= V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}} \\ &+ \frac{2\phi_F}{(L - y_S - y_D)} \left[\frac{V_{SB}}{3\phi_F} (y_{DC} + y_{SC} - y_S - y_D) \right. \\ &- \left. \frac{1}{3} (y_{D0} + y_{S0} - y_S - y_D) \right] \\ &+ \frac{Q_B}{C_{ox}} (y_{D0} + y_D + y_{S0} - y_S) / [2(L - y_S - y_D)]. \end{aligned} \tag{3.121}$$

The complete expression shows the dependence of V_{TS} on V_{SB}, V_{DS} and r_j through equations (3.111)–(3.116), and is also independent of any empirical parameters. The short channel effects are given by terms in lines two to four of the above expression. The terms in lines two and three are a direct consequence of the surface potential not being uniform along the channel and the term in the last line is the well known form of the charge sharing approach. It is interesting to note that by replacing the quantity y_{D0} with W_D (given by (3.67)) and the quantity y_{S0} by W_S (given by (3.66)) the last line in expression (3.121) is the same as the short channel formula given by Taylor [Tayl78], ie expression (3.75).

Toyabe and Asai [ToAs79] used the results of a two-dimensional numerical solution of the Poisson's equation. The authors found that the surface potential had a minimum (ψ_{min}) around the midpoint along the channel, and that $\psi_S - \psi_{min}$ was an exponential function of y. The potential distribution along the x direction, ie into the substrate, was found to be a concave curve in the middle of the channel and a convex function near the drain junction, for drain voltages in excess of the gate voltage. Using the above observations, but only employing analytical methods, a closed form expression is derived for the short channel effect. Their analysis is summarized in what follows.

The two dimensional Poisson's equation reads:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \frac{qN_A}{\epsilon_s}, \tag{3.122}$$

where carrier density is ignored and the impurity concentration (N_A) is assumed to be constant. The results obtained from two-dimensional simulations indicate that the x-dependence of the electrostatic potential (with the reference taken at the source junction

which is at ground potential) can be approximated by a cubic function in x of the form:

$$\psi = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \tag{3.123}$$

The boundary conditions for ψ in the x variable are as follows:

$$\begin{array}{lll} \psi = \psi_S, & \frac{\partial \psi}{\partial x} = \frac{C_{ox}}{\epsilon_s}(\psi_S - V_{GS} + V_{FB}), & \text{for } x = 0, \\ \psi = -V_{SB}, & \frac{\partial \psi}{\partial x} = 0, & \text{for } x = W_C. \end{array} \tag{3.124}$$

Using the above four boundary conditions, the four constant coefficients in (3.123) can be uniquely determined. The following differential equation for the surface potential along the channel is then obtained from (3.122):

$$\frac{d\psi_S^2}{dy^2} - A\psi_S = D, \tag{3.125}$$

where,

$$A = 4\frac{C_{ox}}{\epsilon_s W_C} + \frac{6}{W_C^2}, \tag{3.126}$$

$$D = \frac{q}{\epsilon_s} N_A - \frac{C_{ox}(V_{GS} - V_{FB})}{\epsilon_s W_C} + \frac{6V_{SB}}{W_C^2}. \tag{3.127}$$

The boundary conditions for ψ_S in the y variable are given by:

$$\begin{array}{lll} \psi_S = V_{bi}, & \text{for } y = 0 & \text{source junction}, \\ \psi_S = V_{DS} + V_{bi}, & \text{for } y = L & \text{drain junction}. \end{array} \tag{3.128}$$

The solution of (3.125) is complicated by the fact that the depth (W_C) of the depletion region is a strong function of y in the regions near the drain and source junctions. It is thus assumed in [ToAs79] that the drain and source junctions have negligible depths from which it follows that W_C, as a function of y, can be assumed to be constant. Using these assumptions (3.125) can be solved to give:

$$\psi_S = -D/A + (V_{DS} + V_{bi} + D/A)\exp\left[\sqrt{A}(y - L)\right] + (V_{bi} + D/A)\exp\left(-\sqrt{A}y\right). \tag{3.129}$$

The surface potential given by (3.129) has a minimum given by:

$$\psi_{S,min} = -\frac{D}{A} + 2\sqrt{(V_{DS} + V_{bi} + D/A)(V_{bi} + D/A)}\exp\left(\frac{\sqrt{A}}{2}L\right). \tag{3.130}$$

Substituting (3.126) and (3.127) into (3.130), the following relation between the gate voltage and the surface potential is obtained:

$$\begin{aligned} V_{GS} &= V_{FB} + \psi_{S,min} + \frac{1}{C_{ox}}\sqrt{2\epsilon_s q N_A(\psi_{S,min} + V_{SB})} \\ &- 2\left(1 + \frac{3}{2}\frac{\epsilon_s}{W_C}\frac{1}{C_{ox}}\right)\sqrt{(V_{DS} + V_{bi} + D/A)(V_{bi} + D/A)}\exp\left(-\frac{\sqrt{A}}{2}L\right). \end{aligned} \tag{3.131}$$

The term D/A in expression (3.131) is stated to be approximately equal to $-\psi_{S,min}$. At threshold $V_{GS} = V_{TS}$ and assuming that $\psi_{S,min} = 2\phi_F$, the following analytical expression for the threshold voltage is derived from (3.131):

$$V_{TS} = V_{FB} + 2\phi_F + \frac{1}{C_{ox}}\sqrt{2\epsilon_s q N_A (2\phi_F + V_{SB})}(1-\eta), \tag{3.132}$$

$$\eta = \eta_0 \exp\left(-L/l_0\right), \tag{3.133}$$

$$\eta_0 = \frac{\epsilon_{ox}}{\epsilon_s}\frac{\sqrt{(V_{DS}+V_{bi}-2\phi_F)(V_{bi}-2\phi_F)}}{2\phi_F + V_{SB}}\left(\frac{1}{t_{ox}}W_C + \frac{3}{2}\frac{\epsilon_s}{\epsilon_{ox}}\right), \tag{3.134}$$

$$l_0 = W_C\left(\frac{3}{2} + \frac{\epsilon_{ox}}{\epsilon_s}\frac{W_C}{t_{ox}}\right)^{-1/2} \tag{3.135}$$

Expression (3.132) reduces to the classical formula in the limit $L \gg l_0$. To obtain improved accuracy, the above analytical expressions were modified by replacing the term $2\phi_F$ with $B\phi_F$, where the Brown factor B is given by (3.136), and by introducing a number of empirical constants (c_j) whose values were derived from experimental data and two-dimensional computer simulation results by curve fitting techniques:

$$B = c_1 + \frac{kT}{2\phi_F q}\ln\left(\frac{2\phi_F + V_{SB}}{2\phi_F}\right) - c_2 e^{-L/c_3}, \tag{3.136}$$

$$l_0 = W_C\left(c_4 + c_5\frac{\epsilon_{ox}}{\epsilon_s}\frac{W_C}{t_{ox}}\right)^{-1/2} \tag{3.137}$$

$$\eta_0 = \frac{\epsilon_{ox}}{\epsilon_s}\frac{\sqrt{(V_{DS}+V_{bi}-B\phi_F)(V_{bi}-B\phi_F)}}{B\phi_F + V_{SB}}[c_6(B\phi_F + V_{SB})^{1/2} + c_7], \tag{3.138}$$

$$W_C = \sqrt{\frac{2\epsilon_s}{qN_A}(B\phi_F + V_{SB})}. \tag{3.139}$$

The short channel effect according to Toyabe and Asai [ToAs79] is given by:

$$\Delta V_{TS} = (B - 2\phi_F) - \frac{qN_A W_C}{C_{ox}}\eta_0 \exp\left(-L/l_0\right). \tag{3.140}$$

The analysis presented in [ToAs79] is very elegant in that a very rigorous approach is taken, and the final analytical expression is rather simple. It is interesting to note that the results obtained from the two-dimensional simulations by Toyabe and Asai do not show a strong dependence on the junction depth (r_j). This is in agreement with the main assumption made by the authors about the constant depth of the depletion region along the channel, which can also be interpreted, as pointed out earlier, as an assumption according to which the junction depths are negligible. As a result, the influence of r_j is not included in the final analytical expression and the authors compare their model only with experimental data obtained from transistors with shallow junctions, $r_j = 0.25 \times 10^{-6} m$. Care must therefore be taken in using expression (3.132), and in the opinion of the author of this thesis, the presence of seven empirical parameters

greatly limits the practicality of the modified expressions (3.136)-(3.139) for transistor modelling in a circuit simulation program.

Ratnakumar and Meindl [RaMe82] also solved the two-dimensional Poisson's equations but assumed that $r_j \gg W_C$. Although, in the full form, their final result rivals in complexity the expression derived by Toyabe and Asai [ToAs79] but the authors also give a simplified expression that reads:

$$\Delta V_{TS} = \frac{6t_{ox}}{W_{C0}} [2(V_{bi} + V_{SB}) + V_{DS}] \exp(-\pi L/4W_{C0}), \tag{3.141}$$

where $W_{C0} = W_C(V_{SB} = 0)$. In their paper, Ratnakumar and Meindl demonstrate the validity of expression (3.141) by comparing it with experimental results obtained using transistors with $N_A = 1.2 \times 10^{16} m^{-3}$ ($W_{C0} = 0.3 \times 10^{-6} m$), $r_j = 0.4 \times 10^{-6} m$ and $t_{ox} = 7.2 \times 10^{-9} m$. Good agreement is obtained for transistors with $L < 1.5 \times 10^{-6} m$, as shown in Figure 7 of [RaMe82]. No significant short channel effect is predicted by formula (3.141) for transistors with $L > 1.5 \times 10^{-6}$ (see Figure 4.3), however, the experimental results given in Figure 7 of [RaMe82] show a noticeable deviation, from the long channel MOSFET theory, for transistors with $L \approx 3.0 \times 10^{-6} m$. Hence, it appears that the range of validity of expression (3.141) is rather limited.

3.3.1.3 Empirical Models

The last group of analytical expression presented in this section were derived by the authors in each case empirically fitting experimental results.

Masuda *et al.* [MaNK79] have proposed the following expression for the short channel effect on the basis of experimental results for devices with the drain junction depths in the range $0.15 \times 10^{-6} m < r_j < 0.41 \times 10^{-6} m$ and the substrate impurity concentration in the range $10^{21} m^{-3} < N_A < 10^{22} m^{-3}$:

$$\Delta V_{TS} = -\frac{\eta}{C_{ox}} [2(V_{bi} + V_{SB}) + V_{DS}] L^{-n}, \tag{3.142}$$

where the channel length L is taken in micrometers. In general the parameters n, and η must be found using curve fitting techniques on experimental data. The authors found that for a family of MOS transistors with gate oxide thickness $t_{ox} = 50 \times 10^{-10} m$ the parameters n was independent of the drain junction depth for $0.15 \times 10^{-6} m < r_j < 0.40 \times 10^{-6} m$ and it varied from $n = 2.6$ for $N_A = 10^{21} m^{-3}$ to $n = 3.2$ for $N_A = 10^{22} m^{-3}$. The parameter η was found to depend both on r_j and N_A. For a MOS transistor with $r_j = 0.4 \times 10^{-6} m$, the parameter $\eta = 5.80 \times 10^{-5}$ for $N_A = 10^{21} m^{-3}$ and $\eta = 2.85 \times 10^{-5}$ for $N_A = 10^{22} m^{-3}$, and for a MOS transistor with $N_A = 7.0 \times 10^{21} m^{-3}$, the parameter $\eta = 6.08 \times 10^{-5}$ for $r_j = 0.4 \times 10^{-6} m$, and $\eta = 2.83 \times 10^{-5}$ for $r_j = 0.15 \times 10^{-6} m$.

An empirical relationship, based on the Yau's formulation was developed by Coe *et al.* [CoBN77]. The values of the threshold voltage according to Yau's model were compared both against the threshold voltage values obtained using two-dimensional numerical calculations of the surface potential distribution, and experimental results. It was found that for deep junctions ($r_j/W_C > 1$) the agreement between all three was quite good. However, as the junctions became shallower the decrease in the threshold voltage predicted by Yau's model began to depart from the experimental results and the results obtained by two-dimensional numerical simulations. In order to improve the agreement the model was modified by a fitting procedure and two parameters were introduced:

$$\begin{aligned} a &= 0.94 - 0.17\sqrt{\frac{W_C}{r_j}}, \\ b &= 0.90 - 0.66\, log_{10}\left(\frac{r_j}{W_C}\right) + 0.37\frac{t_{ox}}{W_C} \end{aligned} \tag{3.143}$$

and the reduction in the threshold voltage is given by:

$$\Delta V_{TS} = \frac{Q_B}{C_{ox}}\left\{a\frac{r_j}{L}\left[\left(1+\frac{2W_C}{r_j}\right)^{1/2} - 1\right]\right\}^b . \tag{3.144}$$

The model appears to be valid over a large range of transistor geometries, as demonstrated by these authors, and can easily be incorporated in MOSFET models used for circuit simulation.

Discussion

The a priori charge sharing assumption is, by far, the most popular approach in developing models for the short channel effect. The main advantages of this approach are that it represents the two-dimensional electric field distribution inside the device in an easily understood manner, and it allows for simple closed form expressions to be developed. Detailed study, presented in the previous section, have revealed that in many cases there are only subtle differences between the final expressions, in spite of the initial approximations about the shape of the depletion region charge which is supported by the gate charge being quite different. For example, under the condition of no drain voltage, the expressions (3.58) and (3.55) proposed in [Jant82] and [YaMo83], respectively, differ only by a constant multiplying factor, although in [Jant82] the boundary of the space charge shared by source or drain is assumed to take the form of an ellipse and in [YaMo83] the same boundary is assumed to take the form of a triangle. This would suggest that the approaches taken by some authors to solve the difficult problem of modelling the short channel effect by using imaginative approximations to the shape

of the depletion region and using trigonometry to evaluate the necessary depletion region widths are quite unnecessary, and only lead to long and confusing expressions that unnecessarily complicate the whole picture. Furthermore, the accuracy of such models is limited, as pointed out in [FiPo79] and [RaMe82], to transistors with $L \geq 3 \times 10^{-6}m$, depending on the various process parameters. As demonstrated in [CoBN77], [MaNK79] and [ToAs79], model accuracy may be improved by introducing a number of empirical parameters. However, because the complexities of equations increase rapidly and their physical meaning is lost, as more parameters are introduced, the number of empirical parameters should be kept at a reasonable level.

The models based on the a priori surface potential distribution are considered here to be either to complex, (3.100) given in [Lee73], or too simple, (3.109) given in [BaKo77], in representing the influence of various physical quantities, or were found to contain an excessive number of empirical parameters, (3.132) given in [ToAs79], to be of practical use for circuit simulation.

The analytical models for the short channel effect discussed in this section have assumed a uniform substrate doping concentration. This is a reasonable assumption (although not strictly correct since, in general, some impurity redistribution takes place during oxide growth [Sze81]) for modelling devices with unimplanted channels. In modern fabrication processes channel implantation constitutes a standard processing step [GRWW77], [Penu86], and is used for the purposes of improving device performance. For example, channel implantation is used for threshold voltage adjustment, reduction of punchthrough between source and drain, and reduction of overlap capacitance. In general, further approximations must be used to develop closed form threshold voltage expressions of nonuniformly doped transistors. For a discussion on this subject the reader is referred to [Trou77], [Wang77], [ChLT81], [Asen82] and [RaSa84].

In cases of a shallow channel implant, characterized by a dose D_I, the peak and width of the impurity profile will be very close to the silicon-oxide interface, and the implant will act as a sheet of charge at the silicon-oxide interface [Sze81], [RaSa84]. The threshold voltage expression is then simply modified by an additive term $(qD_I)/C_{ox}$, which term is commonly included in the quantity V_{FB}, ie by treating V_{FB} as an empirical parameter, the effect of shallow channel implant can be taken into account. It was found that this approach was sufficiently accurate to model, over the substrate voltage range of interest $(0 \leq V_{SB} \leq 5V)$, the transistors investigated in this research.

3.3.2 Narrow Width Effect

A narrow channel MOSFET is one in which the width of the channel is of comparable order of magnitude to the depth of gate depletion region under the gate [KrAc76], [Klas77a], [Klas78], [NaBo82]. Figure 3.15 illustrates a width cross-section of a MOSFET. As seen from the figure the polysilicon gate overlaps the thick oxide on both sides of the thin gate oxide, and the thick oxide is tapered and recessed. The gate induces around its edges a fringing field which results in an additional depletion charge being induced under the tapered and thick oxide. As the channel width is reduced, the amount of this extra depletion charge becomes a significant fraction of the total depletion region charge, and its effect must be included in an expression for the threshold voltage. As a result of the extra depletion charge (Q_{BN}) the threshold voltage of narrow width devices is higher than the value predicted by wide transistor theory. Thus, the effective threshold voltage (V_{TN}) of a long and narrow channel transistor is given by:

$$\begin{aligned} V_{TN} &= V_{FB} + 2\phi_F - \frac{Q_B + Q_{BN}/(LW)}{C_{ox}}, \\ &= V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}} - \frac{Q_{BN}}{WLC_{ox}}, \\ &= V_{TL} + \Delta V_{TN}, \end{aligned} \tag{3.145}$$

where ΔV_{TN} is a positive quantity representing the narrow width effect.

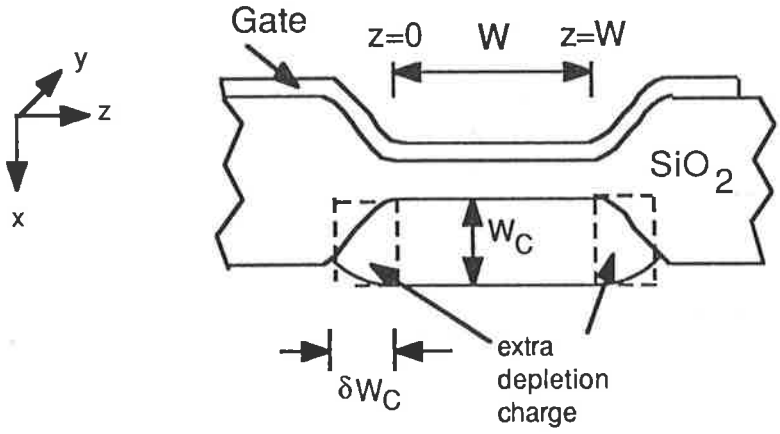


Figure 3.15: Width cross-section.

Kroell and Ackermann [KrAc76] solved the two-dimensional Poisson equation under simple boundary conditions using finite difference method, to explain the narrow width effect. Their work contributed a significant amount to the understanding of the narrow width effect, however, no closed form expression was offered.

One of the first closed form expressions was formulated by Jeppson [Jepp75]. The width cross-section of the region of extra charge is approximated, in the xz plane, by

a rectangle with a depth W_C and a width $\delta_W W_C$, where δ_W is a dimensionless *narrow width* effect empirical parameter. The total extra depletion charge (Q_{BN}), according to Jeppson, is given by:

$$Q_{BN} = -qN_A\delta_W(W_C^2 - W_{C0}^2)L, \tag{3.146}$$

where, as before, L is the length of the channel and W_{C0} is the depth of the depletion region under the gate given by expression (3.38) under the conditions of no substrate bias, ie $W_{C0} = W_C(V_{SB} = 0)$. The resulting expression for the narrow width effect reads:

$$\begin{aligned} \Delta V_{TN} &= -\frac{Q_{BN}}{C_{ox}WL}, \\ &= \frac{qN_A}{C_{ox}}\frac{\delta_W(W_C^2 - W_{C0}^2)}{W}. \end{aligned} \tag{3.147}$$

The model proposed by Akers [Aker81a] considered three approximations of the width cross-sections of the side parts of the depletion region enclosing the total extra charge:

$$\frac{Q_{BN}}{2} = \begin{cases} -qN_ALW_C{}^2/2 & \text{for a triangle,} \\ -qN_ALW_C{}^2\pi/4 & \text{for a quarter circle,} \\ -qN_ALW_C{}^2 & \text{for a rectangle.} \end{cases} \tag{3.148}$$

A general form of the narrow width effect similar to that given by Jeppson was then presented:

$$\begin{aligned} \Delta V_{TN} &= -\frac{Q_{BN}}{C_{ox}WL}, \\ &= \frac{qN_A}{C_{ox}}\frac{\delta'_W W_C^2}{W}, \\ &= -\frac{Q_B}{C_{ox}}\frac{\delta'_W W_C}{W}, \end{aligned} \tag{3.149}$$

where the constant δ'_W has the following values: $\delta'_W = 1$ for a triangle, $\delta'_W = \pi/2$ for a quarter circle and $\delta'_W = 2$ for a rectangle. The last value gave best agreement with the experiment according to Akers [Aker81a]. In his later work [Aker81b] Akers also used the rectangular approximation of the extra depletion charge region. The depth dimension of the rectangular region was taken to be W_C, as before, but the width dimension was taken to be W_{C0}, ie it was assumed that when a substrate bias was applied the width of the extra depletion region remained constant, consequently:

$$\begin{aligned} \Delta V_{TN} &= \frac{qN_A}{C_{ox}}\frac{2W_C W_{C0}}{W}, \\ &= -\frac{Q_B}{C_{ox}}\frac{2W_{C0}}{W}. \end{aligned} \tag{3.150}$$

Merckel [Merc80] used expression (3.149) with $\delta'_W = 1$, ie a triangular region on both sides of the gate was assumed. It was suggested in [Merc80], however, that the constant

δ'_W be replaced by an adjustable narrow width effect parameter δ_W:

$$\Delta V_{TN} = -\frac{Q_B}{C_{ox}}\frac{\delta_W W_C}{W}. \tag{3.151}$$

Silburt *et al.* [SiFP84] used expression (3.149) with $\delta'_W = \pi/2$, ie according to this model the extra depletion charge was contained in a uniform volume, in the length direction, whose width cross-section on both sides of the gate electrode was approximated by a quarter circle. The resulting expression was simplified by introducing a narrow width effect adjustable parameter δ_W:

$$\begin{aligned}\Delta V_{TN} &= \frac{qN_A}{C_{ox}}\frac{\pi}{2}\frac{W_C^2}{W}, \\ &= \frac{\epsilon_s}{C_{ox}}\frac{\pi}{W}(2\phi_F + V_{SB}), \\ &= \frac{\delta_W}{W}(2\phi_F + V_{SB}).\end{aligned} \tag{3.152}$$

Yamaguchi and Morimoto [YaMo83] approximated the region containing the extra depletion charge by a uniform volume in the length dimension with the width cross-section approximated by a rectangle of depth W_C and width $B/2$ (B effectively being an empirical parameter):

$$Q_{BN} = -qN_A LBW_C, \tag{3.153}$$

$$\begin{aligned}\Delta V_{TN} &= -\frac{Q_{BN}}{C_{ox}WL}, \\ &= -\frac{Q_B}{C_{ox}}\frac{B}{W}.\end{aligned} \tag{3.154}$$

Jantsch [Jant82] carried out a similar analysis to that of Akers, to model the narrow width effect. However, a higher field substrate doping concentration (N_F) was introduced to represent the physical device in a more accurate manner. The extra depletion charge was assumed to be contained in two volumes located on either side of the gate electrode. Each volume was assumed to be uniform in the length dimension with the width cross-section given by a rectangle of width $\delta_W W_F$ and depth W_F:

$$\frac{Q_{BN}}{2} = -qN_F\delta_W W_F^2 L, \tag{3.155}$$

where:

$$W_F = \left[\frac{2\epsilon_s}{qN_F}\left(2\frac{kT}{q}\ln\frac{N_F}{n_i} + V_{SB}\right)\right]^{1/2} \tag{3.156}$$

and δ_W being a dimensionless narrow width effect parameter. The final expression for the narrow width effect according to [Jant82] reads:

$$\Delta V_{TN} = \frac{2qN_F}{C_{ox}}\frac{\delta_W {W_F}^2}{W}. \tag{3.157}$$

The expression contains two empirical parameters, δ_W and N_F, compared to only one in other models examined in this section.

Akers *et al.* [AkBC81] developed a more realistic expression that includes the effects of the tapered oxide capacitance, the depletion charge under the recessed field oxide due to gate overlap, and field doping encroachment at the channel edges. The final analytical expression is rather involved. It includes a number of empirical parameters and the following physical quantities: the thickness of the field oxide, the field doping concentration, the amount of gate overlap and the length of the oxide's bird beak in addition to the substrate doping concentration under the gate, the channel width and the gate oxide thickness. As a result of the large amount of information required about the fabrication process, the usefulness of this model, although a very thorough physical representation of the MOS transistor, is very limited for the purpose of circuit simulation.

As a final remark in this section, it should be emphasized that the amount of the threshold voltage shift, with respect to the wide transistor theory, is strongly dependent on the device geometry and accurate treatment is only possible with the use of numerical transistor simulation programs [JiSa83a], [JiSa83b], [ChLa84].

3.3.3 Small Geometry Effect

In order to accurately predict the threshold voltage (V_{th}) of a small geometry transistor, an expression that includes both the short channel effect and the narrow width effect together with a term representing the interaction between the two effects is necessary.

Jeppson [Jepp75] modified Lee's piecewise linear approximation of the short channel effect to include the narrow width effect. For small depletion region depth he proposed the following expression for the small geometry transistor threshold voltage:

$$\begin{aligned} V_{th} &= V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}} \\ &+ \frac{qN_A}{C_{ox}} \left[(W_C - W_{C0}) + \left(\frac{\delta_W}{W} - \frac{1}{L} \right) \left({W_C}^2 - {W_{C0}}^2 \right) \right], \end{aligned} \tag{3.158}$$

where δ_W, defined in (3.147), empirically represents the narrow width effect. This expression, as was also assumed by Wang [Wang78], implies that the short channel and the narrow width effects are independent of each other, and that the threshold voltage expression can be written as an algebraic sum of the threshold voltage of a large geometry transistor and the threshold voltage shift due to the short channel and narrow width effects.

Yamaguchi and Morimoto [YaMo83] and Silburt *et al.* [SiFP84] also used this assumption in deriving an analytical expression for the threshold voltage of a small geometry device. The model for the small geometry MOS transistor threshold voltage given in [YaMo83] reads:

$$\begin{aligned} V_{th} &= V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}}\left(\gamma_S + \frac{B}{W}\right), \\ &= V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}}(\gamma), \end{aligned} \tag{3.159}$$

where γ is the small geometry factor and it combines the short channel effect expressed by the geometry factor γ_S, defined in expression (3.56) and the narrow width effect expressed by the parameter B, defined in expression (3.154). The corresponding formula given in [SiFP84] reads:

$$V_{th} = V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}} + \left(\frac{\delta_W}{W} - \frac{K_S}{L}\right)(2\phi_F + V_{SB}) - \frac{K_D}{L}V_{DS}. \tag{3.160}$$

An expression for the threshold voltage of a small geometry MOSFET was derived from a three-dimensional geometrical approximation of the volume containing the total depletion charge that is supported by the gate charge by Akers [Aker81b]. Thus, his expression includes to the first approximation the effective coupling term between the two effects. The three-dimensional approximation of the required volume is obtained by summing a length cross-section area over the width direction. The length cross-section area is modelled after Yau (Figure 3.4) and is given by:

$$\left\{1 - \left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right]\frac{r_j}{L}\right\} LW_C. \tag{3.161}$$

This length cross-section area needs to be summed over the effective width of the depletion charge region, ie a distance of $W + 2W_{C0}$. The three-dimensional approximation of the volume containing the part of the depletion charge that is supported by the gate charge is then given by:

$$\left\{1 - \left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right]\frac{r_j}{L}\right\} LW_C(W + 2W_{C0}), \tag{3.162}$$

and the total effective depletion charge (Q_{BT}) supported by the gate charge is given by:

$$Q_{BT} = -qN_A\left\{1 - \left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right]\frac{r_j}{L}\right\} LW_C(W + 2W_{C0}). \tag{3.163}$$

The expression for the threshold voltage of a small geometry transistor according to the analysis given in [Aker81b] becomes:

$$V_{th} = V_{FB} + 2\phi_F - \frac{Q_{BT}}{WLC_{ox}},$$

$$= V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}}$$

$$\times \left\{1 - \left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right]\left(\frac{r_j}{L} + \frac{2W_{C0}r_j}{WL}\right) + \frac{2W_{C0}}{W}\right\} \quad (3.164)$$

$$= V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}}\gamma. \quad (3.165)$$

In expression (3.164) the quantity in the square brackets multiplied by r_j/L represents the short channel effect, the term $2W_{C0}/W$ represents the narrow width effect, and the quantity in the square brackets multiplied by $(2W_{C0}r_j)/(WL)$ is a term representing the interaction of the short channel and the narrow width effects.

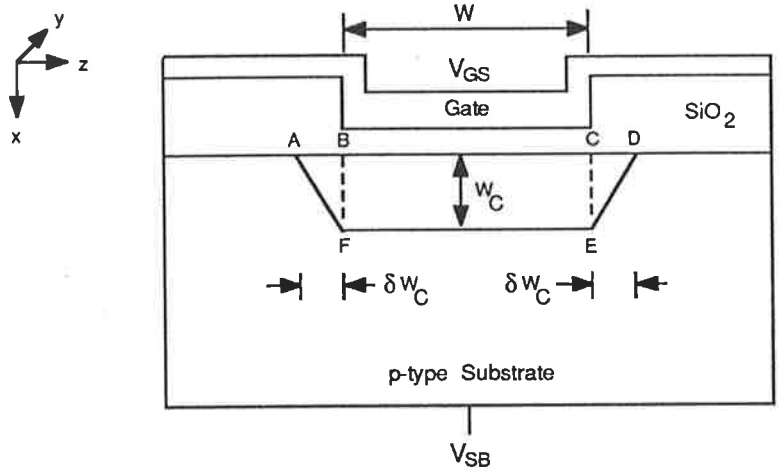


Figure 3.16: Width cross-section approximation in the model used by Merckel [Merc80].

Merckel [Merc80] also used the approach of [Aker81b] to incorporate the narrow width effect, expression (3.151), into the short channel effect formula given by Yau [Yau74]. Figure 3.16 shows the width cross-section of the model used. The total charge induced by the gate voltage is given by:

$$Q_{BT} = Q_{BS} + Q_{BN}, \quad (3.166)$$

where Q_{BS} is the total charge in the depletion region under the gate and given by Yau's analysis, ie:

$$Q_{BS} = -qN_A WLW_C \left\{1 - \frac{r_j}{L}\left[\left(1 + \frac{2W_C}{r_j}\right)^{1/2} - 1\right]\right\}, \quad (3.167)$$

and where Q_{BN} is the total depletion charge supported by the gate fringing field:

$$Q_{BN} = -2qN_A\left[\frac{LW_C\delta_W W_C}{2} - 2\frac{\delta_W W_C^2}{2}\frac{y_1}{3}\right]$$

$$= -qN_A L\delta_W W_C^2\left(1 - \frac{2}{3}\frac{y_1}{L}\right)$$

$$= -qN_A\delta_W W_C^2\left\{1-\frac{2}{3}\frac{r_j}{L}\left[\left(1+\frac{2W_C}{r_j}\right)^{1/2}-1\right]\right\}, \qquad (3.168)$$

where y_1 is defined in Figure 3.4. Combining the expression for the short channel effect with the expression for the narrow width effect results in the following formula for the threshold voltage of a small geometry device:

$$\begin{aligned} V_{th} &= V_{FB}+2\phi_F-\frac{Q_{BT}}{WLC_{ox}} \\ &= V_{FB}+2\phi_F-\frac{Q_B}{C_{ox}} \\ &\times \left\{1-\left[\left(1+\frac{2W_C}{r_j}\right)^{1/2}-1\right]\left(\frac{r_j}{L}+\frac{2}{3}\frac{\delta_W W_C r_j}{WL}\right)+\frac{\delta_W W_C}{W}\right\}, \end{aligned} \qquad (3.169)$$

which is fully analogous to (3.164), with the differences being due to the different approximations of the narrow width effect used by the two authors.

Jantsch [Jant82] formulated a small geometry effect model by combining his model for the short channel effect, expression (3.58), with his narrow width effect model, expression (3.157), in the manner suggested by Akers [Aker81b] to include a term representing the mutual interaction of the two effects. The final expression reads:

$$V_{th} = V_{TL} - 0.7\frac{qN_A}{C_{ox}L}(y_S W_S + y_D W_D) + 2\delta_W\frac{qN_F}{WLC_{ox}}W_F^2(L-g_L W_F), \qquad (3.170)$$

where g_L is an empirical parameter representing the interaction of the short channel and the narrow width effects.

This concludes the discussion of the most important deviations from the large geometry threshold voltage theory experienced in small geometry MOS transistors. The next section examines the techniques used to model the field influence on carrier mobility.

3.3.4 High Field Effects

The mobility of free carriers in the channel is affected both by the *perpendicular* (also referred to as transverse or normal) and the *parallel* (also known as longitudinal, tangential or lateral), to the direction of current flow, electric field components. Analytical models for representing this mobility dependence on the electric field are discussed in the next two subsections.

3.3.4.1 Normal Field Effect

The mechanism that results in a reduction in electron mobility with an increase in the normal electric field component is known as *surface scattering*. As a result of this

scattering mechanism, the mobility of electrons is strongly dependent on the gate voltage and the oxide thickness both of which determine the strength of the normal electric field component. This is a complex mechanism including surface roughness scattering, scattering by interface charges and surface phonons and it is often explained in simple terms as follows (see for example [ArAb66], [Craw67], [Sze81], [Selb84]). As the electric field component perpendicular to the direction of current flow is increased, the free carriers are accelerated towards the silicon-silicon dioxide interface, where they suffer collisions with the silicon surface, which collisions result in decreased mobility. The theoretical models of inversion layer electron mobility (for example [PiSa68], [SaNT72], [ChSu73] and [Ferr76]) are insightful but the presented expressions are too complex for purposes of device modelling and have not yet demonstrated broad applicability. Therefore, all models that are presently used have been constructed on a fully empirical basis with the objective to represent the experimental data as well as possible.

It should also be pointed out that the low field carrier mobility (μ^o) is itself determined by a number of various scattering mechanisms, some of of which include: lattice scattering, ionized impurity scattering, neutral impurity scattering and carrier-carrier scattering [CoWe50], [Choo72], [Hils74], [PlSt77], [LiTh77], [ReRe79], [DoLe81], [NoBL73], [RoAC82], [MaSS83].

Typically, carrier mobility as a function of the normal electric field ($\mu_\perp$) is approximated by combining the low field mobility (μ^o) with the surface scattering component dependent on the normal electric field using the Mathiessen's rule:

$$\frac{1}{\mu_\perp} = \frac{1}{\mu^o} + \frac{1}{\mu^o}\frac{E_\perp}{E_{cx}}, \tag{3.171}$$

where $E_\perp$ is the normal electric field which is a function of the gate voltage, surface potential and the oxide thickness, and E_{cx} is the critical field whose value for a p-type substrate is $7 \times 10^7 Vm^{-1}$ [CoMu80], [AkPo80]. The above mobility expression is normally written as:

$$\mu_\perp = \frac{\mu^o}{1 + E_\perp/E_{cx}}. \tag{3.172}$$

Crawford [Craw67] approximated the normal electric field in expression (3.172) by $E_\perp = (V_{GS} - V_{th})/t_{ox}$ and combined t_{ox} and E_{cx} into an empirical parameter θ, ie:

$$\mu_\perp = \frac{\mu^o}{1 + \theta(V_{GS} - V_{th})}. \tag{3.173}$$

This simple expression has been widely used, for example [MeBC72], [Klas77a], [Dang79], [KlGr80], [CCMG80] and [Risc83].

Rossel *et al.* [RoMV76] approximated $E_\perp$ in expression (3.172) by $(V_{GS} - V_{FB} - \psi_S(y))/t_{ox}$. The resulting relation is attractive because when substituted into (3.5),

results in an expression which can still be integrated to yield an explicit expression for I_{DS}.

Yamaguchi [Yama79] discarded the Mathiessen's rule and instead proposed an expression of the form:

$$\mu_{\perp} = \frac{\mu^o}{(1 + E_{\perp}/E_{cx})^{1/2}}. \tag{3.174}$$

The value for $E_{\perp}$ given in [Yama79] is $6.5 \times 10^6 Vm^{-1}$ for electrons and $1.9 \times 10^6 Vm^{-1}$ for holes. This formula was also used and recommended in [OhWD80] and [EnDM83].

Sabnis and Clemens [SuPl80], [Shic83] demonstrated that doping concentration has an indirect effect on the mobility of carriers in the channel. The higher doping leads to a higher electric field at the interface which forces carriers closer to the interface. As a result, the mobility decreases. It was shown by Sabnis and Clemens that the mobility-field profiles at various doping and substrate bias levels form a universal curve when mobility is plotted as a function of an effective normal electric field. The perpendicular electric field arises both from the bulk depletion charge and the inversion charge. Since the inversion charge (Q_n) is distributed over a small but finite depth ($50 \times 10^{-10}m$ to $100 \times 10^{-10}m$) [Shic83], the average electric field it experiences from its own charge is $Q_n/(2\epsilon_s)$. The total effective perpendicular electric field can thus be approximated by:

$$E_{\perp} = \frac{1}{\epsilon_s}\left(\frac{1}{2}|Q_n| + |Q_B|\right). \tag{3.175}$$

Shichijo [Shic83] used the above result and the usual formula for Q_B, (3.33), approximated Q_n under the condition of strong inversion by:

$$Q_n = -C_{ox}(V_{GS} - V_{th}) \tag{3.176}$$

and used these results in (3.172) to derive the following expression for the surface mobility as a function of the perpendicular electric field:

$$\mu_{\perp} = \frac{\mu^o}{1 + \theta(V_{GS} - V_{th}) + \frac{|Q_B|}{\epsilon_s E_{cx}}}, \tag{3.177}$$

where $\theta = C_{ox}(2\epsilon_s E_{cx})^{-1}$, but it is recommended in [Shic83] to determine its value empirically.

White *et al.* [WhWL80], De La Moneda *et al.* [MoKS82], and Garverick and Sodini [GaSo87] also used (3.177) but in a modified form. The mobile channel charge per unit area is expressed as:

$$Q_n = -C_{ox}(V_{GS} - V_{FB} - 2\phi_F) - Q_B. \tag{3.178}$$

Using this result and expression (3.175) in (3.172) yields:

$$\begin{aligned}\mu_{\perp} &= \frac{\mu^o}{1+\frac{C_{ox}}{2\epsilon_s E_{cx}}\left(V_{GS}-V_{FB}-2\phi_F+\frac{|Q_B|}{C_{ox}}\right)}, \\ &= \frac{\mu^o}{1+\theta\left(V_{GS}-V_{th}+\frac{|2Q_B|}{C_{ox}}\right)}.\end{aligned} \tag{3.179}$$

According to expression (3.173) the mobility increases when a reversed substrate bias is applied. Expressions (3.177) and (3.179), however, predict a decrease in carrier mobility with an increase in the reversed substrate bias. This result is in agreement with the data given by Sabnis and Clemens [SuPl80], White *et al.* [WhWL80], and the experimental results obtained in this work and discussed in Chapter 4.

A different empirical formula for the dependence of mobility on the vertical electric field was suggested by Frohman-Bentchkowsky [Froh68] who approximated the experimental findings of Leistico *et al.* [LeGS65] by the following expression:

$$\mu_{\perp} = \mu^o \left(\frac{E_S}{E_{\perp}}\right)^{c_1}, \tag{3.180}$$

where $E_S = 6 \times 10^6 V m^{-1}$ and $c_1 = 0.36$ (for n-channel MOSFETs) are empirical constants, and the normal electric field is approximated by the following expression:

$$E_{\perp} = (V_{GS} - V_{FB} - 2\phi_F - 0.5V_{DS})C_{ox}/\epsilon_s. \tag{3.181}$$

Expression (3.180) has been used in [YaMo83] with $E_{\perp}$ approximated by:

$$E_{\perp} = (V_{GS} - V_{th} - 0.5V_{DS})C_{ox}/\epsilon_s, \tag{3.182}$$

which represents the substrate bias effect on $\mu_{\perp}$ incorrectly, according to [WhWL80] and the experimental results of this work.

Using expression (3.175) Sun and Plummer [SuPl80] modified (3.180) to include the effects of interface charges and substrate doping:

$$\begin{aligned}\mu_{\perp} &= \frac{\mu^o}{1+\alpha Q_{ss}}\left(\frac{E_S}{E_{\perp}}\right)^{c_1}, \\ \mu^o &= 0.349 - 0.0164(\log_{10} N_A - 6.0), \\ \alpha &= -1.04 \times 10^{-3} + 1.93 \times 10^{-5}(\log_{10} N_A - 6.0), \\ c_1 &= 0.341 - 5.44 \times 10^{-3}(\log_{10} N_A - 6.0), \\ E_{cx} &= 6.495 \times 10^{-8} \exp(1.30 \times 10^{-5} Q_{ss}) N_A^{0.25}.\end{aligned} \tag{3.183}$$

In this analysis N_A is expressed in m^{-3}, Q_{ss} in units of $10^{15} m^{-2}$ and then the field dependent mobility $\mu_{\perp}$ is obtained in SI units, ie $m^2 V^{-1} s^{-1}$. In [SuPl80] $E_{\perp}$ is correctly expressed using (3.175).

3.3.4.2 Velocity Saturation Effect

The variation of carrier mobility with electric field was first reported in [RySh51] for bulk germanium and in [Ryde53] for bulk silicon. More extensive studies have since been conducted by others [Gibb67]. The essential feature of their results was that carrier mobility decreased with an increase in the electric field for lightly doped materials. Furthermore, the carrier drift velocity saturated for n-type samples for fields in excess of $2 \times 10^6 Vm^-$ and thc saturation velocity was found to be about $10^5 ms^{-1}$ at room temperature. The original interpretation of this behaviour [Shoc51] was based on a simple physical model in which acoustic phonon emission was alone considered responsible for limiting the drift velocity in the high field limit. Carriers gain energy from the field between collisions through which they directly or indirectly loose it to the lattice. The carriers, thus, on average acquire more energy than they have at thermal equilibrium and on the whole the mean energy of the electron gas is increased. Balancing the rate at which energy is gained from the field with the rate of loss to acoustic phonons leads to the following equation for the field dependent mobility at low electric fields:

$$\mu_{\|} = \frac{\mu^o}{\left\{\frac{1}{2} + \frac{1}{2}\left[1 + \frac{3\pi}{8}\left(\frac{\mu^o E_{\|}}{c_a}\right)^2\right]^{1/2}\right\}^{1/2}} \tag{3.184}$$

where c_a denotes the speed of the longitudinal acoustic phonons and its value is $1.66 \times 10^4 ms^{-1}$. A semi-empirical expression of this form was used in [ZiNu74] to obtain the current voltage characteristics of an MOS device.

The physics behind velocity saturation are complicated and most of the models presently in use are of the empirical nature. Dacey and Ross [DaRo55] proposed the following relationship for the mobility dependence on the longitudinal field:

$$\mu_{\|} = \mu^o \left(E_{cy}/E_{\|}\right)^{1/2}. \tag{3.185}$$

Trofimenkoff [Trof65] suggested that the hyperbolic mobility-field relationship was more realistic:

$$\mu_{\|} = \frac{\mu^o}{\left[1 + (E_{\|}/E_{cy})\right]}. \tag{3.186}$$

This formula has the advantage of allowing a closed form expression for the current-voltage characteristics and therefore it has been used by a number of authors [Trof65], [Wrig70], [HoMe72] and [MaRV72].

Caughey and Thomas [CaTh67] used the measurements of carrier drift velocity dependence on the field strength performed by [Irvi62], [NoGi67] and [RoRN67] and summa-

$E_{cy,n}$ $[Vm^{-1}]$	β_n	$E_{cy,p}$ $[Vm^{-1}]$	β_p	Reference
8.00×10^5	2.00	1.95×10^6	1.00	[CaTh67]
7.24×10^5	1.30			[CMMO74]
6.98×10^5	1.11	1.80×10^6	1.21	[CMMO75]
8.70×10^5	2.90	1.20×10^6	2.60	[CoMu80]
2.00×10^6	1.00			[ONNI80]
1.00×10^6	2.00			[AkPo80]

Table 3.1: Coefficients for velocity saturation given by (3.187), for $T = 300K$. The subscript n indicates electrons and p indicates holes.

rized their findings by the following expression:

$$\mu_{\|} = \frac{\mu^o}{\left[1 + \left(\frac{E_{\|}}{E_{cy}}\right)^{\beta}\right]^{1/\beta}} \tag{3.187}$$

where β and E_{cy} are parameters dependent on temperature [JCOQ77] and for electrons are given by:

$$E_{cy} = 6.98 \times 10^5 \left(\frac{T}{300}\right)^{1.55} Vm^{-1}, \tag{3.188}$$

$$\beta = 1.11 \left(\frac{T}{300}\right)^{0.66}. \tag{3.189}$$

Some of the numerical values given by various authors are given in Table 3.1.

The value of saturation velocity associated with (3.187) is given by $v_{sat} = \mu^o E_{cy}$. This implies, however, that the saturation velocity depends on the impurity concentration through μ^o, which is a function of impurity concentration. Sabnis and Clemens have demonstrated on the basis of their experimental findings that v_{sat} is almost completely independent of the impurity concentration. Thornber [Thor80] has pointed out that as a consequence of these results it is incorrect to use expression (3.187) and proposed instead:

$$\mu_{\|} = \frac{\mu^o}{\left[1 + \left(\frac{\mu^o E_{\|}}{v_{sat}}\right)^{\beta}\right]^{1/\beta}}. \tag{3.190}$$

A typical value for v_{sat} at $T = 300K$ is $10^5 ms^{-1}$ [CMMO74], [JCOQ77].

Scharfetter and Gummel [ScGu69] proposed the following empirical expression which includes impurity and electric fields effects at room temperature:

$$\mu_{\|} = \mu^{oo} \left\{1 + \frac{N_A}{N_A/S + N_{ref}} + \frac{(E_{\|}/A)^2}{E_{\|}/A + F} + \left(\frac{E_{\|}}{B}\right)^2\right\}^{-1/2}, \tag{3.191}$$

where the empirical constants are μ^{oo}, N_{ref}, A, B, S and F, whose values are given in [ScGu69], [JCOQ77] and [Yama83], for example. The above expression can be written in terms of the low field mobility, which in this case is given as a function of doping concentration by [Thor80]:

$$\mu^o = \frac{\mu^{oo}}{\left[1 + \frac{N_A}{N_A/S + N_{ref}}\right]^{1/2}}, \tag{3.192}$$

ie:

$$\mu_{\|} = \mu^o \left\{ 1 + \left[\left(\mu^o \frac{E_{\|}}{\mu^{oo} A} \right)^2 / \left(\frac{\mu^o E_{\|}}{\mu^{oo} A} + F \right) + \left(\frac{\mu^o E_{\|}}{\mu^{oo} B} \right)^2 \right] \right\}^{-1/2}. \tag{3.193}$$

According to (3.193) the impurity scattering dependence is contained in μ^o and the saturation velocity $\mu^{oo}B$ is independent of N_A. This is in agreement with the results of Sabnis and Clemens. This complicated formula has been mainly used in numerical MOS transistor simulation programs [Yama83], [COCM86].

Several other formulae have been used by various authors for modelling the influence of the longitudinal electric field on mobility, and an excellent treatment of this subject can be found in [Selb84] and [Engl86].

There are a number of approaches used to derive an expression for the effective mobility as a function of both components of the electric field $\mu = \mu(E_{\perp}, E_{\|})$. A common approach is to combine the two effects by multiplying them together [AkPo80], [Yama83], [Shic83]:

$$\mu = \mu_{\perp} \mu_{\|} / \mu^o. \tag{3.194}$$

For example, substituting expressions (3.172) and (3.186) into expression (3.194) results in the following mobility expression for field dependent mobility:

$$\mu = \frac{\mu^o}{(1 + E_{\perp}/E_{cx})\left(1 + E_{\|}/E_{cy}\right)}. \tag{3.195}$$

A different approach suggested by Thornber [Thor80] is to replace μ^o in an expression for $\mu_{\|}$ by $\mu_{\perp}$. Combining expressions (3.172) and (3.190) with $\beta = 1$ using this approach, results in the mobility model used by [MeBC72], [Dang79], [WhWL80] and [HaCD82]:

$$\begin{aligned} \mu &= \frac{\mu^o}{(1 + E_{\perp}/E_{cx})} \frac{1}{\left\{1 + \left[\frac{\mu^o}{(1+E_{\perp}/E_{cx})} \frac{E_{\|}}{v_{sat}}\right]\right\}}, \\ &= \frac{\mu^o}{1 + \frac{E_{\perp}}{E_{cx}} + \mu^o \frac{E_{\|}}{v_{sat}}}. \end{aligned} \tag{3.196}$$

Expression (3.196) is equivalent to considering μ^o, $\mu_{\perp}$ and $\mu_{\|}$ as being independent and combining them using the Mathiessen's rule:

$$\frac{1}{\mu} = \frac{1}{\mu^o} + \frac{1}{\mu_{\perp}} + \frac{1}{\mu_{\|}},$$

$$\mu = \frac{\mu^o}{1 + \mu^o/\mu_\perp + \mu^o/\mu_\parallel}. \tag{3.197}$$

The final major effect that becomes significant as the physical dimensions of devices are decreased is the effect of the drain and source series resistances.

3.3.5 Junction Resistances

As the devices get smaller the effect of the resistances associated with the source and drain junctions become of increasing importance [SuJo80], [Mans82] and [Risc83]. This effect acts like an apparent increase in surface scattering parameter θ. This can be demonstrated as follows. Consider the following simplified drain current expression in the linear mode of operation:

$$I_{DS} = \frac{\mu^o}{1 + \theta_0(V'_{GS} - V_{th})} \frac{W}{L} C_{ox}(V'_{GS} - V_{th})V'_{DS}, \tag{3.198}$$

where the effective terminal voltages are given by:

$$\begin{aligned} V'_{GS} &= V_{GS} - I_{DS}R_S \approx V_{GS}, \\ V'_{DS} &= V_{DS} - I_{DS}R_T. \end{aligned} \tag{3.199}$$

The quantity R_S is the resistance associated with the source junction, R_D is the resistance associated with the drain junction and $R_T = R_S + R_D$. A direct substitution of expressions (3.199) into expression (3.198) yields:

$$I_{DS} = \frac{\mu^o}{1 + [\theta_0(V_{GS} - V_{th})]} \frac{W}{L} C_{ox}\,(V_{GS} - V_{th})\,(V_{DS} - I_{DS}R_T) \tag{3.200}$$

which reduces to:

$$I_{DS} = \frac{\mu^o}{1 + (\theta_0 + \mu^o \frac{W}{L} C_{ox} R_T)(V_{GS} - V_{th})} (V_{GS} - V_{th})V_{DS}. \tag{3.201}$$

Thus, it can be seen that any resistance in series with the device can be interpreted as an additional mobility degradation term. In fabrication processes which use deep junctions, thick oxides and large contact windows, the term added to θ_0 in (3.201) can be neglected, however, in aggressive, short channel fabrication processes the series resistance effect is not negligible and must be included.

The total series resistance (R_T) is composed of the resistance (R_w) of any wire in series with the transistor, contact resistance (R_c), and the drain and source junction resistances (R_j) [Mans82]:

$$\begin{aligned} R_T &= R_w + R_j + R_c, \\ &= R_w + R_c + \rho \frac{L_D}{W r_j}, \end{aligned} \tag{3.202}$$

where ρ is the resistivity of the diffused junction and L_D is the diffusion interconnection length from the edge of the channel to the contact. Thus, θ can now be written in the following form:

$$\begin{aligned}\theta &= \theta_0 + \mu^o \frac{W}{L} C_{ox} R_T, \\ &= \theta_0 + \mu^o C_{ox}(R_w + R_c)\frac{W}{L} + \mu^o C_{ox}\rho\frac{L_D}{r_j L} \\ &= \theta_0 + \theta_1\frac{W}{L} + \frac{\theta_2}{L}, \end{aligned} \tag{3.203}$$

where θ_0 represents the carrier mobility degradation with the perpendicular component of the electric field, θ_2 represents the effect of the junction resistance and θ_1 represents the effects of the drain and source contact resistances and any resistance external to the device.

The remaining sections of this chapter examine the most common techniques of incorporating the various small geometry effects, discussed so far, into analytical expressions for the transistor current-voltage characteristics.

3.3.6 Linear Region of Operation

In the previous sections the various effects which become dominant with decreasing physical dimensions of a MOS transistor have been considered separately. In a practical device all such effects must be considered simultaneously. Such a treatment would require considering the interactions between these effects and the expressions obtained would be very complicated. Therefore, an empirical approach is usually taken, where it is assumed that the interactions between these effects are negligible. This approach is characterized by an attempt to preserve the general form of the current-voltage relations, derived for large geometry transistor,s and to extend the range of validity of such expressions by introducing additional expressions and adjustable parameters, so that the modified expressions can be used for small geometry devices. What is considered to justify such an approach is its success in modelling the transistor behaviour observed experimentally.

The basic approach taken in deriving expressions for the drain current in the linear region of operation assumes the validity of the gradual channel approximation, and uses equation (3.4) presented in Section 3.2.1, ie:

$$I_{DS} = -W\mu(E_\perp, E_\parallel)Q_n\frac{dV}{dy}, \tag{3.204}$$

however, since the mobility is voltage dependent it needs to be included inside the subsequent integral. The final expressions are derived by various authors by using different

models for the field dependent mobility $\mu(E_\perp, E_\parallel)$, by employing different models for the small geometry effect to obtain Q_n, and by making different simplifying assumptions.

One of the first models for the DC current-voltage characteristics of a short channel MOS device was developed by Merckel *et al.* [MeBC72]. Expression (3.197) is used to model the field dependent mobility, and the following expressions are used for components of the right hand side:

$$\frac{\mu^o}{\mu_\parallel} = \eta \left|E_\parallel\right| = \eta \frac{dV}{dy}, \tag{3.205}$$

$$\frac{\mu^o}{\mu_\perp} = \eta_\theta \left|E_\perp\right|. \tag{3.206}$$

The normal electric field component ($E_\perp$) is approximated using (3.175) with the depletion charge under the gate (Q_B) and the mobile charge (Q_n), both per unit area, given from the large geometry MOSFET theory by expressions (3.33) and (3.34), respectively:

$$E_\perp = \frac{C_{ox}}{2\epsilon_s}\left\{V_{GS} - V_{FB} - 2\phi_F - V + \frac{1}{C_{ox}}\left[2\epsilon_s q N_A \left(2\phi_F + V + V_{SB}\right)\right]^{1/2}\right\}. \tag{3.207}$$

Substituting expression (3.34) for Q_n and the mobility model just described in expression (3.204) gives an expression for the transistor drain current as a function of V:

$$I_{DS} = \frac{\mu^o W C_{ox}\left[V_{GS} - V_{FB} - 2\phi_F - V - \frac{1}{C_{ox}}\sqrt{2\epsilon_s q N_A (V + 2\phi_F + V_{SB})}\right]}{1 + \theta\left[V_{GS} - V_{FB} - 2\phi_F - V + \frac{1}{C_{ox}}\sqrt{2\epsilon_s q N_A (V + 2\phi_F + V_{SB})}\right] + \eta\frac{dV}{dy}} \frac{dV}{dy}, \tag{3.208}$$

where the mobility degradation parameter θ is given by:

$$\theta = \frac{\eta_\theta}{2}\frac{C_{ox}}{\epsilon_s} \tag{3.209}$$

and V_{th} is given by the long channel transistor model (3.32). It is possible to integrate equation (3.208), but the result is rather complicated involving logarithmic terms (see the cited reference). The authors resolve to a simplified analysis which yields:

$$\begin{aligned} I_{DS} &= \frac{\mu^o}{1 + \theta(V_{GS} - V_{th})}\frac{W}{L}C_{ox}\left\{(V_{GS} - V_{FB} - 2\phi_F)V_{DS} - \frac{V_{DS}^2}{2}\right. \\ &- \left.\frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}\left[(V_{DS} + 2\phi_F + V_{SB})^{3/2} - (2\phi_F + V_{SB})^{3/2}\right]\right\}. \end{aligned} \tag{3.210}$$

The final expression includes the effect of the perpendicular electric field on carrier mobility. This effect becomes significant, as discussed in Section 3.3.4.1, in transistors with small vertical dimensions rather than small mask dimensions. The use of the variable V_{th} is not consistent in this model, since it only explicitly appears in the model for the field effective mobility and the model does not account for the short channel or the narrow channel effect. In addition the velocity saturation effect is not included. It

is possible, however, (not done in [MeBC72]) to include the effects of source and drain resistances by making the parameter θ depend on the channel length and width, as described in Section 3.3.5. Finally, it is interesting to note that the analysis followed in [MeBC72] led to an expression that could have simply been arrived at by substituting the mobility expression (3.173) into the classical expression (3.28) derived in [InMo64].

Fukuma and Okuto [FuOk80] investigated the current-voltage characteristics using two-dimensional device simulations, and on the basis of their findings reported a model for the current-voltage characteristics claimed to be accurate for devices with effective channel lengths as low as 1 micrometer. The velocity saturation effect is modelled with (3.186) and is considered, by these authors, to be the main influence on the mobility of carriers in the channel. Using (3.186) for the field dependent mobility and (3.34) for the channel charge per unit area in (3.204), and integrating yields a formula for the drain current in the linear region of operation:

$$\begin{aligned} I_{DS} &= \frac{\mu^o}{\left(1+\frac{V_{DS}}{LE_{cy}}\right)}\frac{W}{L}C_{ox}\left\{(V_{GS}-V_{FB}-2\phi_F)V_{DS}-\frac{V_{DS}^2}{2}\right. \\ &- \left.\frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}[(V_{DS}+V_{SB}+2\phi_F)^{3/2}-(V_{SB}+2\phi_F)^{3/2}]\right\}. \end{aligned} \tag{3.211}$$

Expression (3.211) again corresponds to the classical expression (3.28) modified by an expression for the field dependent mobility. In contrast to (3.210), however, the modification involves a model for the velocity saturation effect.

Yamaguchi and Morimoto [YaMo83] incorporated their small geometry threshold voltage model, (3.159), into an expression for the drain current. The short channel effect is incorporated in the form of a correction factor, an approach suggested by Taylor [Tayl79]. The expression representing the charge neutrality condition, (3.29), modified to include the small geometry effect is given by:

$$Q_G + Q_{ss} + \gamma Q_B + Q_n = 0. \tag{3.212}$$

Using this relation, it is possible to obtain the following expression, analogous to (3.34), which gives the channel charge per unit area:

$$Q_n = -C_{ox}(V_{GS}-V_{FB}-2\phi_F-V)-\gamma\sqrt{2\epsilon_s q N_A(2\phi_F+V_{SB}+V)}, \tag{3.213}$$

where γ is defined in expression (3.159). Substituting (3.213) into (3.204) and integrating as before yields (γ is treated as a constant during the integration process):

$$\begin{aligned} I_{DS} &= \mu\frac{W}{L}C_{ox}\left\{(V_{GS}-V_{FB}-2\phi_F)V_{DS}-\frac{1}{2}V_{DS}^2\right. \\ &- \left.\frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}\gamma\left[(V_{DS}+2\phi_F+V_{SB})^{3/2}-(2\phi_F+V_{SB})\right]^{3/2}\right\}, \end{aligned} \tag{3.214}$$

which is the expression derived by Inhantola-Moll with the short and narrow channel effects included in γ. The expression for μ includes the transversal field dependence of carrier mobility which is modelled by (3.180) and the carrier velocity saturation modelled by (3.190) with $\beta = 1$, ie:

$$\begin{aligned} \mu &= \mu^o f_1 f_2, \\ f_1 &= \left[\frac{\epsilon_s E_S}{C_{ox}(V_{GS} - V_{th} - 0.5V_{DS})}\right]^{c_1}, \\ f_2 &= \frac{1}{\left(1 + \frac{\mu^o f_1}{v_{sat}}\frac{V_{DS}}{L}\right)}, \end{aligned} \tag{3.215}$$

where the threshold voltage V_{th} is given by (3.159). All voltage dependent terms in (3.159) and (3.216) are held constant during integration. The complete model includes the small geometry effect and accounts for both the velocity saturation and the surface scattering effects.

Dang [Dang79] used the same approach as [MeBC72] and extended his long channel device model given in [Dang77b], by deriving an analogous expression to (3.214). Both the gate voltage dependence and the drain voltage dependence of carrier mobility are included by using the mobility model given by (3.196). The drain current in the linear region of operation, according to Dang, is given by integrating expression (3.216) which includes both the diffusion and drift current components:

$$I_{DS} = \mu W \left[-d\psi_S/dy Q_n + (kT/q) dQ_n/dy)\right], \tag{3.216}$$

from source ($\psi_{S0} = 2\phi_F$, $y = 0$) to drain ($\psi_{SL} = 2\phi_F + V_{DS}$, $y = L$) resulting in the following expression (again, all voltage dependent terms in the expressions for μ and γ_S are held constant during integration) for the drain current in the linear region of operation:

$$\begin{aligned} I_{DS} &= \mu \frac{W}{L} C_{ox} \left\{ \left(V_{GS} - V_{FB} - 2\phi_F + \frac{kT}{q} \right) V_{DS} - \frac{V_{DS}^2}{2} \right. \\ &- \frac{2}{3} \frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}} \gamma_S \left[(V_{DS} + 2\phi_F + V_{SB})^{3/2} - (2\phi_F + V_{SB})^{3/2} \right] \\ &+ \left. \frac{kT}{q} \frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}} \gamma_S \left[(V_{DS} + 2\phi_F + V_{SB})^{1/2} - (2\phi_F + V_{SB})^{1/2} \right] \right\}, \end{aligned} \tag{3.217}$$

where γ_S is the short channel effect geometrical factor calculated according to Dang's threshold voltage model (3.83) and the field dependent carrier mobility is given by:

$$\mu = \frac{1}{1 + \theta(V_{GS} - V_{th}) + \frac{\mu^o}{v_{sat}} \frac{V_{DS}}{L}}. \tag{3.218}$$

The expression for V_{th} only includes the short channel effect. The diffusion term in expression (3.217) contributes only a few percent to the drain current [Dang79], but it

becomes important near turn-on and near saturation. This model does not include the narrow channel effect and the effect of the source and the drain junction resistances. It is a simple matter, however, to include the narrow width effect by modifying γ_S, as demonstrated in [Aker8b] or [YaMo83]. The effects of the source and the drain resistances can be included by modifying the mobility degradation parameter θ, as discussed in Section 3.3.5.

The expressions for the drain current, discussed so far, involve computations of two terms involving the 3/2 power terms. The remaining models for the drain current in the linear region of operation employ a numerical approximation to eliminate the 3/2 power terms in the final expression and incorporate V_{th} in an explicit form.

Merckel *et al.* [MeBC72] used the following approximation to represent the depletion charge contribution:

$$\sqrt{2\phi_F + V + V_{SB}} \approx \sqrt{2\phi_F + V_{SB}} + \frac{V}{\sqrt{2\phi_F + V_{SB}}}. \tag{3.219}$$

This allows to include the device threshold voltage in an explicit form while retaining the bulk doping term. The simplified differential equation for the drain current according to [MeBC72] reads:

$$I_{DS} = \frac{\mu^o}{1 + \theta(V_{GS} - V_{th})} W C_{ox} \left[(V_{GS} - V_{th}) - V(1 + \delta)\right] \frac{dV}{dy}, \tag{3.220}$$

from which the current expression valid in the linear mode of operation is obtained by integration:

$$I_{DS} = \frac{\mu^o}{1 + \theta(V_{GS} - V_{th})} \frac{W}{L} C_{ox} \left[(V_{GS} - V_{th}) V_{DS} - \frac{V_{DS}^2}{2}(1 + \delta)\right], \tag{3.221}$$

where

$$\delta = \frac{\sqrt{2\epsilon_s q N_A}/C_{ox}}{2\sqrt{2\phi_F + V_{SB}}}. \tag{3.222}$$

Expression (3.220) contains the V_{th} parameter in an explicit form. This makes it possible to properly account for the small geometry effects (although this was not done in [MeBC72]) by using an appropriate expression for V_{th}. The mobility dependence on the normal electric field is included using expression (3.173), but the velocity saturation is not accounted for in this analysis.

Klassen and de Groot [KlGr80] modified expression (3.220). Firstly the velocity saturation effect modelled by (3.186) is included, and secondly the short channel effect is also included by using the following expression for V_{th}:

$$V_{th} = V_{FB} + 2\phi_F - \gamma_S Q_B - \alpha V_{DS}, \tag{3.223}$$

where the short channel effect factor γ_S is given according to Yau's model [Yau74] (expression (3.46)), and since that model does not include the dependence of V_{th} on V_{DS}, this effect is modelled in an empirical way, suggested in [TrFo77], [MaNK79] with the parameter α. The resulting differential equation which is the starting point in this analysis reads:

$$I_{DS} = \frac{\mu^o W C_{ox} [V_{GS} - V_{th} - V(1+\delta)]}{[1+\theta(V_{GS}-V_{th})]\left[1+\frac{dV/dy}{E_{cy}}\right]} \cdot \frac{dV}{dy}, \tag{3.224}$$

where δ is again given by (3.222). In reference [Klas78] by one of these authors, however, δ is represented by a semi-empirical expression obtained by replacing the constant 2 in the denominator of (3.222) with 2.5.

Integration of expression (3.224) from source to drain, with the voltage dependent term in the model for V_{th} held constant, results in the following expression for the drain current in the linear mode of operation:

$$I_{DS} = \frac{\mu^o W C_{ox}\left[(V_{GS}-V_{th})V_{DS} - \frac{V_{DS}^2}{2}(1+\delta)\right]}{L[1+\theta(V_{GS}-V_{th})][1+V_{DS}/(LE_{cy})]}. \tag{3.225}$$

Expression (3.225) is analogous to (3.221) but it contains an additional factor in the denominator, which factor represents the velocity saturation effect.

Hanafi *et al.* [HaCD82] used the following, numerically more accurate, approximation for δ, the body effect term, in their model:

$$\delta = \frac{\sqrt{2\epsilon_s q N_A}/C_{ox}}{2\sqrt{2\phi_F + V_{SB}}}\left[1 - \frac{1}{1.744 - 0.836(2\phi_F + V_{SB})}\right]. \tag{3.226}$$

A detailed discussion of the derivation of (3.226) can be found in the appendix of [HaCD82]. The field dependent mobility is modelled in [HaCD82] with expression (3.196) with the normal electric field $E_\perp$ approximated by:

$$E_\perp = (|Q_B| + |Q_n|)/\epsilon_s, \tag{3.227}$$

rather than the more commonly accepted (3.175). The starting point is a differential equation equivalent to (3.224):

$$I_{DS} = \frac{\mu^o W C_{ox}[V_{GS} - V_{th} - V(1+\delta)]}{\left[1+\theta(V_{GS}-2\phi_F - V_{FB} - V) + \eta\frac{dV}{dy}\right]}\frac{dV}{dy}. \tag{3.228}$$

This is a simplified version of expression (3.208) with the bulk charge term missing in the denominator as a result of (3.227) being used for $E_\perp$ instead of (3.175). In order to simplify the integration of equation (3.228), a constant channel electric field in the lateral direction is assumed, ie $\frac{dV}{dy} = \frac{V_{DS}}{L}$. This permits an average of the surface scattering contribution to be taken over the channel:

$$\frac{1}{L}\int_0^L (V_{GS} - V_{FB} - 2\phi_F - V)dy = \frac{1}{V_{DS}}\int_0^{V_{DS}} (V_{GS} - V_{FB} - 2\phi_F - V)dV. \tag{3.229}$$

With the preceding simplification expression (3.228) may be integrated to give:

$$I_{DS} = \frac{\mu^o C_{ox} W \left[(V_{GS} - V_{th}) V_{DS} - \frac{V_{DS}^2}{2}(1+\delta) \right]}{1 + \theta \left[V_{GS} - V_{th} - \frac{V_{DS}}{2} + \frac{1}{C_{ox}} \sqrt{2\epsilon_s q N_A (2\phi_F + V_{SB})} \right] + \eta \frac{V_{DS}}{L}} \tag{3.230}$$

where θ and η are two mobility modulation parameters of the form:

$$\theta = \theta_0 + \frac{\theta_2}{L}, \tag{3.231}$$

$$\eta = \eta_0 L + \eta_2. \tag{3.232}$$

In expression (3.231) θ_0 represents the mobility degradation due to surface scattering of carriers in the inversion channel and θ_2/L is interpreted as the effect of the source and drain junction resistances on the mobility of carriers. In expression (3.232) $\eta_0 V_{DS}$ is due to the errors introduced by the approximations used in deriving the mobility expression and $\eta_2 V_{DS}/L$ represents the saturation velocity phenomenon. No analytical formula for the threshold voltage is given in [HaCD82], where V_{th} is treated as an adjustable parameter.

A very similar model was given by Silburt *et al.* [SiFP84]. This model differs from that given in Hanafi *et al.* [HaCD82] in the expression used for the threshold voltage which is not treated as a model parameter but calculated using the threshold voltage model given by (3.160). An additional difference between the analysis presented in [SiFP84] to the analysis presented in [HaCD82] is the use of expression (3.175) to approximate the quantity $E_\perp$ in the expression for the field dependent mobility.

Ipri *et al.* [IMGB82] proposed the following expression for the linear region of operation for devices with channel lengths below $2 \times 10^{-6} m$ on the basis of results obtained from two-dimensional device simulations:

$$I_{DS} = \frac{\mu^o}{\left[1 + \left(\frac{\mu^o}{v_{sat}} \frac{V_{DS}}{L} \right)^2 \right]^{1/2}} \frac{W}{L} C_{ox} \left(V_{GS} - V_{th} - \frac{V_{DS}}{2} \right) V_{DS}. \tag{3.233}$$

This expression corresponds to a simple treatment where the contribution of the bulk charge term is neglected. The velocity saturation effect is treated as a disturbance (in a mathematical sense) and is incorporated using expression (3.187).

All models described in this section are valid in the linear region of operation only, as pointed out in the introduction of this section. The transition to the saturation region is characterized by a saturation voltage. The common expressions used to model the saturation voltage of short and narrow channel transistors are reviewed in the following section.

3.3.7 Saturation Voltage

In early theory of the long channel MOSFET the saturation of the drain current was attributed to the fact that the channel charge decreased to zero at the drain end of the channel for $V_{DS} = V_{sat}$, where V_{sat} is the saturation voltage. The channel was said to be *pinched-off*. The saturation voltage was determined from $Q_n(V_{sat}) = 0$, as discussed in Section 3.2.4.

The geometry normally used to analyse the short channel transistor beyond saturation is given in Figure 3.17. For drain voltages larger than the saturation voltage the channel is subdivided into a source region, corresponding to the channel section between the source junction and point P, and a drain region, corresponding to the channel section between point P and the drain junction. The point P along the channel is referred to by many authors as the pinch-off point, in analogy to long channel transistor theory. It is defined for short channel transistors as the point along the channel at which the longitudinal electric field has reached some critical value (E_{cy}) and the carriers travel at the saturation velocity (v_{sat}).

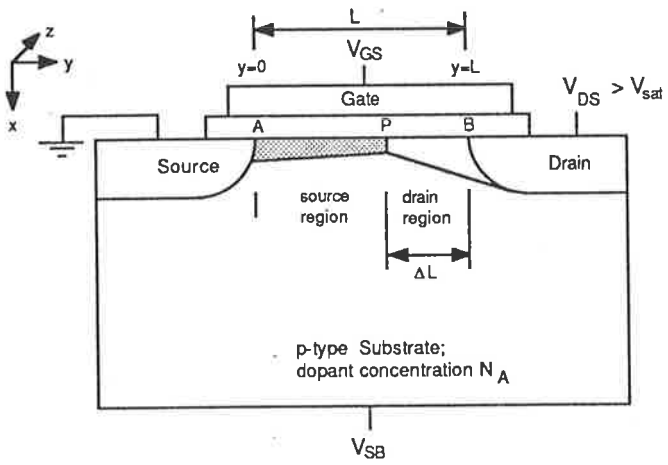


Figure 3.17: A simple saturation model for a short channel MOS transistor, after [MeBC72].

In practice it is not possible to identify a particular channel length which divides transistors into two groups: one group in which current saturation is attributed to channel pinch-off and the second group in which the saturation is attributed to velocity saturation. The early models can be regarded as a subset of the models derived using velocity saturation effect as the current saturation mechanism, which models reduce to the early ones for large channel lengths.

In the model developed by Merckel *et al.* [MeBC72] the saturation voltage is obtained from (3.208) and (3.210) using the condition that at saturation $V_{DS} = V_{sat}$ and the lateral electric field at the saturation point in the channel is given by $E_{\|} = E_{cy}$. Substituting $dV/dy = |E_{cy}|$ in expression (3.208) and $V_{DS} = V_{sat}$ in expressions (3.208) and (3.210) and equating them yields:

$$\begin{aligned} & LE_{cy}\left\{V_{GS} - V_{FB} - 2\phi_F - V_{sat} - \frac{1}{C_{ox}}\left[2\epsilon_s q N_A (V_{sat} + 2\phi_F + V_{SB})\right]^{1/2}\right\} \\ = \; & (V_{GS} - V_{FB} - 2\phi_F)V_{sat} - \frac{V_{sat}^2}{2} \\ - \; & \frac{2}{3}\frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}\left[(V_{sat} + 2\phi_F + V_{SB})^{3/2} - (2\phi_F + V_{SB})^{3/2}\right], \end{aligned} \tag{3.234}$$

which must be solved for V_{sat} using an iterative technique. This value is then used in equation (3.210) to calculate the corresponding saturation current (I_{sat}).

A closed form expression for the saturation voltage results, if equations (3.220) and (3.221) are employed in place of (3.208) and (3.210), as described in [MeBC72] and [HaCD82]. Equating (3.220) with $dV/dy = |E_{cy}|$ and $V_{DS} = V_{sat}$ and (3.221) with $V_{DS} = V_{sat}$ gives:

$$\left[V_{GS} - V_{th} - V_{sat}(1+\delta)\right] LE_{cy} = (V_{GS} - V_{th})V_{sat} - \frac{V_{sat}^2}{2}(1+\delta). \tag{3.235}$$

Solving for V_{sat} yields:

$$V_{sat} = \frac{V_{GS} - V_{th}}{1+\delta} + LE_{cy} - \left[\left(\frac{V_{GS} - V_{th}}{1+\delta}\right)^2 + (LE_{cy})^2\right]^{1/2}. \tag{3.236}$$

This result gives either a first order approximation by putting $\delta = 0$ or a second order approximation with $\delta \neq 0$.

Yamaguchi and Morimoto [YaMo83] estimated the surface carrier density, n_S, at the point along the channel where the carriers have reached the saturation velocity by:

$$n_S = \frac{I_{sat}}{W q v_{sat}}, \tag{3.237}$$

where I_{sat} is given by expression (3.214) evaluated at $V_{DS} = V_{sat}$. If it is also assumed that the gradual channel approximation remains valid at the velocity saturation point, the carrier concentration at the velocity saturation point can be found by:

$$n_S = \frac{C_{ox}}{q}\left\{V_{GS} - V_{FB} - 2\phi_F - \frac{\gamma}{C_{ox}}\left[2\epsilon_s q N_A (2\phi_F + V_{SB} + V_{sat})\right]^{1/2} - V_{sat}\right\}. \tag{3.238}$$

Equating (3.237) and (3.238) results in the following expression:

$$\frac{I_{sat}}{W q v_{sat}} = \frac{C_{ox}}{q}\left\{V_{GS} - V_{FB} - 2\phi_F - \frac{\gamma}{C_{ox}}\left[2\epsilon_s q N_A (V_{sat} + 2\phi_F + V_{SB})\right]^{1/2}\right\}, \tag{3.239}$$

which must be numerically solved for V_{sat}. Although not stated in [YaMo83], this condition for saturation is equivalent to (3.234). This can be easily shown by substituting (3.214) with $V_{DS} = V_{sat}$ for I_{sat} in (3.239) and using $E_{cy} = v_{sat}/\mu$. However, note that since this last relationship is not used in the analysis of [YaMo83] expression (3.239) contains the field dependent mobility expression through the expression for I_{sat}, whereas expression (3.234) does not.

Fukuma and Okuto [FuOk80] also used this approach to calculate the saturation voltage. In their analysis the drain current equation (3.211) with $\gamma = 1$ is used, in place of (3.214).

Klassen and de Groot [KlGr80] assumed that at the point along the channel beyond which the gate does no longer control the current the longitudinal electric field has a value $E_{\|} = E_p > 3E_{cy}$ (based on numerical solutions of the basic semiconductor equations). The potential at the point is defined as V_{sat}. The saturation voltage is obtained from the continuity condition (the approach first proposed by Merckel *et al.* [MeBC72]) which requires that expressions (3.224) and (3.225) give the same result at $V_{DS} = V_{sat}$. The solution reads:

$$\begin{aligned} V_{sat} &= \frac{LE_{cy}}{1 - E_{cy}/E_p}\left\{1 + \left[\frac{V_{GS} - V_{th}}{(1+\delta)LE_p}\right]\right\} \\ &\times \left\{\left[\frac{2(V_{GS} - V_{th})(1 - E_{cy}/E_p)}{\left[1 + \frac{V_{GS}-V_{th}}{(1+\delta)LE_p}\right]^2 (1+\delta)LE_{cy}} + 1\right]^{1/2} - 1\right\}. \end{aligned} \tag{3.240}$$

For the limiting case of $E_p \to \infty$, the above expression simplifies to:

$$V_{sat} = \left[\frac{2(V_{GS} - V_{th})}{1+\delta}LE_{cy} + (LE_{cy})^2\right]^{1/2} - LE_{cy}. \tag{3.241}$$

Expressions (3.240) and (3.241) differ only by few percent if the result $E_p/E_{cy} = 3$ is used. In this case the saturation voltage is a function of V_{DS} as well as of V_{GS} through the threshold voltage dependence on V_{DS}, expressed by (3.223).

In the models developed by Dang [Dang79], Ipri *et al.* [IMGB82] and Silburt *et al.* [SiFP84] the saturation voltage is found from the condition used for the long channel transistors:

$$dI_{DS}/dV_{DS} = 0 \qquad \text{at} \quad V_{DS} = V_{sat}. \tag{3.242}$$

For small L and $V_{GS} \gg V_{th}$ the result given by Ipri *et al.* [IMGB82] which is derived by applying condition (3.242) to expression (3.233) reads:

$$V_{sat} = \left[\left(\frac{v_{sat}L}{\mu^o}\right)^2 (V_{GS} - V_{th})\right]^{1/3}. \tag{3.243}$$

The expression derived by Silburt *et al.* [SiFP84] by applying condition (3.242) to expression (3.230) reads:

$$V_{sat} = \frac{2(V_{GS} - V_{th})}{1+\delta}\left[1 + \left(1 + \frac{2(V_{GS} - V_{th})}{(1+\delta)E_{cy}L[1+\theta(V_{GS} - V_{th} - 2\frac{\gamma Q_B}{C_{ox}})]}\right)^{1/2}\right]^{-1}, \quad (3.244)$$

where δ is given by (3.226), and γ accounts for the small geometry effects.

An analogous treatment of (3.217), given in [Dang79], results in an expression for the saturation voltage which must be solved using iterative techniques, eg the Newton-Raphson method.

As discussed in Section 3.2 the drain current for long channel transistors remains constant beyond saturation, however, short channel transistors experience a finite output conductance in saturation. The next section describes the common approaches to modelling this effect.

3.3.8 Finite Output Conductance in Saturation

The first departure from the long channel transistor model of the current-voltage characteristics, observed experimentally, is the finite output conductance in saturation which occurs even for devices with moderate channel lengths, $L < 25 \times 10^{-6}m$. In the analysis of this effect, it is assumed that the gradual channel approximation is still valid in the source region of the channel, and that the gate does not control the drain current in the drain region. The drain current in saturation ($V_{DS} \geq V_{sat}$) is normally modelled by:

$$I_{DS} = \frac{I_{sat}}{1 - \Delta L/L}, \quad (3.245)$$

where the saturation current I_{sat} is given by $I_{sat} = I_{DS}(V_{GS}, V_{sat}, V_{SB})$ and ΔL is the length of the drain region (Figure 3.17). The finite output conductance in saturation is attributed to the spreading of the drain region with the drain bias which results in a reduction of the effective channel length [ReSa65], [FrGr69], [Sze81]. To compute the exact values of ΔL and V_{sat} is a difficult mathematical problem because of the two-dimensional nature of the electric field near the drain. Therefore, all models proposed and in use are semiempirical in nature.

All the models discussed in this section have a common starting point given below:

$$\frac{\partial^2 \psi_S}{\partial y^2} = -\frac{\rho_S}{\epsilon_s} - \frac{\partial^2 \psi_S}{\partial x^2}, \quad (3.246)$$

where ρ_S is the surface carrier density. Equation (3.246) is then approximately solved in the drain region of the channel, subject to the following boundary conditions in the

y variable:

$$\begin{aligned} \psi_S(y = L - \Delta L) - \psi_S(y = L) &= V_{sat} - V_{DS}, \\ \frac{\partial \psi_S}{\partial y}(y = L - \Delta L) &= E_i \end{aligned} \tag{3.247}$$

and with the use of a number of approximations in the x variable. The quantity E_i is the unknown lateral electric field at the transition point of the two regions to be empirically or otherwise assigned later.

The concept of *channel shrinkage* (ie quantity ΔL, where ΔL is zero for $V_{DS} \leq V_{sat}$) was first introduced by Reddi and Sah [ReSa65]. In their model the authors employed the pinch-off condition, ie $Q_n(V_{sat}) = 0$, and the following approximations:

$$E_i = 0, \tag{3.248}$$

$$\frac{\partial^2}{\partial x^2}\psi_S = 0, \tag{3.249}$$

$$\rho_S = -qN_A. \tag{3.250}$$

to solve (3.246) and obtain an explicit equation for ΔL of the form:

$$\Delta L = \sqrt{\frac{2\epsilon_s}{qN_A}(V_{DS} - V_{sat})}. \tag{3.251}$$

The above one-dimensional model totally ignores the effect of the gate potential on the electric field near the drain. Thus, the strength of the electric field near the drain is underestimated and consequently (3.251) overestimates the channel shortening effect, a result experimentally confirmed in [FrGr69], [MeBC72], [Popa72], [Mans82].

Frohman-Bentchkowsky and Grove [FrGr69] proposed, on the basis of their experimental results, the following modification to (3.251):

$$\Delta L = \frac{V_{DS} - V_{sat}}{E_{||}}, \tag{3.252}$$

where $E_{||}$ is the average transversal electric field in the drain region. Three different contributions to $E_{||}$ are identified:

1. The electric field component (E_1) due to the fixed charge in the reversed biased drain junction (this is the only field component in the Reddi-Sah analysis);
2. The fringing electric field (E_2) due to the potential difference between the drain and the gate; and
3. The fringing electric field (E_3) due to the potential difference $V_{GS} - V_{sat}$ between the gate and the end of the channel.

The component E_1 is approximated using the analysis of Reddi and Sah [ReSa65] and is given by (3.253) below:

$$\begin{aligned} E_1 &= \frac{V_{DS} - V_{sat}}{\Delta L}, \\ &= \frac{V_{DS} - V_{sat}}{\left[\frac{2\epsilon_s}{qN_A}(V_{DS} - V_{sat})\right]^{1/2}}, \\ &= \left[\frac{qN_A}{2\epsilon_s}(V_{DS} - V_{sat})\right]^{1/2}. \end{aligned} \tag{3.253}$$

The components E_2 and E_3 are taken to be proportional to the respective normal fields across the oxide:

$$E_2 = \alpha_1 \frac{\epsilon_{ox}}{\epsilon_s} \frac{V_{DS} - V_{GS}}{t_{ox}}, \tag{3.254}$$

$$E_3 = \alpha_2 \frac{\epsilon_{ox}}{\epsilon_s} \frac{V_{GS} - V_{sat}}{t_{ox}}, \tag{3.255}$$

where α_1 and α_2 are two field fringing factors whose values according to [FrGr69] are $\alpha_1 \approx 0.2$ and $\alpha_2 \approx 0.6$. The extent of channel length reduction is then expressed as:

$$\Delta L = \frac{V_{DS} - V_{sat}}{E_1 + E_2 + E_3}. \tag{3.256}$$

Expression (3.256), although developed for large geometry transistors, was adopted by Yamaguchi and Morimoto [YaMo83] in their analysis of the current voltage characteristics of a small geometry MOS transistor.

According to these simple models the conduction in saturation is as follows. Electrons travelling along the channel enter the drain region of the channel, which is practically a depletion region. The electric field in that region is in such a direction, however, as to sweep these electrons through this depletion region to the drain.

In short channel MOS devices, this definition of pinch-off as a condition for onset of saturation is not realistic; the principle saturation mechanism is the velocity saturation of carriers. In this formulation the pinch-off is taken to occur when the field near the drain becomes sufficiently high to cause velocity saturation. For $V_{DS} > V_{sat}$ the electrons are assumed to travel in the drain region of the channel at maximum velocity, while in the source region of the channel a linear relationship between the carrier velocity and the transversal electric field is assumed to hold. Thus, is this formulation the channel reduction does not correspond to the spreading of the drain depletion region into the channel but rather to the length of the channel where the mobile carriers travel with the saturation velocity.

Baum and Beneking [BaBe70] used the above saturation model according to which the following condition holds at the transition point:

$$E_i = \frac{v_{sat}}{\mu^o} = E_{cy}, \tag{3.257}$$

where E_{cy} is the critical electric field at which the carrier velocity saturates. Solution of (3.246) using (3.247), (3.249) and (3.250), assuming that the electric field lines in the drain region are approximately horizontal, results in the following expression for ΔL:

$$\Delta L = \sqrt{\frac{2\epsilon_s}{qN_A}} \left\{ \left[\frac{\epsilon_s E_{cy}^2}{2qN_A} + (V_{DS} - V_{sat}) \right]^{1/2} - \left(\frac{\epsilon_s E_{cy}^2}{2qN_A} \right)^{1/2} \right\}. \tag{3.258}$$

In addition to the assumption made by Baum and Beneking [BaBe70], Merckel *et al.* [MeBC72] assumed that mobile carriers in the drain region are pushed away from the surface and are spread uniformly over a depth ranging from inversion layer thickness (d) to the drain junction depth (r_j). The effect of the mobile carriers on the space charge was also included, ie:

$$\rho_S = -\left(qN_A + \frac{I_{DS}}{W x v_{sat}} \right), \tag{3.259}$$

where:

$$x = \frac{r_j - d}{\Delta L} y + d. \tag{3.260}$$

The Poisson equation (3.246) is solved using (3.247), (3.259) and (3.260), and the approximation $d \ll r_j$ to give:

$$V_{DS} - V_{sat} = \frac{qN_A}{2\epsilon_s} \Delta L^2 \left[1 + \frac{2I_{DS}}{qN_A W v_{sat} r_j} \left(\ln \frac{r_j}{d} - 1 \right) \right] + \Delta L E_{cy}. \tag{3.261}$$

The quantity ΔL is calculated by solving expressions (3.245) and (3.261) simultaneously, which solution must be carried out using iterative methods.

Expression (3.261) can be simplified to give ΔL explicitly by substituting $I_{DS} = I_{sat}$ [MeBC72], [HaCD82]:

$$\frac{\Delta L}{L} = \frac{\left[(LE_{cy})^2 + 2A(1 + BI_{sat})(V_{DS} - V_{sat}) \right]^{1/2} - LE_{cy}}{A(1 + BI_{sat})}, \tag{3.262}$$

where:

$$\begin{aligned} A &= \frac{qN_A L^2}{\epsilon_s}, \\ B &= 2\left[\ln(r_j/d) - 1 \right] / (qN_A v_{sat} W r_j). \end{aligned} \tag{3.263}$$

Popa [Popa72] and Rossel *et al.* [RoMV76] simplified Merckel's analysis and assumed that the channel in the drain region has a uniform effective thickness $\eta_1 r_j$, with η_1 being a dimensionless empirical parameter. Expression (3.259) can then be simplified to:

$$\rho_S = -\left(qN_A + \frac{I_{DS}}{W v_{sat} \eta_1 r_j} \right). \tag{3.264}$$

The Poisson equation, (3.246), is then solved as before to give:

$$\Delta L = \frac{\left\{E_{cy}^2 + \frac{2}{\epsilon_s}\left(qN_A + \frac{I_{DS}}{W\eta_1 r_j v_{sat}}\right)(V_{DS} - V_{sat})\right\}^{1/2} - E_{cy}}{\frac{1}{\epsilon_s}\left(qN_A + \frac{I_{DS}}{W\eta_1 r_j v_{sat}}\right)}, \tag{3.265}$$

which is solved simultaneously with (3.245) for ΔL. The above analysis was also adopted in [Dang77b], [Dang79], [FuOk80] and [LeGD81].

Klassen and de Groot [KlGr80] used results from two-dimensional numerical simulations and proposed the following approximation:

$$\frac{\partial^2 \psi_S}{\partial x^2} = -\frac{\psi_S - (V_{GS} - V_{th})}{X_o^2}, \tag{3.266}$$

where the quantity X_o has a value of the order of the drain diffusion layer thickness. The magnitude of the electric field (E_i) was found using two-dimensional numerical simulations to be $E_i > 3E_p$ at the velocity saturation point. The authors introduced the quantity E_p, as an adjustable parameter, to reflect this fact. The solution of Poisson's equation, given by (3.246), subject to the boundary conditions, given by (3.247), and using (3.266) yields in a simplified form:

$$\Delta L = \alpha_{sat} L \left\{ \left[V_{DS} - V_{sat} + \left(\frac{\alpha_{sat} L E_p}{2}\right)^2 \right]^{1/2} - \frac{\alpha_{sat} L E_p}{2} \right\}, \tag{3.267}$$

where $\alpha_{sat} = [\epsilon_s/(qN_A L^2)]^{1/2}$ but the authors suggest it should be regarded as an empirical parameter.

Silburt *et al.* [SiFP84] proposed a very simple empirical expression for the channel modulation effect:

$$\Delta L = C_M (V_{DS} - V_{sat}), \tag{3.268}$$

where, C_M is an empirical parameter.

Ipri *et al.* [IMGB82] could not model their experimental data in the saturation region by using an expression analogous to the channel modulation effect, as was done in [FuOk80], [MeBC72] and [HaCD82]. The following empirical expression was proposed instead:

$$I_{DS} = \frac{10^{-8}(V_{GS} - V_{th})^{1/2}}{(t_{ox})^{1/2} L}(V_{DS} - V_{sat}) + I_{sat}, \tag{3.269}$$

where I_{sat} is the saturation current given by (3.233) for $V_{DS} = V_{sat}$, and where as usual the quantities are in SI units.

3.4 Summary

The traditional approach to the analytical modelling of MOS transistors is to use a number of assumptions and simplifying approximations in the solution of the basic semiconductor equations. It is then possible to formulate closed form expressions which describe the current-voltage characteristics of MOS transistors. The accuracy of such expressions are often improved by introducing a number of adjustable parameters, whose values are determined using experimental results. Analytical models are especially useful in circuit simulation programs and in providing a basic insight both into the operation of MOS transistors and the dependence of the transistor terminal characteristics on a number of important process parameters and mask dimensions.

The large geometry MOS transistor theory is based on the assumption that the electric field lines are everywhere perpendicular to the channel surface. Using this assumption, also known as the gradual channel approximation, it is possible to derive closed form expressions using one-dimensional analysis. The equations that form the basis for practically all analytical MOS transistor models presently in use were derived by Inhantola and Moll [InMo64], and Pao and Sah [PaSa66a], [PaSa66b]. It has been shown how the charge sheet model which was proposed by Brews [Brew78] relates to the Pao-Sah model, and how various modifications of the charge sheet analysis lead to the classical "text book" model of Inhantola and Moll. However, equations derived in this way fail to characterize MOS transistors with short and narrow channels.

The main approach in deriving analytical models for small geometry MOS transistors is to introduce additional expressions into the classical equations used for the large geometry transistors, which expressions model the physical phenomena whose influences become significant in small geometry MOS transistors. These additional expressions retain terms with physical meaning, where possible, and include a number of empirical parameters to model subtle device characteristics. For each single effects a number of different approaches have been proposed in literature.

It has been shown in this chapter that, although in many cases the particular assumptions made, and the final expressions obtained, for a particular effect by different authors, appear to be very different. A closer examination of these expressions has revealed that in many cases there are only subtle differences. For example, the final short channel effect expressions derived by [Jant82] and [YaMo83] differ only by a constant multiplying factor of 1.4, under the condition of no drain voltage, in spite of the initial approximations about the shape of the depletion region charge which contributes to the threshold voltage being radically different. Most models proposed for the narrow channel effects

can also be expressed in one form, by introducing one parameter, which parameter is calculated from geometry by some authors and treated as an adjustable quantity by others.

Most of the analytical models have been assessed by comparison with experimental data. The new improved analytical model, which has been incorporated into the circuit analysis program, FACTS, has been formulated by using mathematical approaches described in this chapter. The resulting analytical model is described and compared with experimental results in the following chapter.

Chapter 4

MOSFET Analytical Model for Circuit Simulation

4.1 Introduction

It has become apparent, from the study presented in Chapter 3, that the analytical MOS transistor models for the current-voltage characteristics, presented by different authors, either do not account for all major effects present in small geometry transistors or that some expressions used to model particular physical effects are too simple and cannot possibly provide sufficient levels of accuracy.

The experimental results presented in literature, used to demonstrate the validity of a particular model being proposed, have been found to be of limited value in this work for the purposes of comparing different models. In general, different authors use results from different fabrication processes. It then becomes difficult to decide whether the model is limited to the range of the presented experimental data or whether the expression has a wider range of application. For example, the short channel threshold voltage models of Yamaguchi and Morimoto [YaMo83], expression (3.55), and Jantsch [Jant82], expression (3.58), differ by a constant multiplying factor 1.4 (under the condition $V_{DS} = 0$) yet both are claimed to be in good agreement with experimental result. The main process parameters are: $r_j = 0.4 \times 10^{-6}m$, $N_A = 10^{22}m^{-3}$, $t_{ox} = 42 \times 10^{-9}m$ for [YaMo83] and $r_j = 0.4 \times 10^{-9}m$, $N_A = 4 \times 10^{21}m^{-3}$, $t_{ox} = 30 \times 10^{-9}m$ for [Jant82]. Thus, in order to carry out a unified comparison of the analytical models, reviewed in Chapter 3, measurements on test structures obtained from two different fabrication processes (described in Section 4.4) were performed in this work.

Most of the short channel effect models, studied in Chapter 3, have been found to

overestimate the short channel effect (Figure 4.3), and the threshold voltage dependences on the substrate bias (Figure 4.4) and on the drain bias (Figure 4.5). This result has been previously reported by Fichtner and Pötzl [FiPo79] who proposed a model based on the model of [Yau74], ie expressions (3.55), (3.64) and (3.65). It has been found, however, that although for the transistors considered in this work their model well represents the short channel effect, it underestimates the effect of the substrate voltage (Figure 4.4) and overestimates the effect of the drain voltage (Figure 4.5).

The expressions of [ToAs79] and [RaMe82], which are based on two-dimensional numerical simulations and the a priori potential distribution, predict a much greater decrease in the threshold voltage of devices with channel lengths in the submicrometer region (Figure 4.3), than the models based on the a priori charge sharing assumption.

With regard to complete current-voltage transistor models, it has been found that for a single transistor many models, eg [Dang79], [YaMo83], [HaCD82] and [SiFP84], accurately represent the experimental measurements in the linear and saturation regions of operation, but when they are applied to devices (from the same process) having widely varying geometries the agreement is rather unsatisfactory. This is not a surprising result, since apart from the model of [HaCD82], none of the other models incorporate geometry dependence of the respective model parameters. Thus, the only solution is to have a range of model parameters for a selected group of transistors. Although the model of [HaDC82] incorporated a large degree of geometry dependence, a number of the following problems have been identified:

1. It does not include a model for V_{th} but instead treats this quantity as a model parameter to be extracted from experimental data;
2. It employs a numerical approximation for the body effect term, expression (3.226), rather than the exact expression;
3. It uses expression (3.227) to approximate $E_{\perp}$, instead of the more accurate (3.175) suggested and used in [SuPl80], [WhWL80], [MoKS82], [Shic83] and [GaSo87]; and
4. The geometry dependence of the mobility degradation parameter (θ), given by (3.232), has been found to be less satisfactory than a more general (3.203).

On the basis of these findings an improved analytical model is proposed herein which represents within the experimental error the current-voltage characteristics, in the linear and saturation regions of operation, of the complete set of test structures investigated in this work. The proposed model includes all of the major second order effects which are important for accurate modelling of small geometry MOS transistors. One of the

objective was to develop a model with the least possible number of empirical parameters. This approach ensures that the model is physically based as far as possible (which potentially increases the forecasting capability of the model), reduces the effort of parameter extraction and is less likely to require "strange" sets of parameters. The model includes geometry dependence of all relevant model parameters and, consequently, the only required input is the mask length and width for each transistor fabricated using a given fabrication process. The main approach used in formulating the new model is as follows: in cases where an existing model for a particular physical effect has been determined to be satisfactory, that model has been used, otherwise a new expression has been formulated.

An examination of expressions presented in Chapter 3 quickly reveals that the complexity, and thus the computational requirements, of a model incorporating even simple expressions to represent small geometry effects is such that a direct implementation of such a model into the circuit analysis program, FACTS, would result in an unacceptable time response of the program. This is because, especially for large circuits, most of the simulation time is spent in evaluating the transistor models [ChGK75], [SSMY82]. A well established technique used to increase the response time of circuit analysis programs is to use a table look-up modelling scheme. This approach was first adopted in the timing simulator MOTIS [ChGK75] and it is also used in this work.

The proposed equations, for the current-voltage transistor characteristics, are incorporated into FACTS using the table look-up scheme described in [ChGK75], [SSMY82] and [Shim86]. Briefly, the equations are used to produce a set of tables of values of the drain current for various bias conditions and an interpolation technique is then used to provide the necessary drain current values during a simulation run. A number of interpolation schemes involving the Hermite polynomials in one and two dimensions [Ahli63], [Ferg64], [Forr72], [ShYD83], [Shim86], [ShDa86] have been investigated and compared to decide on the eventual scheme used.

The main advantage of a table model, apart from increasing the computational efficiency of a circuit analysis program, is that its computational cost is largely independent of the computational costs of the expressions used to generate the tables. Therefore, it becomes possible to use analytical models with higher complexity, and thus better accuracy, and even two-dimensional numerical device simulation models to characterize MOS transistors for circuit simulation purposes. As a result, main effort was placed on model accuracy, rather than simplicity, in developing the analytical model presented in this chapter.

This chapter presents the proposed model and discusses the main criteria used in its

formulation. The accuracy of the expressions are demonstrated by comparison with experimental results. The table look-up scheme is outlined and finally the target fabrication process for the systems designed in this work is described.

4.2 Proposed Model Equations

Based on the conclusions from the comparative study of the various models for the short and narrow channel effects a more complete current-voltage characteristics analytical model is proposed.

4.2.1 Threshold Voltage

As discussed in the introduction of this chapter, none of the short channel effect models have been found to be sufficiently accurate for the transistors considered in this work. It became necessary to propose a new short channel effect model, which could possibly model the threshold voltage of MOS transistors with submicrometer mask dimensions. Any short channel threshold voltage model based on the a priori charge sharing assumption (Section 3.3.1.1) fails to predict the effect of the surface potential distribution along the channel on the threshold voltage [OmOh79], [ToAs79], [RaMe82]. This effect becomes dominant for devices with channel lengths in the submicron region, as shown in Figure 4.3, and therefore must be included in any threshold voltage model intended for such transistors.

Analytical threshold voltage models that include the surface potential influence are based on the a priori surface potential distribution, as described in Section 3.3.1.2. An example of such a model was given by Toyabe and Asai [ToAs79]. In order to obtain a closed form expression for the threshold voltage, the authors made a number of approximations, as described in Section 3.3.1.2. In particular, a uniform depletion depth along the channel was assumed. This is equivalent to assuming that the drain and source junction depths are negligible. The model proposed in [ToAs79] is extended in this work to include the effect of finite junction depths on the threshold voltage, by incorporating the charge sharing concept into the expression given in [ToAs79]. The resulting expression is then modified to include the narrow width effect.

Expression (3.134) derived by [ToAs79] may be manipulated as follows, assuming $V_{DS} = 0V$:

$$\eta_0 = \frac{\epsilon_{ox}}{\epsilon_s}\frac{(V_{bi} - 2\phi_F)}{(2\phi_F + V_{SB})}\left(\frac{1}{t_{ox}}W_C + \frac{3}{2}\frac{\epsilon_s}{\epsilon_{ox}}\right),$$

$$= \left(1+\frac{3}{2}\frac{\epsilon_s}{W_C}\frac{t_{ox}}{\epsilon_{ox}}\right)\frac{W_C}{t_{ox}}\frac{\epsilon_{ox}}{\epsilon_s}\frac{(V_{bi}-2\phi_F)}{(2\phi_F+V_{SB})},$$
$$= \left(1+\frac{3}{2}\frac{C_B}{C_{ox}}\right)\frac{W_C}{t_{ox}}\frac{\epsilon_{ox}}{\epsilon_s}\frac{(V_{bi}-2\phi_F)}{(2\phi_F+V_{SB})}. \tag{4.1}$$

Expression (3.132) for the threshold voltage of a short channel MOS transistors may then be written as:

$$V_{TS} = V_{FB} + 2\phi_F - \frac{Q_B}{C_{ox}} - \left(1+\frac{3}{2}\frac{C_B}{C_{ox}}\right)\eta_v, \tag{4.2}$$

where C_B is the bulk capacitance per unit area given by (4.3), the Brown factor (B) used in the model is set to 2 and the quantity η_v is related to the parameter η of the original model of [ToAs79] and for no drain bias is given by (4.4):

$$C_B = \frac{\epsilon_s}{W_C}, \tag{4.3}$$
$$\eta_v = 2(V_{bi}-2\phi_F)\exp(-L/l_o), \tag{4.4}$$

where the expression (3.135) for l_o is written as:

$$l_o = W_C\left(\frac{3}{2}+\frac{C_{ox}}{C_B}\right)^{-1/2}. \tag{4.5}$$

In order to relax the assumption of negligible junction depths made by Toyabe and Asai [ToAs79], the two quantities, Q_B and C_B, which are affected by the existence of the source and drain junctions need to be modified. These two quantities are related to each other by:

$$C_B = \left|\frac{dQ_B}{d\psi_S}\right|. \tag{4.6}$$

Using (4.6) it is possible to introduce the a priori charge sharing assumption into the analysis of [ToAs79] which is based on the a priori surface distribution assumption. The *new* threshold voltage model for a small geometry transistor becomes:

$$V_{th} = V_{FB} + 2\phi_F - \frac{Q_{Be}}{C_{ox}} - \left(1+\frac{3}{2}\frac{C_{Be}}{C_{ox}}\right)\eta_v, \tag{4.7}$$

where Q_{Be} and C_{Be} are (for a small geometry transistor) the *effective* depletion charge and the *effective* depletion capacitance per unit area, respectively, obtained by applying the a priori charge sharing assumption.

In the opinion of the author, in many cases the complicated procedures based on the a priori charge sharing assumption (studied in Chapter 3) used for the derivation of the short channel effect models produce only an illusion of sophistication and accuracy (as demonstrated by Figures 4.3, 4.4 and 4.5). The difficult problem of determining the effects of the complicated two-dimensional electric field distribution in a short channel

transistor cannot be solved by assuming a simple geometrical shape for the region of the depletion charge that is supported by the gate charge, no matter how imaginative that shape may be. Thus, an *empirical* approach *guided* by the a priori charge sharing assumption is taken in this work.

The effective value of the bulk charge for a short channel transistor is found using an approach proposed by Yau [Yau74], ie the depletion region under the gate is assumed to be a trapezoid as shown in Figure 3.4, however, in contrast to Yau's model the length y_1 is treated as an empirical parameter, assumed constant for a given fabrication process. This quantity may be related to other process parameters using expression (3.76) proposed in [Dang79]:

$$y_{S,p}/r_j = (1 + y_{S0}/r_j)\left[\ln(1 + y_{S0}/r_j) - 0.5 + 0.5(1 + y_{S0}/r_j)^{-2}\right]^{1/2}, \tag{4.8}$$

where $y_{S0} = y_1(V_{SB} = 0)$, and where $y_{S,p} = \sqrt{2\epsilon_s/(qN_A)(V_{bi} - 2\phi_F)}$ is the depletion width estimated from the one sided *plane* junction theory and $V_{bi} - 2\phi_F$ is the potential drop across the *pn* junction.

The geometry factor for the short channel effect under the conditions $V_{SB} = V_{DS} = 0V$ is then given by:

$$\gamma_S = 1 - \frac{y_1}{L}. \tag{4.9}$$

When a substrate bias is applied the increase in the length y_1 with V_{SB} is approximated by the following empirical expression:

$$y_1 = y_{S0}\left(1 + \sigma V_{SB}^{1/2}\right), \tag{4.10}$$

where σ is an empirical parameter (with dimensions $V^{-1/2}$) which also allows to model the effect of channel implantation.

The narrow width effect, measured experimentally in this work, has been found to be well modelled using the approach proposed by Akers [Aker81b], ie expression (3.150). However, a narrow width effect dimensionless parameter (δ_W) is included, ie the extra depletion region charge terminated by the gate charge is assumed to be contained in two volumes, one on either side of the gate electrode, with each volume having a rectangular width cross-section of depth W_C and width $\delta_W W_{C0}$:

$$\Delta V_{TN} = \frac{qN_A W_C}{C_{ox}}\left(2\frac{\delta_W W_{C0}}{W}\right). \tag{4.11}$$

For a small geometry MOS transistor, the total volume including the effective depletion region charge is obtained by integrating the length cross-section of the depletion region

over the width direction, as suggested in [Aker81b]. Thus, the small geometry effect factor is given by:

$$\gamma = \left[1 - \frac{y_{S0}}{L}\left(1 + \sigma V_{SB}^{1/2}\right)\right]\left(1 + \frac{2\delta_W W_{C0}}{W}\right). \tag{4.12}$$

Under the conditions of no drain bias the effective depletion region charge per unit area, for a small geometry MOS transistor, is given by:

$$Q_{Be} = \gamma Q_B. \tag{4.13}$$

The effective depletion region capacitance per unit area is found using relationship (4.6) evaluated at $\psi_S = 2\phi_F$:

$$\begin{aligned} C_{Be} &= \left|\frac{dQ_{Be}}{d\psi_S}\right|, \\ &= \gamma\left|\frac{dQ_B}{d\psi_S}\right| + |Q_B|\frac{d\gamma}{d\psi_S}. \end{aligned} \tag{4.14}$$

Using expression (3.19) for Q_B and expression (4.12) for γ the various terms in expression (4.14) can be evaluated as follows:

$$\left|\frac{dQ_B}{d\psi_S}\right| = \frac{2\epsilon_s q N_A}{|Q_B|}, \tag{4.15}$$

$$\frac{d\gamma}{d\psi_S} = \frac{2\delta_W}{W}\frac{dW_{C0}}{d\psi_S} - \frac{(1+\sigma V_{SB}^{1/2})}{L}\frac{dy_{S0}}{d\psi_S} - \frac{2\delta_W(1+\sigma V_{SB}^{1/2})}{WL}\left(W_{C0}\frac{dy_{S0}}{d\psi_S} + y_{S0}\frac{dW_{C0}}{d\psi_S}\right), \tag{4.16}$$

where:

$$\frac{dy_{S0}}{d\psi_S} = \frac{\epsilon_s}{qN_A r_j(1 + y_{S0}/r_j)\ln(1 + y_{S0}/r_j)}, \tag{4.17}$$

$$\frac{dW_{C0}}{d\psi_S} = \frac{2\epsilon_s/(qN_A)}{W_{C0}}. \tag{4.18}$$

Finally, the drain induced barrier lowering effect [Trou79] is assumed to be linear with V_{DS} [TrFo77], [MaNM79], [KlGr80], [LeGD81]:

$$\Delta V_{th}(V_{DS}) = -\alpha V_{DS}, \tag{4.19}$$

where α is a dimensionless empirical parameter depending on the channel length [Klass78], [KlGr80], and it is given by $\alpha = \alpha_0/L^{-1.5}$. The complete small geometry threshold voltage model is given by:

$$V_{th} = V_{FB} + 2\phi_F + \gamma Q_B - \left(1 + \frac{3}{2}\frac{C_{Be}}{C_{ox}}\right)\eta_v - \alpha V_{DS}, \tag{4.20}$$

where γ is given by (4.12), C_{Be} is given by (4.14) and η_v is given by (4.4). This threshold voltage model is composed of three parts. The geometry factor γ represents the small geometry effect due to the charge sharing, η_v represents the influence of the surface potential distribution along the channel and the product $\left(\frac{3}{2}\frac{C_{Be}}{C_{ox}}\right)\eta_v$ represents the coupling between the two terms.

The contribution of the term depending on η_v in expression (4.20) is negligible for the device dimensions used in this work (eg $\eta_v \approx 10^{-7}$, for a P1 (P1 is defined later) n-channel transistor with $L = 2.9 \times 10^{-6}m$) and instead a simplified form of expression (4.20), with the surface potential term omitted, is used:

$$V_{th} = V_{FB} + 2\phi_F + \frac{\gamma}{C_{ox}}\sqrt{2\epsilon_s q N_A(2\phi_F + V_{SB} - kT/q)} - \alpha V_{DS}, \tag{4.21}$$

where γ is still given by (4.12). The empirical parameters associated with the threshold voltage model are δ_W, σ, and α. The quantities V_{FB}, N_A and ϕ_F which are related to fabrication process characteristics are also treated as adjustable model parameters whose values are determined using curve fitting techniques, but whose values should be close to the values predicted theoretically.

4.2.2 Carrier Mobility

The dependence of carrier mobility (μ) on the normal electric field ($E_\perp$) and on the lateral electric field ($E_{||}$) is approximated by an expression analogous to (3.196), which has been used in [MeBC72], [Dang79], [WhWL80] and [HaCD82]:

$$\mu = \frac{\mu^o}{1 + \eta_\theta |E_\perp| + \eta \left|E_{||}\right|}, \tag{4.22}$$

where μ^o is the low field mobility and η_θ and η are two mobility modifying parameters, both with dimensions mV^{-1}. The mobility dependence with temperature, an effect neglected by all models which have been studied, can be modelled with a simple power law expression proposed in [DoLe81]:

$$\mu^o = \mu^{oo}\left(\frac{T}{300}\right)^{-2.2}. \tag{4.23}$$

The effective perpendicular field in the channel is approximated by the following expression, which is analogous to (3.175) and which includes the small geometry effect:

$$\begin{aligned} |E_\perp| &= \frac{1}{\epsilon_s}\left(\frac{1}{2}|Q_n| + |Q_{Be}|\right), \\ &= \frac{C_{ox}}{2\epsilon_s}\left[V_{GS} - V_{FB} - 2\phi_F - V + \frac{\gamma}{C_{ox}}\sqrt{2\epsilon_s q N_A(2\phi_F + V_{SB} + V - kT/q)} - \alpha V_{DS}\right], \end{aligned} \tag{4.24}$$

where V is the channel voltage at a point y taken with respect to the source voltage, the effective depletion region charge per unit area (Q_{Be}) is given by (4.29) and the channel mobile charge per unit area (Q_n) is given by (4.28). The magnitude of lateral electric field is given by the usual expression:

$$\left|E_{\|}\right| = \frac{dV}{dy}. \tag{4.25}$$

Combining (4.22), (4.24) and (4.25), and setting $\theta = \eta_\theta C_{ox}/(2\epsilon_s)$ results in the following expression for the field dependent mobility:

$$\begin{aligned} \mu &= \mu^o / \{1 + \theta [V_{GS} - V_{FB} - 2\phi_F - V \\ &+ \frac{\gamma}{C_{ox}} \sqrt{2\epsilon_s q N_A (2\phi_F + V_{SB} + V - kT/q)} - \alpha V_{DS}] + \eta \frac{dV}{dy} \}. \end{aligned} \tag{4.26}$$

The empirical parameters associated with the mobility model are θ (dimensions V^{-1}) and η. The small geometry effects are included in the parameter γ and by including the term αV_{DS}. The parameter θ, given by (3.203), also includes the effects of the finite junction resistances. However, expression (3.203) assumes the presence of a fixed contact resistance R_c. Normally for wide channel transistors larger than minimum size contact cuts are used at the source and drain terminals to decrease metal-diffusion contact resistance and to minimize current crowding around the contacts. If R_c is taken to be the resistance of a minimum metal-diffusion contact and if the size of the drain or source contact is expressed as an equivalent number (n_c) of the minimum size contacts then expression (3.203) can be written as:

$$\theta = \theta_0 + \theta \frac{W}{L} \frac{1}{n_c} + \theta_2 \frac{1}{L}. \tag{4.27}$$

The complete mobility model incorporates the effect of the normal electric field, velocity saturation, small geometry effect and the finite junction resistance, including variable size drain and source contacts. In addition, it predicts a decrease in carrier mobility with an increase in the substrate bias, as reported in [WhWL80] and confirmed in this work (Figures 4.8 and 4.9).

4.2.3 Linear Region of Operation

In order to obtain a closed form expression for the drain current the usual approach, as explained in Section 3.3.6, is taken. Assuming that the gradual channel approximation applies in the linear region of operation, the charge conservation principle gives:

$$C_{ox}(V_{GS} - V_{FB} - \psi_S) = -(Q_n + Q_{Be}), \tag{4.28}$$

where the effective depletion charge per unit area at a given substrate and drain voltages is given by:

$$Q_{Be} = \gamma Q_B + C_{ox}\alpha V_{DS}. \tag{4.29}$$

It is further assumed that the drain voltage dependent term in the threshold voltage represents an average drain voltage influence on the channel potential. Proceeding in an analogous manner to the analysis given by Brews [Brew78], described in Section 3.2.2 and including the terms representing the small geometry effect gives the following differential equation, analogous to expression (3.18), for the drain current:

$$I_{DS}dy = \mu W C_{ox}\left(V_{GS} - V_{FB} - \psi_S + \frac{kT}{q} + \frac{\gamma}{C_{ox}}Q_B + \alpha V_{DS}\right)d\psi_S - \gamma\frac{kT}{q}dQ_B. \tag{4.30}$$

The surface potential along the channel at strong inversion is approximated by:

$$\psi_S = 2\phi_F + V. \tag{4.31}$$

Introducing the mobility expression (4.26) and rearranging yields:

$$\begin{aligned} & I_{DS}\left\{\int_0^L\left[1+\theta\left(V_{GS}-V_{FB}-2\phi_F-V-\frac{\gamma}{C_{ox}}Q_B-\alpha V_{DS}\right)\right]dy+\eta\int_0^{V_{DS}}dV\right\} \\ = \; & \mu^o W C_{ox}\left\{\int_0^{V_{DS}}\left[\left(V_{GS}-V_{FB}-2\phi_F-V+\frac{kT}{q}+\frac{\gamma}{C_{ox}}Q_B+\alpha V_{DS}\right)\right]dV\right. \\ - \; & \left.\frac{\gamma}{C_{ox}}\int_{Q_B(0)}^{Q_B(V_{DS})}dQ_B\right\}. \end{aligned} \tag{4.32}$$

The first integral on the left hand side of expression (4.32) is evaluated with the use of the assumption that the longitudinal electric field along the channel is constant [WhWL80], ie $dV/dy = V_{DS}/L$. The final expression for the drain current reads:

$$\begin{aligned} I_{DS} = \; & \mu\frac{W}{L}C_{ox}\left\{(V_{GS}-V_{FB}-2\phi_F+kT/q+\alpha V_{DS})V_{DS}-\frac{1}{2}V_{DS}^2\right. \\ - \; & \gamma\frac{2}{3}\frac{\sqrt{2\epsilon_s qN_A}}{C_{ox}}\left[(V_{DS}+2\phi_F+V_{SB}-kT/q)^{3/2}-(2\phi_F+V_{SB}-kT/q)^{3/2}\right] \\ + \; & \left.\gamma\frac{kT}{q}\frac{\sqrt{2\epsilon_s qN_A}}{C_{ox}}\left[(V_{DS}+2\phi_F+V_{SB}-kT/q)^{1/2}-(2\phi_F+V_{SB}-kT/q)^{1/2}\right]\right\} \end{aligned} \tag{4.33}$$

with the field dependent mobility given by:

$$\begin{aligned} \mu = \; & \mu^o/\left\{1+\frac{\theta}{V_{DS}}\left\{(V_{GS}-V_{FB}-2\phi_F-\alpha V_{DS})V_{DS}-\frac{1}{2}V_{DS}^2+\gamma\frac{2}{3}\frac{\sqrt{2\epsilon_s qN_A}}{C_{ox}}\right.\right. \\ \times \; & \left.\left.\left[(V_{DS}+2\phi_F-kT/q+V_{SB})^{3/2}-(2\phi_F-kT/q+V_{SB})^{3/2}\right]\right\}+\eta\frac{V_{DS}}{L}\right\}. \end{aligned} \tag{4.34}$$

Expression (4.33) has the general form of the large geometry formula (3.20) proposed in [Brew78] and based on the charge sheet model. It includes the small geometry effect

through the quantities γ and α, the velocity saturation effect and the carrier mobility dependence on the normal electric field through expression (4.34), the effect of finite junction resistance through expression (4.27) for θ, and it also includes the diffusion current terms which become significant near threshold and saturation.

It is possible, as demonstrated in Sections 3.2.3 and 3.2.4, to simplify (4.33) to forms equivalent to (3.24) and (3.28). Thus, the accuracy of the model can be varied according to the needs. In particular, by using the numerical approximation (3.226) to the body effect term, expression (4.33) may be written in a form analogous to (3.230).

4.2.4 Saturation Voltage

Expressions (3.236), (3.239), (3.241) and (3.243) for the saturation voltage given by [MeBC72], [YaMo83], [KlGr80] and [IMGB82], respectively are compared for a P1 n-channel transistor (model parameters are given in Table 4.1) in Figure 4.1.

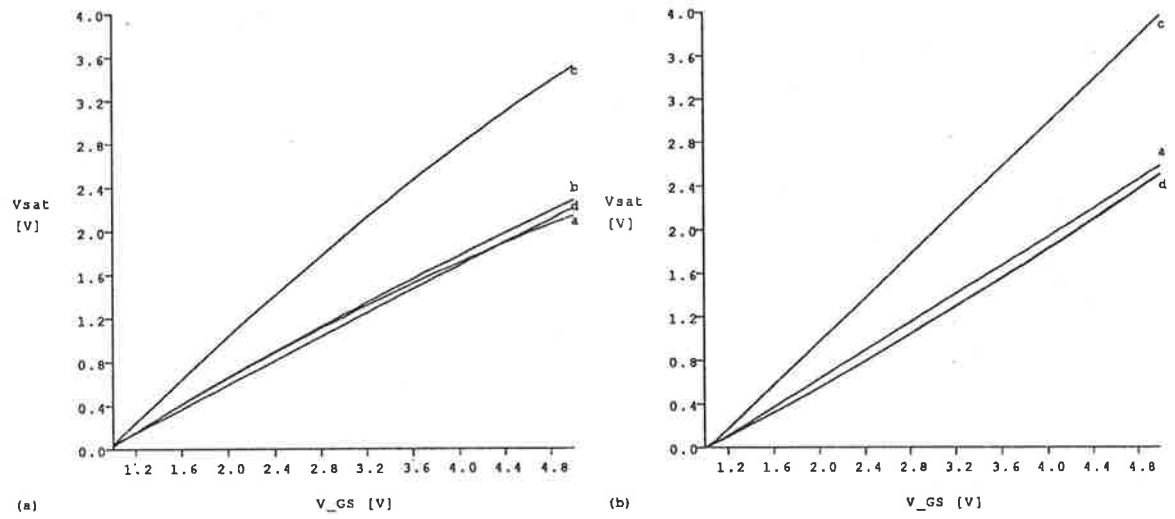


Figure 4.1: V_{sat} versus V_{GS} according to the models of a-[MeBC72], b-[KlGr80], c-[IMGB82], d-[YaMo83] calculated for a P1 n-channel transistor with: (a) $L = 2.9 \times 10^{-6}m$, $W = 50 \times 10^{-6}m$, (b) $L = 222.9 \times 10^{-6}m$, $W = 50 \times 10^{-6}m$.

In all cases the threshold voltage and the drain currents are calculated according to the models proposed in this chapter. The models due to [MeBC72], [KlGr80] and [YaMo83] differ only by a few percent. For the short channel transistor the model proposed in [MeBC72] experiences an interesting behaviour of following model of [KlGr80] for low gate voltages and being closer to the model of [YaMo83] for larger values of V_{GS}. The maximum difference, in this voltage range, between the three models is $\approx 5\%$.

For a long channel transistor, the models due to [MeBC72] and [KlGr80] are exactly the same and about 3% larger than the model due to [YaMo83], Figure 4.1(b). The model due to [YaMo83] is within 0.1% of the long channel threshold voltage model given by (3.36). However, by treating E_{cy} and v_{sat} as empirical parameters the three models can be made to agree with each other to within 0.5%.

The approach proposed in [YaMo83] is used in this work, in which the surface electron density at the velocity saturation point is calculated from the drain current expression (4.33) at $V_{DS} = V_{sat}$ and is also calculated by the charge neutrality condition at the channel potential $V = V_{sat}$, ie:

$$\begin{aligned} \frac{I_{sat}}{Wqv_{sat}} &= \frac{C_{ox}}{q}\{V_{GS} - V_{FB} - 2\phi_F + kT/q - V_{sat} \\ &- \frac{\gamma}{C_{ox}}[2\epsilon_s qN_A(v_{sat} + 2\phi_F - kT/q + V_{SB})]^{1/2} + \alpha V_{DS}\Big\}, \end{aligned} \tag{4.35}$$

where I_{sat} is given by expressions (4.33) and (4.34) with $V_{DS} = V_{sat}$. Expression (4.35) is solved for V_{sat} by iteration. Normally two or three iterations of the Newton-Raphson method are sufficient for the solution to be correct to eight significant figures with the saturation voltage given by the long channel MOSFET theory used as the starting value.

4.2.5 Saturation Region of Operation

The finite output conductance in saturation is modelled by the usual channel length modulation approach:

$$I_{DS} = \frac{I_{sat}}{1 - \frac{\Delta L}{L}}, \tag{4.36}$$

where I_{sat} is the drain current evaluated at V_{sat}.

Expressions (3.251), (3.258), (3.262), (3.267) and (3.265) for the channel shortening quantity ΔL proposed by [ReSa65], [BaBe70], [MeBC72], [KlGr80] and [RoMV76], respectively, are compared in Figure 4.2. Model specific parameters have been assigned values suggested by the respective authors, eg in expression (3.262) the channel thickness $d = 100 \times 10^{-9}m$, and in expression (3.267) the quantity α_{sat} is calculated from $[\epsilon_s/(qN_AL^2)]^{1/2}$. The other parameters are given in Table 4.1.

All models, with the exception of the expression given by [ReSa65] have one or more quantities which can be treated as adjustable parameters, and by suitable choosing their values model results can be made to agree with one another. It would be desirable, however, to have the same values for the parameters E_{cy} and v_{sat} in the models for the saturation voltage and the channel modulation effect, since these parameters represent the same physical quantities in both cases.

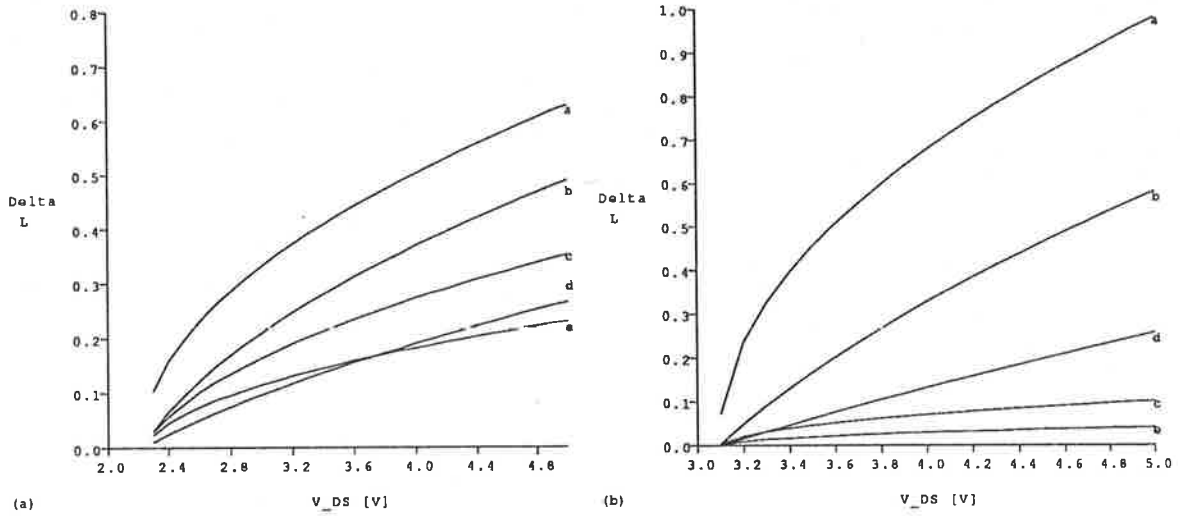


Figure 4.2: ΔL versus V_{DS} according to the models of a-[ReSa65], b-[BaBe70], c-[MeBC72], d-[KlGr80], e-[RoMV76] for an n-channel device: (a) P1 process, $L = 2.9 \times 10^{-6}m$, $W = 50 \times 10^{-6}m$, (b) P2 process, $L = 1.7 \times 10^{-6}m$, $W = 50 \times 10^{-6}m$. The units of ΔL are $10^{-6}m$.

The expression derived by [RoMV76], ie (3.265), which is based on the analysis of [MeBC72] has been widely used, for example [Dang77b], [Dang79], [FuOk80], [LeGD81], and for this reason is also adopted in this work:

$$\Delta L = \frac{\left\{E_{cy}^2 + \frac{2}{\epsilon_s}\left(qN_A + \frac{I_{DS}}{W\eta_1 r_j v_{sat}}\right)(V_{DS} - V_{sat})\right\}^{1/2} - E_{cy}}{\frac{1}{\epsilon_s}\left(qN_A + \frac{I_{DS}}{W\eta_1 r_j v_{sat}}\right)}. \tag{4.37}$$

4.3 Parameter Extraction

As discussed previously, in the process of deriving analytical models a number of simplifying assumptions are made, which are sometimes difficult to justify rigorously, in order to arrive at relatively simple expressions. For example, in such models it is assumed that the surface potential at strong inversion is "pinned" to $2\phi_F$ ($V_{SB} = 0$). However, the surface potential is never exactly pinned to a single value but instead it varies gradually with bias (see for example [Sze81]). In fact, parameters which can be specified by a single theoretical value, eg the body effect factor K, are actually specified exactly only for the theoretical transistors corresponding to the simplifying assumptions that are made, eg that the substrate doping concentration is exactly uniform. Consequently, when analytical models are used to represent real devices, the parameter values which will result in the minimum error between model predictions and experimental values are not necessarily dictated by simple theory. Thus, model parameters are usually deter-

Parameter name	Units	P1 n-channel	P2 n-channel	P1 p-channel
μ^o	Vm^{-2}	0.075	0.070	0.0345
t_{ox}	m	970×10^{-10}	370×10^{-10}	970×10^{-10}
N_A	m^{-3}	9.10×10^{21}	2.5×10^{21}	1.11×10^{21}
r_j	m	1.85×10^{-6}	0.35×10^{-6}	1.70×10^{-6}
ΔL_m	m	2.1×10^{-6}	0.8×10^{-6}	0.6×10^{-6}
ΔW_m	m	1.3×10^{-6}	0.5×10^{-6}	1.3×10^{-6}
θ_0	V^{-1}	0.0013	0.023	0.050
θ_1	V^{-1}	0.0029	0.005	0.006
η	mV^{-1}	0.55	0.88	0.50
η_1		0.03	0.03	0.10
E_{cy}	Vm^{-1}	2.20×10^6	2.05×10^6	2.5×10^6
δ		0.55	0.50	0.65
σ	$V^{-1/2}$	0.05	0.00	0.030
α_0		0.10	0.18	0.20
V_{FB}	V	-1.0	-0.38	0.25
V_{bi}	V	0.85	0.83	0.75
V_{th}(large)	V	1.05	0.52	-0.78

Table 4.1: Model parameters ($T = 300K$).

mined empirically by curve fitting techniques to provide best agreement between model predictions and measurements. In this process of parameter extraction the theoretical parameter values may be used as initial estimates which are then improved upon.

The parameters for the model proposed in this chapter were extracted manually according to techniques published in [Klas78], [AnSm79], [HaCD82], [MoKS82], [Hask83] and [GaSo87]. The procedures used are described in detail in Appendix B of this thesis.

4.4 Comparison with Experiment

In order to verify the analytical model presented herein and to study the validity of the analytical models reviewed in Chapter 3, experimental measurements were carried out, using a computer controlled data acquisition system and a Tektronics Curve Tracer System, on a number of MOS transistors from two different fabrication processes.

The first set of test transistors consisted of a group of n-channel and p-channel devices fabricated using a CMOS, oxide isolated, p-well process (denoted here the P1 process) having a minimum feature size of 5-micrometers, ie $\lambda = 2.5 \times 10^{-6}m$. The initial material was a 5 ohm-cm, n-type substrate, oxide thickness was $970 \times 10^{-10}m$, the drain and source junction depths were in the range $1.7 \times 10^{-6}m$ to $1.9 \times 10^{-6}m$ for

the n-channel transistors and $1.6 \times 10^{-6}m$ to $1.8 \times 10^{-6}m$ for the p-channel transistors. A single boron implant was used in the p-channel transistors to adjust the threshold voltage to below $1.4V$.

Groups of 12 transistors of each type with different mask dimensions were available, with the transistors being located in 4 different places on a wafer. Two wafers from the same batch were available. The electrical channel lengths ranged from $22.9 \times 10^{-6}m$ to $2.9 \times 10^{-6}m$ for the n-channel transistors and from $24.4 \times 10^{-6}m$ to $4.4 \times 10^{-6}m$ for the p-channel transistors. The electrical channel widths ranged from $23.7 \times 10^{-6}m$ to $3.3 \times 10^{-6}m$ for both types of transistors.

The second set of devices was fabricated using a 2.5-micrometer, HMOS process (denoted here the P2 process). Groups of 10 enhancement n-channel transistors with different mask dimensions from 4 wafers were available. The electrical channel lengths ranged from $29.2 \times 10^{-6}m$ to $1.7 \times 10^{-6}m$ and the electrical widths ranged from $29.5 \times 10^{-6}m$ to $4.5 \times 10^{-6}m$. The oxide thickness was $370 \times 10^{-10}m$ and the junction depths were approximately $0.3 \times 10^{-6}m$.

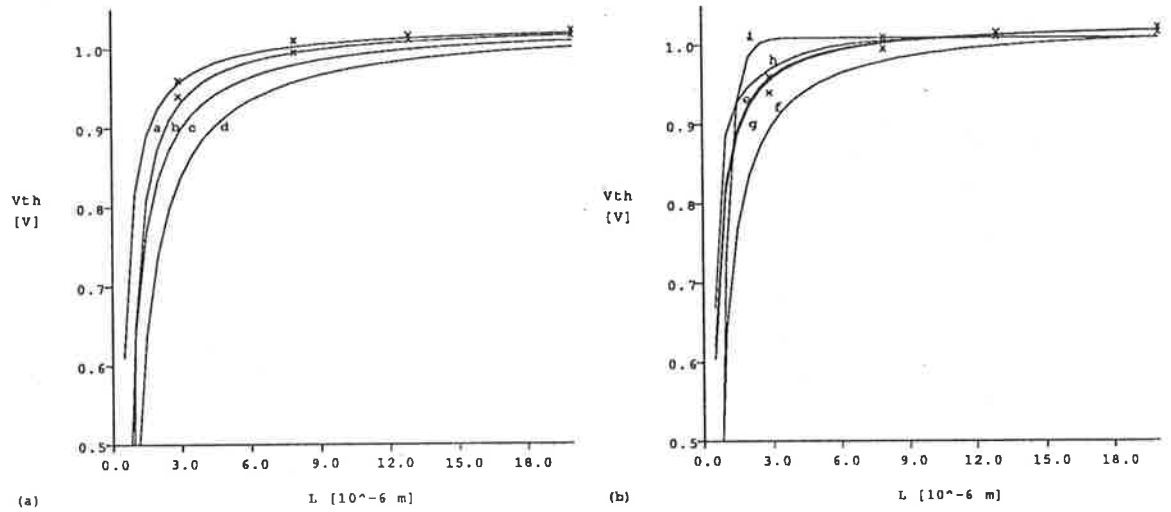


Figure 4.3: V_{th} versus L (n-channel P1 process transistors $W = 23.7 \times 10^{-6}m$) according to the models of: (a) a-this work, b-[Tayl78], c-[Yau74], d-[OmOh79], (b) e-[FiPo79], f-[Dang79], g-[Lee73], h-[ToAs79], i-[RaMe82].

The measurements were performed on at least six transistors of each type and size on a wafer. In this way a range of experimental results, representing the process variations across the wafer and from wafer to wafer, was obtained. The range of experimental results is indicated by crosses in all figures. All measurements were carried out at room temperature, $T = 300K$. The model parameters are listed in Table 4.1 for both fabrication processes. The values of the parameters t_{ox} and r_j were supplied by the

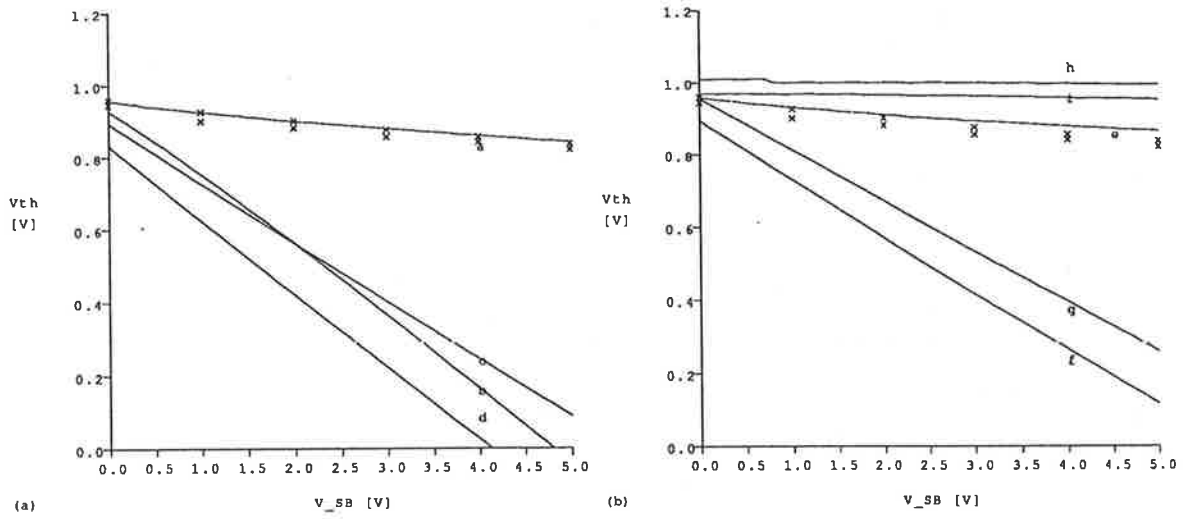


Figure 4.4: V_{th} versus V_{SB} (n-channel P1 process transistor $L = 22.9 \times 10^{-6}m$, $W = 23.7 \times 10^{-6}m$) according to the models of: (a) a-this work, b-[Tayl78], c-[Yau74], d-[OmOh79], (b) e-[FiPo79], f-[Dang79], g-[Lee73], i-[RaMe82].

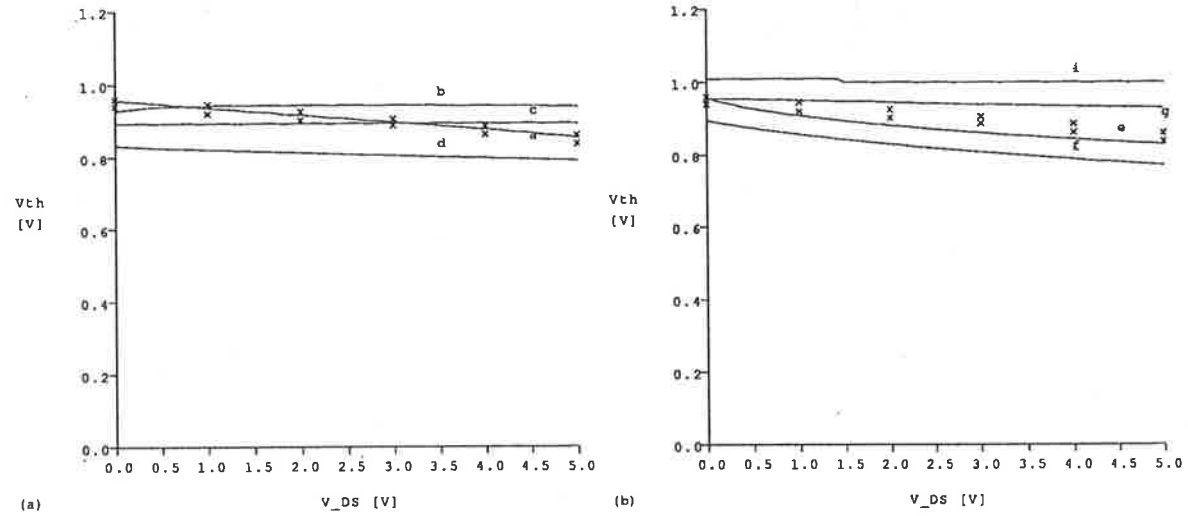


Figure 4.5: V_{th} versus V_{DS} (n-channel P1 process transistor $L = 22.9 \times 10^{-6}m$, $W = 23.7 \times 10^{-6}m$) according to the models of: (a) a-this work, b-[Tayl78], c-[Yau74], d-[OmOh79], (b) e-[FiPo79], f-[Dang79], g-[Lee73], i-[RaMe82].

industrial firms. The accuracy of the analytical model presented here is demonstrated by comparing the predictions obtained using this model with the experimental results for the threshold voltage and current-voltage transistor characteristics. The small geometry effects were most prominent for the n-channel transistors fabricated with the P1 process, owing to the deep junctions and the thick gate oxide. The comparison is carried out using mainly the experimental results for these transistors, as this represents the more

demanding test.

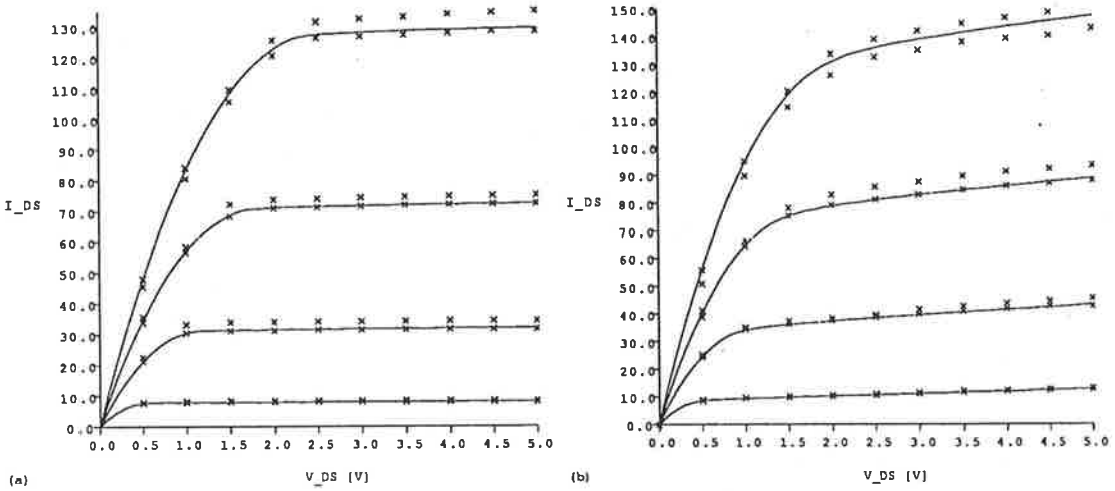


Figure 4.6: I_{DS} versus V_{DS}: (a) n-channel, P1 process, $W = 23.7 \times 10^{-6}m$, $L = 22.9 \times 10^{-6}m$, $V_{SB} = 0V$, (b) n-channel, P1 process, $W = 3.7 \times 10^{-6}m$, $L = 2.9 \times 10^{-6}m$, $V_{SB} = 0V$. V_{GS} ranges from 2V to 5V in steps of 1V. I_{DS} is in units of 10^{-6}A.

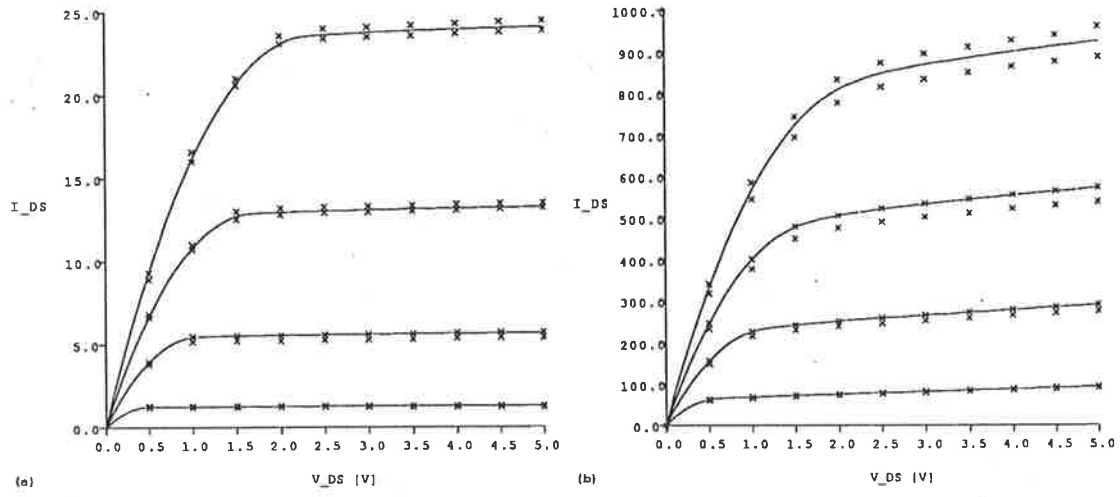


Figure 4.7: I_{DS} versus V_{DS}: (a) n-channel, P1 process, $W = 3.7 \times 10^{-6}m$, $L = 17.9 \times 10^{-6}m$, $V_{SB} = 0V$, (b) n-channel, P1 process, $W = 23.7 \times 10^{-6}m$, $L = 2.9 \times 10^{-6}m$, $V_{SB} = 0V$. V_{GS} ranges from 2V to 5V in steps of 1V. I_{DS} is in units of 10^{-6} A.

It is interesting to compare the large number of different analytical expressions for the short channel effect which have been published in the literature and reviewed in Chapter 3. The threshold voltage, as a function of channel length, according to a

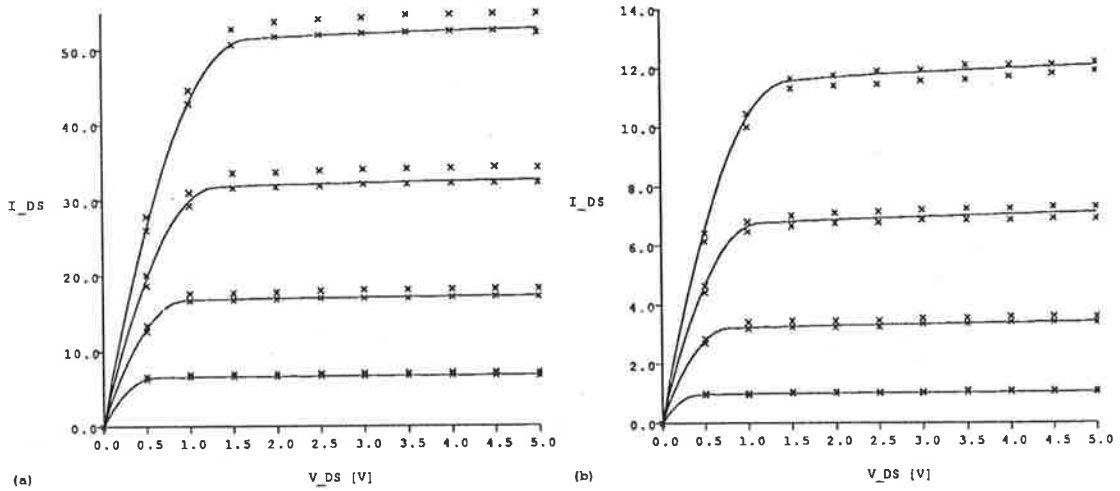


Figure 4.8: I_{DS} versus V_{DS}: (a) n-channel, P1 process, $W = 23.7 \times 10^{-6}m$, $L = 22.9 \times 10^{-6}m$, $V_{SB} = 3V$, (b) n-channel, P1 process, $W = 3.7 \times 10^{-6}m$ $L = 12.9 \times 10^{-6}m$, $V_{SB} = 3V$. V_{GS} ranges from 3.5V to 5V in steps of 0.5V. I_{DS} is in units of 10^{-6} A.

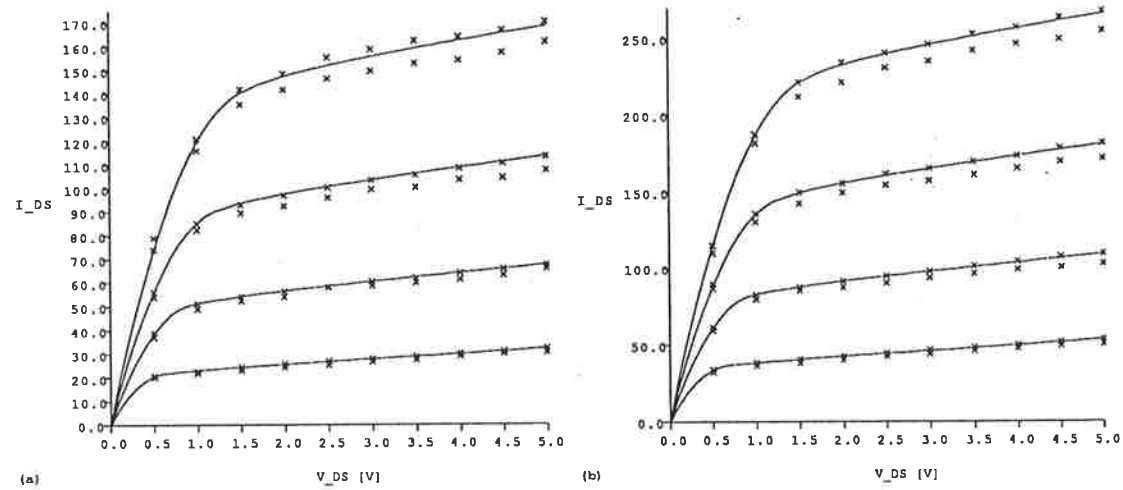


Figure 4.9: I_{DS} versus V_{DS}: (a) n-channel, P1 process, $W = 8.7 \times 10^{-6}m$, $L = 2.9 \times 10^{-6}m$, $V_{SB} = 3V$, (b) n-channel, P1 process, $W = 13.7 \times 10^{-6}m$, $L = 2.9 \times 10^{-6}m$, $V_{SB} = 3V$. V_{GS} ranges from 3.5V to 5V in steps of 0.5V. I_{DS} is in units of 10^{-6} A.

number of models studied in Chapter 3 and the threshold voltage model presented in this section are compared against the experimental results in Figure 4.3, for the n-channel transistors fabricated using the P1 process. The widths of the transistors were $W = 23.7 \times 10^{-6}m$. It was found that the narrow channel effect was also noticeable for this channel width. In order to take this effect into account in comparing the various

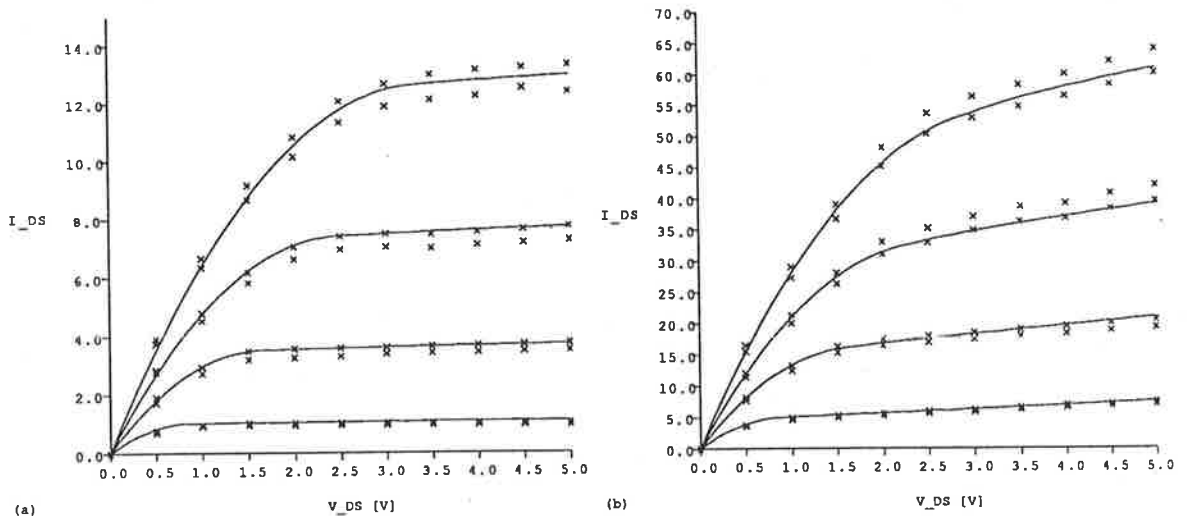


Figure 4.10: I_{DS} versus V_{DS}: (a) p-channel, P1 process, $W = 3.7 \times 10^{-6}m$, $L = 19.4 \times 10^{-6}m$, $V_{SB} = 0V$, (b) p-channel, P1 process, $W = 3.7 \times 10^{-6}m$, $L = 4.4 \times 10^{-6}m$, $V_{SB} = 0V$. V_{GS} ranges from -2V to -5V in steps of -1V. I_{DS} is in units of 10^{-6} A.

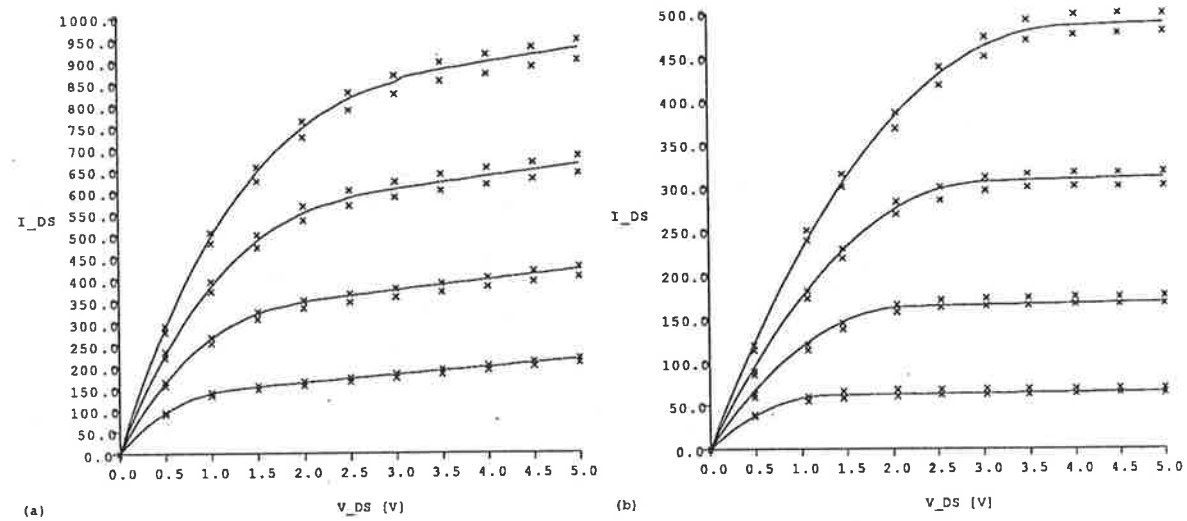


Figure 4.11: I_{DS} versus V_{DS}: (a) n-channel, P2 process, $W = 9.5 \times 10^{-6}m$, $L = 1.7 \times 10^{-6}m$, $V_{SB} = 0V$, (b) n-channel, P2 process, $W = 9.5 \times 10^{-6}m$, $L = 9.2 \times 10^{-6}m$, $V_{SB} = 0V$. V_{GS} ranges from 2V to 5V in steps of 1V. I_{DS} is in units of 10^{-6} A.

models, the corresponding equations were modified to include the narrow width effect model used in the threshold voltage model developed in this work.

It is seen that in all cases the predictions given by the expression proposed in this chapter for the short channel effect fall within the range of experimental results.

The series of figures 4.6-4.11 demonstrates the accuracy of the complete analytical model for the current-voltage transistor characteristics proposed in this chapter. Rather than optimizing the model for a particular set of test results, the range of experimental measurements is shown, as was done for the threshold voltage results. Again, good agreement with the experimental results is obtained in all cases. However, in some cases, especially for transistors with a large W and small L (Figure 4.11(a)), there is a small discontinuity in the drain current at the saturation voltage. This is the main disadvantage of analytical MOS transistors models [KlGr80], [HaCD82], [EnDM83], and it is a consequence of subdividing the channel, in the saturation region of operation, into two sections. This is, however, not a significant problem in simulating the behaviour of digital circuits, provided the numerical solution technique incorporated in the circuit simulation program being used does not require that the continuity of the first derivative of the drain current with respect to the drain voltage be preserved [JLLR73].

4.5 Modification of FACTS

The greatest advantage of the circuit analysis program, FACTS, is its fast response, even for circuits containing $\approx$ 10000 transistors. This is mainly due to the very simple expressions used to represent the transistor characteristics. In the original version of this program (version v0.1) the MOS transistor models were as follows:

$$\begin{aligned} I_{DS} &= \mu^{o}\frac{W}{L}C_{ox}\left[(V_{GS}-V_{th})V_{DS}-\frac{1}{2}V_{DS}^{2}\right], \quad V_{DS}\leq V_{sat}, \\ &= \mu^{o}\frac{W}{L}C_{ox}\frac{1}{2}(V_{GS}-V_{th})^{2}, \quad V_{DS}>V_{sat}. \end{aligned} \tag{4.38}$$

An additional reason for the fast response of FACTS is that it is not a conventional circuit simulation program like SPICE [Nage75], but rather it is a timing simulation program based on the time integration technique and time step advancement algorithms used in the timing simulation program MOTIS [ChGK75].

Clearly, the transistor models given by (4.38) are not even applicable for modelling transistors with moderate channel dimensions, eg $W, L \approx 10 \times 10^{-6}m$. This was the primary reason for developing the analytical MOS transistor model presented in this chapter. The major objective in improving the accuracy of FACTS was to preserve, as far as practicable, the fast response of the simulator. Thus, a fast calculation of transistor drain current for a given bias condition is of prime importance. Therefore, a direct replacement of (4.38) with the expressions presented in the previous sections of this chapter, or even with simpler expressions incorporating small geometry effects, will not achieve the desired objective.

One possible solution, which was adopted in the timing simulator MOTIS [ChGK75], is to use the analytical equations to produce a set of tables of data, prior to performing circuit simulation. These tables are then used during the actual simulation run to look-up the drain current values for a given bias condition, and an interpolation technique is employed to calculate the missing entries in the tables. This approach has also been adopted in this work. Since the problem of implementing a table model inside a timing simulation program has been previously dealt with in [ChGK75], [SSMY82], [ShYD83], [Shim86] and [ShDa86], only an outline of this task will be presented herein.

There are three closely related factors which need to be addressed when considering the performance of a table model: storage requirements, speed and accuracy. In order to restrict the number of table models that need to be generated, ie to a large extent limit the storage requirements, the total space of geometry, structure and temperature is generally quantized. In a typical digital integrated circuit there is only a limited number of transistors with different channel lengths. In fact, in all designs carried out in this work, only transistors with the minimum possible channel lengths have been used. Therefore, channel length quantization does not present any limitations. The simulations are generally performed under "worst case" conditions so that temperature quantization does not present a design limitation either. Transistor channel widths, on the other hand, vary over a wide range, as designers scale devices to meet delay requirements. Typically, however, channel width is either kept close to the minimum possible width to minimize silicon area, or in the case of a control line driver, for example, is made much larger than the minimum possible width. Thus, a degree of quantization in channel width is also possible in digital circuits.

The device width (W) not only affects the threshold voltage (V_{th}) but also the channel current (I_{DS}). As long as the effects of W on V_{th} and on θ are large, then it is not possible to represent the effects of different values of W by simply scaling the drain current. However, for W sufficiently large the narrow width effect is negligible. Furthermore, by adopting a design methodology where for larger values of W correspondingly larger drain and source contact cuts are used (ie n_c is increased to keep W/n_c constant), the value of the mobility degradation parameter θ, given by (4.27), can be kept approximately independent of W. Under such conditions drain current of wide channel transistors may be calculated by scaling drain current of some *reference* transistor.

In this work five different tables were found to be sufficient. Tables were constructed for transistors having minimum channel lengths and channel widths given by: the minimum, twice the minimum, four times the minimum, six times the minimum and having the reference width. The reference width was taken to be sixteen times the minimum

channel width. This was found to be sufficiently large for the narrow width effect to be negligible. Larger width transistors were subsequently modelled by scaling the values obtained from the table for the reference transistor. The table size chosen and the interpolation technique used are discussed next.

A simple linear interpolation method is faster than a higher order techniques but it requires a bigger table of values to achieve the same accuracy. In some cases the interpolation technique which must be used, in order for the the simulation program to converge, is dictated by the numerical requirements of the solution method employed. For example, if the nonlinear circuit equations are solved using a Newton-Raphson iterative method then it is necessary that the table model for the drain current be continuous, monotonic and smooth (continuity in the first derivative) along the direction of V_{DS} [JLLR73]. This makes the linear interpolation technique unsuitable in such a simulation program, since it does not provide continuous first derivative, and higher order interpolation methods must be employed, eg quadratic or cubic splines or second or third order Hermit polynomials [Ahli63], [Ferg64], [ShYD83], [ShDa86], [Shim86]. The linear interpolation technique, however, can be used in a simulation program which uses an explicit time integration method combined with "the current source" or "the secant" model for individual transistors [JLLR73]. In either model all that is required is the value of the drain current determined by the bias conditions of a given device, and the continuity of the first derivative is not necessary. The timing simulator FACTS employs an explicit integration method combined with the source current model.

A number of interpolation schemes have been examined and compared here, with regard to speed and memory requirements for a given accuracy of results. The interpolation schemes compared are:

1. Scheme 1: linear interpolation along all three variables, V_{DS}, V_{GS} and V_{SB};
2. Scheme 2: linear interpolation along V_{DS} and V_{SB} and a third order Hermite polynomial along V_{GS};
3. Scheme 3: linear interpolation along V_{GS} and V_{SB} and a third order Hermite polynomial along V_{DS};
4. Scheme 4: a two-dimensional third order Hermite polynomial in the V_{DS}-V_{GS} plane combined with linear interpolation along V_{SB}.

A Hermite polynomial is one that matches the values of the function and of the first derivative at all sample points in the table [Ahli63], [Ferg64], [Forr72], [ShYD83]. Mono-

Interpolation scheme	Table size $P_{DS} \times P_{GS} \times P_{SB}$		
	$40 \times 20 \times 5$	$20 \times 10 \times 5$	$10 \times 5 \times 5$
1	1.094%	8.107%	27.163%
2	0.602%	3.485%	8.123%
3	0.791%	5.443%	14.243%
4	0.092%	1.854%	3.843%

Table 4.2: RSM percentage error for interpolation schemes 1 to 4, for uniformly spaced tables of different sizes. P_{DS}, P_{GS} and P_{SB} denote the number of points along V_{DS}, V_{GS} and V_{SB}, respectively.

tonicity of interpolated data is not guaranteed, however, it can be achieved by introducing additional parameters, as described in [ShYD83].

The percentage root mean square (rms) error relative to the analytical model is chosen as a criterion for the comparison. This approach, however, masks the errors in the very low current region of operation. Since this model is intended for the timing simulation of digital MOS transistor circuits, this is well justified.

In the first comparison a table with uniformly spaced values, for a P1 process n-channel transistor having $L = 2.9 \times 10^{-6}m$ and $W = 3.7 \times 10^{-6}m$, is used. The rms percentage error was evaluated by comparing the interpolated at calculated drain current values at 10000 randomly chosen bias points. The results are given in Table 4.2.

A substantial improvement on the figures presented in Table 4.2 is obtained if a nonuniform spacing of entries is adopted in the table, with the entries for low bias conditions being spaced closer together. The original table has been subdivided into 5 regions along the V_{DS} direction, 3 regions along the V_{GS} direction, and 2 regions along the V_{SB} direction. The data spacings in separate regions have then been manually adjusted to minimize the rms error. The results of the best efforts are shown in Table 4.3.

The first interpolation scheme is quite simple and requires the storage of function values only.

There are two possibilities of implementing the second and the third interpolation schemes. Either the values of the function and of the first derivative are stored, and the interpolation is performed by first calculating the four necessary coefficients, or the four coefficients themselves are precalculated and stored instead [ShYD83], [Shim86], [ShDa86]. In the first case the storage requirement is twice that of Scheme 1 and the speed has been found to be about one third of the speed of Scheme 1. The second implementation proposal of Scheme 2 and Scheme 3 requires a storage which is four

Interpolation scheme	Table size $P_{DS} \times P_{GS} \times P_{SB}$		
	$40 \times 20 \times 5$	$20 \times 20 \times 5$	$10 \times 5 \times 5$
1	0.304%	0.946%	3.632%
2	0.091%	0.213%	1.193%
3	0.128%	0.443%	2.321%
4	0.042%	0.072%	0.213%

Table 4.3: RSM percentage error for interpolation schemes 1 to 4, for nonuniformly spaced tables of different sizes. P_{DS}, P_{GS} and P_{SB} denote the number of points along V_{DS}, V_{GS} and V_{SB}, respectively.

times that of Scheme 1 and its speed is about 70% of the speed of Scheme 1.

In Scheme 4 there is a need to store either the values of the function, the values of the function's first derivatives with respect to V_{DS} and V_{GS}, and the values of the function's first mixed derivative, ie $d^2I_{DS}/dV_{DS}dV_{GS}$ evaluated at all sample points in the table, or the values of the corresponding sixteen coefficients [ShYD83], [Shim86], [ShDa86]. In the first case the storage is four times that of Scheme 1 and the speed is about an order of magnitude slower than the speed of Scheme 1, and in the second case the storage requirement is sixteen times that of Scheme 1 and the speed is about one fifth of the speed of Scheme 1.

Since the simulator speed is of prime importance, it was decided to use the simple linear interpolation scheme, ie Scheme 1. With a nonuniform table of approximately 2000 points (Table 4.3), an rms error of less than 1% is achieved. This is quite acceptable, especially in light of the fact that the analytical model itself represents the experimental results within the experimental range of $5\% - 8\%$ caused by the spread of process parameters across a wafer and from wafer to wafer. The timing simulator, FACTS, modified in this way was approximately 10% slower than in its original version. Some response times are presented in Chapters 7 and 8.

4.6 The Target Fabrication Process

The circuit designed during the course of this research are targeted at a hypothetical double metal p-well CMOS process, termed here the *Orbit*$^+$ process, with a minimum feature size of 1-micrometer. This has been chosen for two major reasons: firstly 1-micrometer CMOS technology is already well within the capabilities of commercial firms (in fact a 0.8-micrometer CMOS technology is being currently tested and a 0.5-

Parameter name	Units	Orbit n-channel	Orbit p-channel	$Orbit^+$ n-channel	$Orbit^+$ p-channel
μ^o	Vm^{-2}	0.065	0.025	0.065	0.025
t_{ox}	m	400×10^{-10}	400×10^{-10}	200×10^{-10}	200×10^{-10}
N_A	m^{-3}	1.25×10^{22}	8.7×10^{21}	2.50×10^{21}	1.74×10^{22}
r_j	m	0.25×10^{-6}	0.35×10^{-6}	0.13×10^{-6}	0.18×10^{-6}
ΔL_m	m	0.30×10^{-6}	0.40×10^{-6}	0.10×10^{-6}	0.10×10^{-6}
ΔW_m	m	0.70×10^{-6}	0.80×10^{-6}	0.20×10^{-6}	0.40×10^{-6}
θ_0	V^{-1}	0.02	0.04	0.04	0.08
θ_1	V^{-1}	0.005	0.008	0.005	0.008
η	mV^{-1}	0.80	0.70	0.80	0.70
η_1	m^{-1}	0.03	0.05	0.03	0.05
E_{cy}	Vm^{-1}	2.10×10^6	2.60×10^6	2.10×10^6	2.60×10^6
δ	$V^{-1/2}$	0.50	0.60	0.50	0.60
σ	$V^{-1/2}$	0.02	0.02	0.02	0.02
α_0	$V^{-1/2}$	0.15	0.20	0.15	0.20
V_{FB}	V	-0.60	0.60	-0.60	0.60
V_{bi}	V	0.85	0.75	0.85	0.75
V_{th}(large)	V	0.75	-0.75	0.60	-0.60
Layer Capac.					
metal2	$10^{-4}Fm^{-2}$	0.15	0.15	0.15	0.15
metal1	$10^{-4}Fm^{-2}$	0.30	0.30	0.30	0.30
poly	$10^{-4}Fm^{-2}$	0.60	0.60	0.60	0.60

Table 4.4: Model parameters corresponding to the Orbit and the $Orbit^+$ processes.

micrometer CMOS technology is being investigated [Koll88]) and secondly experimental data and process parameters, to be described shortly, for a 2-micrometer CMOS process have been available, and it was envisaged that with this information and the modified FACTS timing simulation program performance predictions, corresponding to the hypothetical 1-micrometer process, of the designs could be made with a sufficient level of accuracy to be of useful value.

The design rules for the hypothetical $Orbit^+$ have been obtained from the existing 2-micrometer Orbit [Orb87] process with the use of the scaling principle [DGYR74]. The model parameters for the Orbit process, extracted as described in Appendix B, and the corresponding parameters determined with the scaling principle, for the hypothetical $Orbit^+$ process are shown in Table 4.4. The current-voltage characteristics of the minimum size n-channel transistor fabricated with the Orbit process are compared with the analytical model in Figure 4.13(a). The characteristics of the corresponding $Orbit^+$ transistor, as predicted by the model, are given in part (b) of the figure.

A major module of the Floating Point Adder system, namely the 27-bit mantissa integer adder (Mantissa Adder module) was fabricated using the 2-micrometer, double metal,

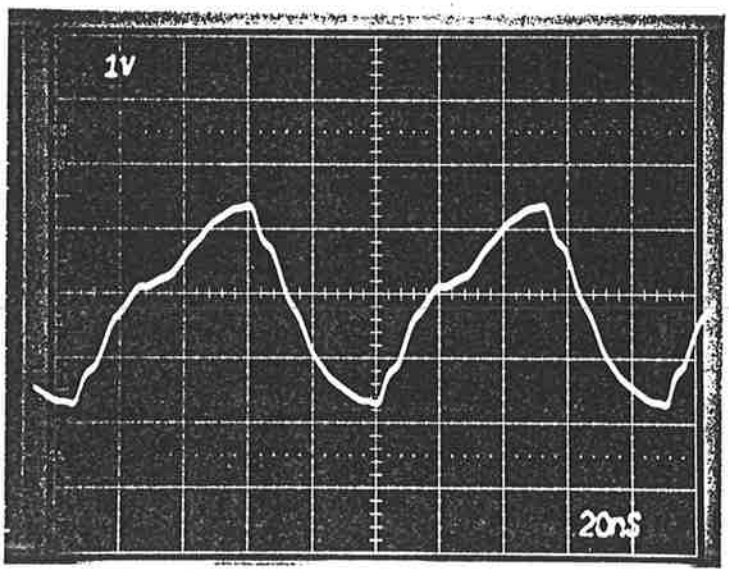


Figure 4.12: Mantissa Adder: experimental results for the critical path delay.

p-well CMOS process of the Orbit Semiconductor foundry. The speed of the critical path of this module, rather than the module's functional correctness was of major importance, since a 12-bit version have also been fabricated using the 5-micrometer, p-well CMOS P1 process, and the circuit was found to be functionally correct. The experimental data obtained from testing the fabricated module was used to "calibrate" the FACTS simulator, by adjusting the layer capacitance values so that the simulated and the measured results for this module and a number of unloaded ring oscillator circuits, to be described shortly, were in agreement. Such a calibration was necessary for the following reason. With the improved analytical MOS transistor model embedded into FACTS, the behaviour of individual transistors is modelled accurately, however, because of the simple way that circuit node and transistor terminal capacitances are treated within FACTS (ie all circuit node capacitances are voltage independent and only capacitance from a given circuit node to ground is allowed) it becomes necessary to empirically adjust the values of layer capacitances to compensate for simulator errors which result from the simple treatment of the circuit node capacitances.

In order to measure the time delay the Mantissa Adder was arranged as an oscillator. This was done by arranging the module as a closed loop and connecting the most significant bit of the sum output to one of the inputs in the least significant bit position. The remaining inputs were connected to the positive power supply and ground in such a way as to result in a carry propagation from the least significant to the most significant bit positions of the module. The circuit oscillates by generating a carry signal in the least significant bit position and propagating it to the most significant bit of the sum.

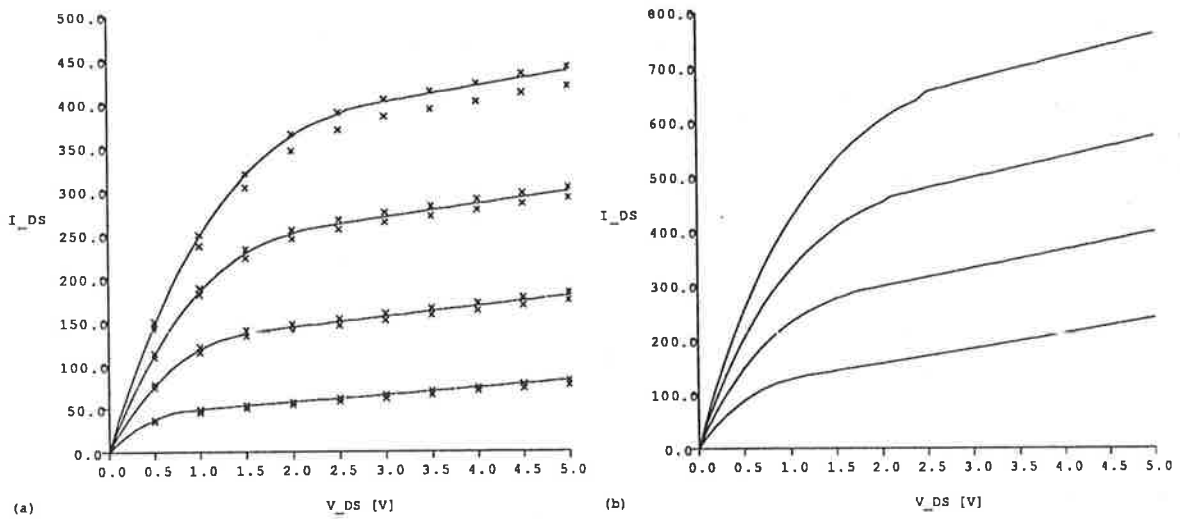


Figure 4.13: $I_{DS} - V_{DS}$ characteristics: (a) n-channel, Orbit, $W = 2.1 \times 10^{-6}m$, $L = 1.7 \times 10^{-6}m$, $V_{SB} = 0V$, (b) n-channel, $Orbit^{+}$ $W = 1.35 \times 10^{-6}m$, $L = 0.9 \times 10^{-6}m$, $V_{SB} = 0V$. V_{GS} ranges from 2V to 5V in steps of 1V. I_{DS} is in units of 10^{-6} A.

This reverses the logic level on the most significant bit of the sum and results in a new carry signal, having the opposite logic level to the previous carry signal, to be propagated. In order not to artificially load the circuit a three stage pad driver was included, with a minimum first stage, between the least significant sum output and a probe output pad.

The experimental results are shown in Figure 4.12. The load capacitance presented to the output buffer by the probe and the oscilloscope was approximately $40 \times 10^{-12}F$. The circuit oscillates at 10.87 Mhz at room temperature, with the power supply voltage at 5V, and dissipates 5.3mW of power. This gives a delay of the critical signal path of approximately $46ns$ or on average $1.70ns$ per 1-bit adder stage. Eight circuits were measured and the range of delays was found to be $42ns - 48ns$. After adjusting the layer capacitance values to those shown in Table 4.4 the simulated frequency of the circuit was found to be 10.7 Mhz, as shown in Figure 4.14 (which is within the range of the experimental results), and the power dissipation according to FACTS was 5.0mW.

The predicted performance according to the modified FACTS program of the same circuit, corresponding to the 1-micrometer $Orbit^{+}$ process is shown Figure 4.15. The circuit simulation results indicate that the Mantissa Adder module will be approximately 2.5 faster if it is fabricated using the $Orbit^{+}$ process.

In addition a number of unloaded ring oscillator circuits was implemented. The ring oscillators (all 21-stage) included: minimum size inverter ring oscillator, double size in-

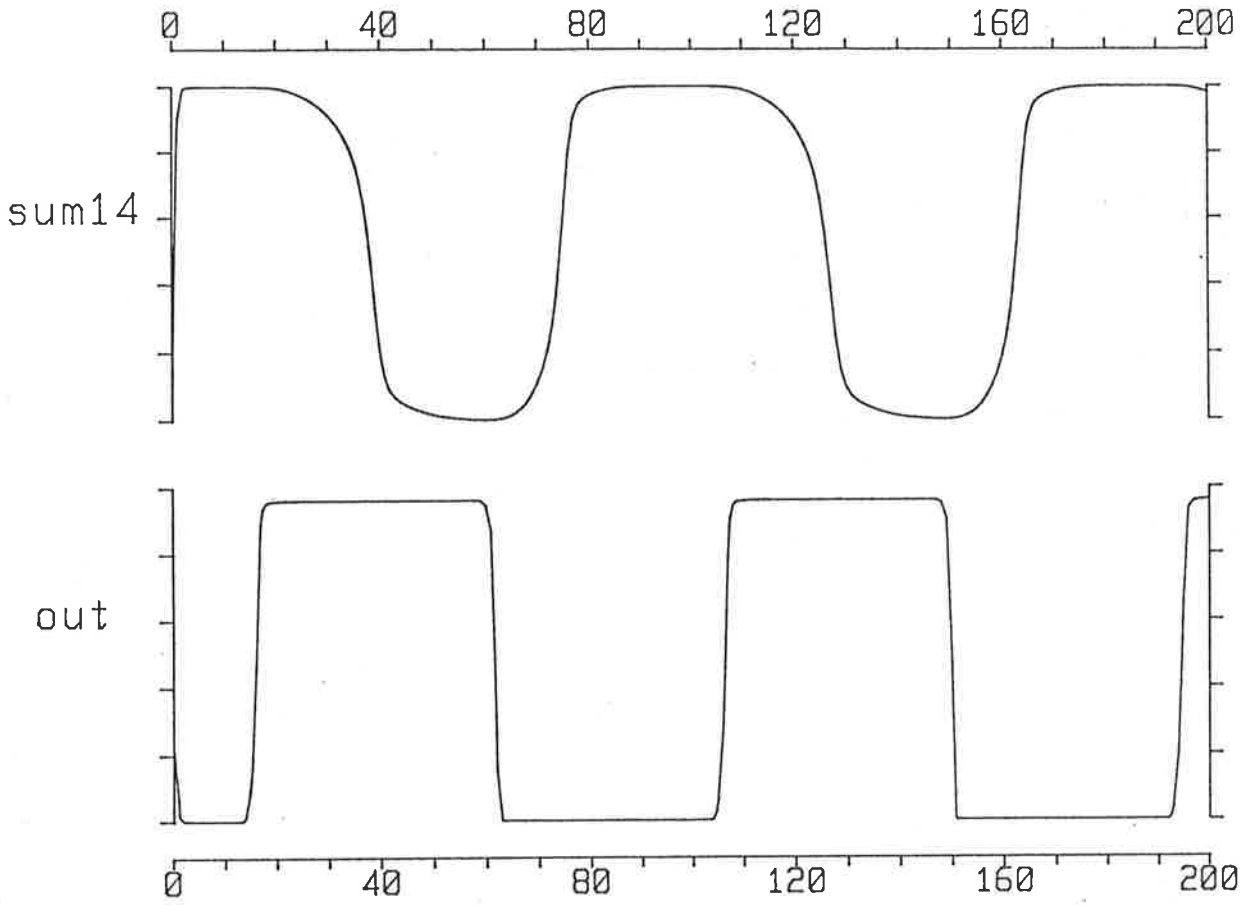


Figure 4.14: Mantissa Adder: FACTS simulation results for the Orbit process parameters. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

verter ring oscillator, minimum size NAND gate ring oscillator, minimum size NOR gate ring oscillator, minimum size inverter in series with an open minimum size transmission gate ring oscillator. It was found that the simulation and experimental results agreed to within 7% for all circuits described in this section.

4.7 Summary

An improved accuracy analytical MOS transistor model has been presented. This model was formulated following a detailed study presented in Chapter 3 and comparison of reviewed analytical models with experimental measurements performed during the course of this work. In particular, it has been found that all (with the exception of the model given in [FiPo79]) short channel effect models proposed in literature are in poor agreement with the experimental data, even for transistors with $L \approx 5 \times 10^{-6}m$.

A novel short channel effect analytical model has been presented. The model combines the two most widely used approaches in literature: the approach based on the a priori

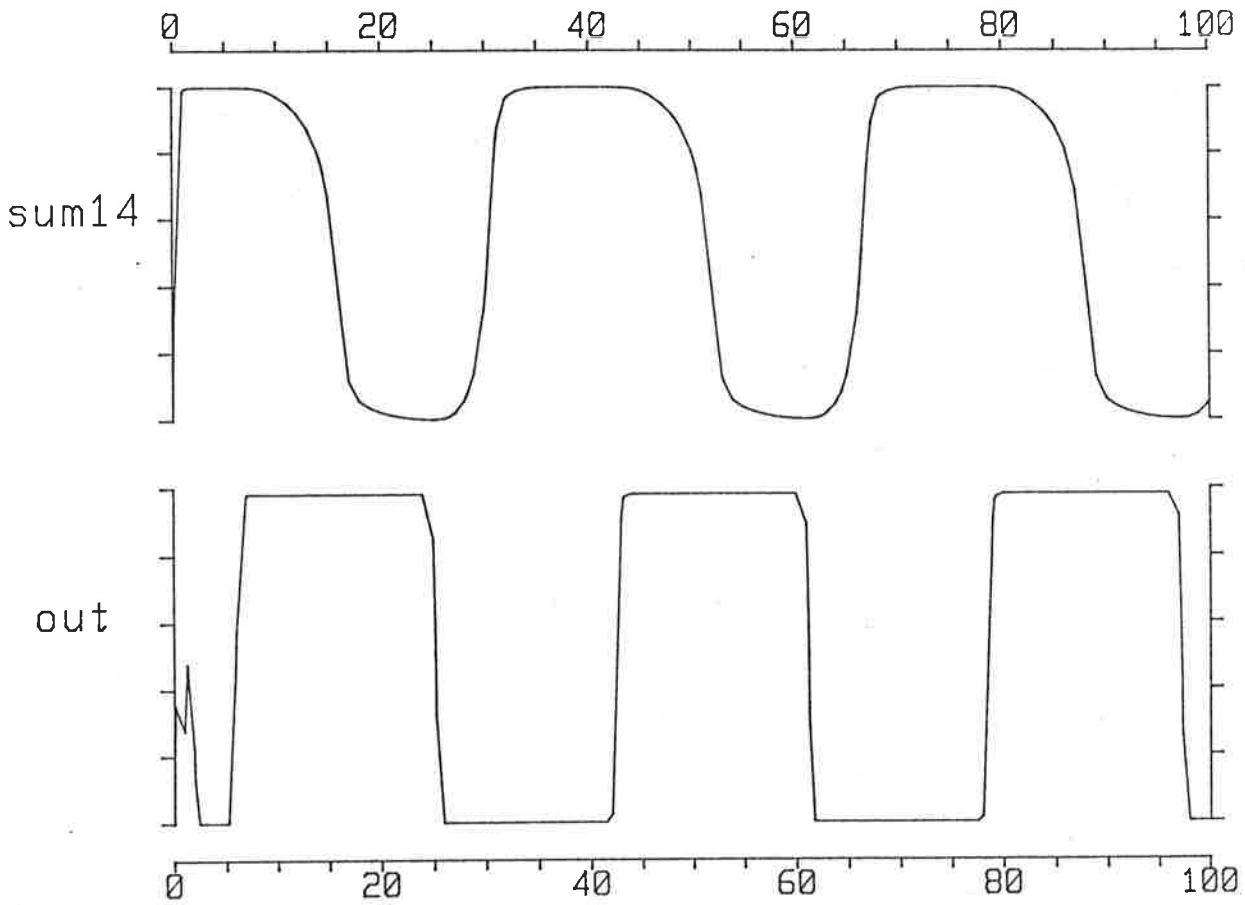


Figure 4.15: Mantissa Adder: FACTS simulation results for the $Orbit^+$ process parameters. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

charge sharing assumption and the approach based on the a priori potential distribution assumption. The derivation of the model has been guided by physical considerations, but four empirical parameters have been introduced to improve model's accuracy. The models based on the first assumption show a L^{-1} dependence of the short channel effect, whereas models based on the second assumption show an exponential dependence of the short channel effect with L. Consequently, these models predict negligibly small short channel effect for transistors with $L > 10^{-6}m$, and a much larger short channel effect for transistors with $L < 10^{-6}m$, than the models based on the a priori charge sharing assumption.

For other effects, which become significant as the transistor active dimensions are reduced, existing models have been found to be satisfactory. The complete analytical model incorporates the short channel effect, the narrow width effect, the small geometry effect, mobility dependence on the normal electric field component, velocity saturation effect, and the effect of source and drain resistances. Parameter dependence with geometry is also incorporated into the model, so that the only input for a given transistor

is its L and W.

This MOS transistor model has been incorporated into the FACTS simulator using a table look up scheme with a simple linear interpolation technique, to provide fast response. The modified simulator is still not suitable for accurate and reliable analog circuit simulation because of its simple treatment of transistor terminal and circuit node capacitances, the absence of a model for the subthreshold region of operation and the inaccurate representation of the transistor output conductance in the saturation region of operation.

The experimental results, obtained from measuring the performances of a number of circuits fabricated using the 2-micrometer CMOS Orbit process, have been used to calibrate the modified FACTS program, such that the measured and predicted circuit delays would agree. The modified (and calibrated) FACTS program is used in Chapters 7 and 8 for predicting the performances of the Floating Point Adder system and the Floating Point Multiplier system, respectively, for the hypothetical 1-micrometer CMOS process $Orbit^+$ process. The $Orbit^+$ process has been defined by scaling the Orbit process parameters by one half, as described in [LEGK74].

Having discussed the design methodology followed in this work and the work that resulted in an improved circuit analysis program used extensively in this work, the remainder of this thesis deals with the design work undertaken in this research project.

Chapter 5

Numerical Modelling of MOS Transistors

5.1 Introduction

The main disadvantage of analytical models is not the elaborate model parameter evaluation but the very limited ability of these models to predict the achievable performance of new transistors. This limitation is the direct consequence of the various assumptions and approximations on which such models are based. Thus, in order to develop new fabrication processes and novel transistors in a cost effective manner, more general and accurate solutions of the basic semiconductor equations, governing the internal behaviour of semiconductor devices, are needed. Such solutions are only obtainable by numerical techniques.

Using numerical MOS transistor models it is possible to accurately predict the effects of material properties (eg doping variations), physical boundaries (eg contacts, surface states) and the environment (eg adjacent transistors) on device operations. As a result, such models can serve as an aid both for a deeper understanding of transistor behaviour and for the development and validation of analytical transistor models, as well as being a tool for evaluating and optimizing the performance of a new transistor prior to that transistor being fabricated.

Numerical modeling was first suggested by Gummel [Gumm64] for the one-dimensional bipolar transistor. A two-dimensional numerical analysis of a semiconductor device was first carried out by Kennedy and O'Brien [KeOB69] who investigated the junction field-effect transistor. Two-dimensional MOS transistor models soon followed [VaXu71], [KeMu73], [Mock73]. Many successful attempts at simulating the MOS transistor as a

two-dimensional structure have been carried out since then. Some examples of the large number of MOS transistor simulation programs that have been developed include: GEMINI [GrDu80], MINIMOS [SeSP80], FIELDAY [BCGS81], SIFCOD and CADDET [COCM86].

The recent miniaturization of MOS transistors demonstrates the need for three-dimensional numerical simulation for transistors with aspect ratios close to one, and operating in the subthreshold region or under high field conditions. The simulation of three-dimensional device structures presents a formidable problem requiring large computing resources. This is the main reason why relatively few attempts have been reported up to now [HuCh82], [KYYS82], [YKTH82], [ShDa85], [TMAS85], [ShDa86]. All of these have been carried out using super and vector computers, still not widely available in the scientific community.

This chapter examines the mathematical model normally used in MOS transistor numerical simulation software, one of the most common techniques used for the discretization of the basic semiconductor equations, the form of the resulting linear systems of algebraic equations and the common solution methods. The main objective is to accent the features of this problem which are relevant to the subsequent discussion of the design of the parallel computer intended for an efficient solution of the linear systems of algebraic equations and which is described in the following chapter.

5.2 The Mathematical Model

5.2.1 Basic Semiconductor Equations

The partial differential equations that describe the behaviour of electrons and holes under the influence of external electric field can be derived from Boltzmann transport equation, assuming the motion of charge carriers is semiclassical [BaRF83], [EnDM83], [Selb84], [FNPP85], [Engl86]. In that treatment it is assumed that the response of charge carriers to a change in the electric field is much faster than the effective rate of change in the field. For the electrical field strengths and temperatures encountered in the present day silicon devices this simplified description is adequate.

The most familiar model of charge transport in a semiconductor device is that proposed by Van Roosbroeck [Roos50]. It consist of the Poisson equation (5.1) describing the potential distribution in the interior of the semiconductor device being studied, the electrical current density continuity equations for electrons (5.2) and holes (5.3) which

characterize the balance of sinks and sources for electron and hole currents, and the current density relations for electrons (5.4) and holes (5.5) that describe the absolute value, direction and orientation of electron and hole currents [EnDM83], [Selb84], [SeRi84]:

$$\nabla^2\psi = -\frac{q}{\epsilon_s}(p - n + N), \tag{5.1}$$

$$q\frac{\partial n}{\partial t} - \nabla \cdot \vec{J}_n = -qR(\psi, n, p), \tag{5.2}$$

$$q\frac{\partial p}{\partial t} + \nabla \cdot \vec{J}_p = -qR(\psi, n, p). \tag{5.3}$$

In equations (5.1)–(5.3) the fundamental unknowns are as follows (some of these quantities have previously been defined in Chapter 3, and their definitions are repeated here for completeness): ψ is the electrostatic potential, n is the free electron concentration and p is the free hole concentration. The other quantities include: the magnitude of the electronic charge (q), the permittivity of the substrate material (ϵ_s) and the net recombination rate (R). The net impurity concentration (N) contains all ionized donors (N_D) and acceptors (N_A) and it is given by $N = N_D - N_A$. The current density equations for electrons ($\vec{J}_n$) and holes ($\vec{J}_p$) are given by:

$$\vec{J}_n = -q\mu_n n \nabla\psi + qD_n \nabla n, \tag{5.4}$$

$$\vec{J}_p = -q\mu_p p \nabla\psi - qD_p \nabla p, \tag{5.5}$$

where μ_n and μ_p are the mobilities of electrons and holes, respectively, and D_n and D_p are the corresponding diffusion coefficients. Under nondegenerate conditions the diffusion coefficients and the mobilities are related by the Einstein relation [Smit78], [Sze81]:

$$D_n = \mu_n kT/q, \quad D_p = \mu_p kT/q, \tag{5.6}$$

where kT is the thermal energy.

Equations (5.1)–(5.5) form a set of coupled, nonlinear partial differential equations. These equations contain quantities such as the net recombination rate and the mobilities that themselves are the result of rather complicated physical mechanisms. These quantities depend on the local values of carrier concentrations, current densities and electric fields. In addition, a relation between carrier densities, the corresponding quasi-Fermi levels and the electrostatic potential is needed. This relation results from carrier statistics and has to include the effects of bang-gap narrowing and degeneration. Therefore adequate physical models must be incorporated into equations (5.1)–(5.5), so that all relevant physical effects are properly accounted for.

For the purposes of numerical device simulation the abovementioned physical mechanisms are normally modelled by means of empirical expressions. Analytical models for

R, for example, can be found in [BlWi82], [EnDM83], [Selb84] and [BRGC86]. Moderate degeneracy is normally included by introducing an effective doping dependent intrinsic concentration ($n_{i,e}$) which is expressed as a function of N_D and N_A [EnDM83], [Selb84], [BRGC86]. The net doping density, as a function of position, ie the doping profile geometry within the device, is obtained from fabrication process simulation programs such as SUPREM or SUPRA [COCM86], [Penu86]. The effective mobility is obtained by combining the various physical mechanisms present in the device. Some of these include: lattice scattering [Grov67], [NoBL73], [DoLe81], ionized impurity scattering [CoWe50], [NoBl73], [LiTh77], surface scattering and velocity saturation discussed in Chapter 3. An excellent discussion on this subject is given in [EnDM83], [Selb84] and [BRGC86].

A detailed discussion of the derivation and the relevant assumptions of the basic semiconductor equations can be found in [FiRB83] and [Selb84]. The model is justified on the basis of agreement between computational and experimental results [SeRi84]. Equations (5.1)–(5.3) describe the current flow in the semiconductor and determine the electrical performance of the simulated device. These equations must be solved for the unknowns n, p and ψ under a given set of boundary conditions.

5.2.2 Equation Domain and Boundary Conditions

Figure 5.1 depicts a simplified two-dimensional simulation geometry for a planar MOS transistor which represents the domain on which the partial differential equations are posed.

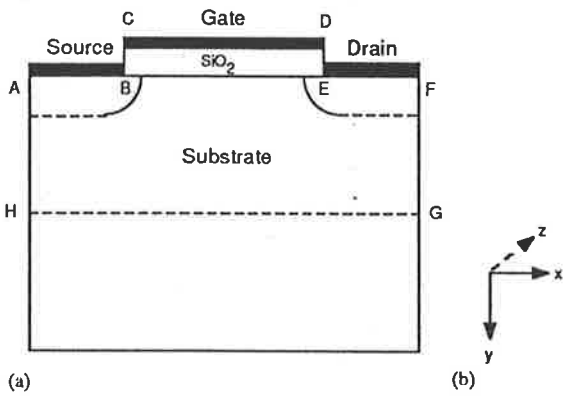


Figure 5.1: (a) Simulation geometry of a planar MOSFET (after [Selb84]), (b) definition of the coordinate system.

The boundaries A-B, C-D, E-F, B-E can be interpreted as physical boundaries repre-

senting the three idealised contacts and an interface between the semiconductor (Si) and the insulator (SiO_2), respectively. The boundaries A-H, B-C, D-E, F-G and G-H form the set of artificial boundaries and are only introduced to simplify the solution of the basic equations [GrDu80], [SeSP80], [EnDM83], [Selb84].

Ohmic contacts (A-B: source, G-H: bulk, E-F: drain) are often idealized by assuming thermal equilibrium, $np = n_{i,e}^2$, which corresponds to infinite contact recombination velocities, and space charge neutrality, $n - p - N = 0$. These two conditions can be transformed into Dirichlet boundary conditions (one where a constant value is specified for the unknown) for electrons and holes:

$$n = \frac{\sqrt{N^2 + 4n_{i,e}^2} + N}{2}, \quad p = \frac{\sqrt{N^2 + 4n_{i,e}^2} - N}{2}. \tag{5.7}$$

The electrostatic potential at an ohmic contact is kept at the applied voltage (V_{appl}) plus the appropriate built in voltage (V_{bi}) caused by the doping, $\psi = V_{appl} + V_{bi}$. Thus, under the conditions of voltage controlled ohmic contacts, Dirichlet boundary conditions hold for the electrostatic potential and for the carrier concentrations.

Two approaches are possible for the treatment of the gate oxide region. One can either determine the field within the gate oxide region by solving Laplace equation there, specifying the gate electrode as a Dirichlet boundary condition (with the appropriate work function for the material included in the expression for the electrostatic potential), or one can assume a one-dimensional field perpendicular to the surface [SeSP80], [EnDM83], [Selb84].

At the noncontact boundaries (lines A-H, B-C, D-E and F-G) homogeneous Neumann (Neuman boundary conditions, ie where the value of the normal gradient is specified) boundary conditions are applied to insure that no current can flow in or out of the device along these edges. Thus, the normal components of the electron and hole current densities and electric field strength are assumed to be zero:

$$\hat{n} \cdot \vec{J}_n = \hat{n} \cdot \vec{J}_p = 0, \quad \hat{n} \cdot \nabla\psi = 0, \tag{5.8}$$

where $\hat{n}$ is the normal unit vector. The resulting error caused by the introduction of these artificial boundaries in order to limit the simulation domain to a manageable size can be made negligible small by placing these boundaries at a sufficiently large distance from the region of interest [FoWa60], [EnDM83].

At the silicon-insulator interface (B-E) Gauss law in differential form must be obeyed:

$$\hat{n} \cdot (\epsilon_s \nabla\psi) - \hat{n} \cdot (\epsilon_{ox} \nabla\psi) = Q_{ss}, \tag{5.9}$$

where Q_{ss} is the sum of all effective net oxide charges per unit area at the silicon-insulator interface and ϵ_{ox} is the permittivity of the oxide. For the current continuity

equations the current components normal to the interface must equal to the surface recombination rate (R_s) multiplied by the charge per carrier:

$$\hat{n} \cdot \vec{J_n} = -qR_s, \;\; \hat{n} \cdot \vec{J_p} = qR_s. \tag{5.10}$$

It is to note that the basic semiconductor equations only constitute a time dependent problem if the boundary conditions for the electrostatic potential are time dependent. In the case of these boundary conditions being time invariant, ie a static (DC) analysis problem, the semiconductor equations reduce to a system of three coupled elliptic equations, in the unknowns ψ, n and p.

5.2.3 Scaling

By substituting the current density relations (5.4) and (5.5) into the continuity equations (5.2) and (5.3), respectively, one obtains a system of three partial differential equations with the unknown variables being ψ, n, p:

$$\nabla^2 \psi = -\frac{q}{\epsilon_s}(p - n + N), \tag{5.11}$$

$$\frac{\partial n}{\partial t} = \nabla \cdot (D_n \nabla n - \mu_n n \nabla \psi) - R(\psi, n, p), \tag{5.12}$$

$$\frac{\partial p}{\partial t} = \nabla \cdot (D_p \nabla p + \mu_p p \nabla \psi) - R(\psi, n, p). \tag{5.13}$$

For nondegenerate materials the carrier concentrations are given by the well known Boltzmann approximations:

$$n = n_i \, e^{(\psi - \phi_n)q/kT}, \;\; p = n_i \, e^{-(\psi - \phi_p)q/kT}, \tag{5.14}$$

where n_i, the intrinsic concentration, is often replaced by $n_{i,e}$, the effective intrinsic carrier concentration, to model the effects of moderate levels of degeneracy and band gap narrowing in an empirical fashion [MeVM81].

Since the unknown variables in the system (5.11)–(5.13) are of different orders of magnitude the first step towards structural analysis of the basic equations involves appropriate scaling [DeMa68]. Scaling is also used for the simple reasons of computational convenience [SeSP80], [BaRF83], [Selb84]. The most common method is that developed by De Mari [DeMa68]. The scaling factors are summarized in Table 5.1. Other scaling factors have been also proposed in [Selb84].

Retaining the appropriate symbols but remembering that they represent normalized quantities the scaled equations become:

$$\nabla^2 \psi = n - p - N, \tag{5.15}$$

Physical quantity	Normalization factor
x, y, z	$[\epsilon_s kT/(q^2 n_i)]^{1/2} (= L_D)$
ψ	kT/q
n, p, N	n_i
D_n , D_p	$D_0 = 10^{-4} m^2 s^{-1}$
μ_n , μ_p	q/kT
R	n_i/L_D^2
t	L_D^2/D_0

Table 5.1: Scaling factors after De Mari [DeMa68].

$$\frac{\partial n}{\partial t} = -\nabla \cdot [\mu_n (n \nabla \psi - \nabla n)] - R(\psi, n, p), \tag{5.16}$$

$$\frac{\partial p}{\partial t} = +\nabla \cdot [\mu_p (p \nabla \psi + \nabla p)] - R(\psi, n, p). \tag{5.17}$$

5.3 Solution of Semiconductor Equations

The system of the basic semiconductor equations with the given boundary conditions cannot be solved explicitly and only an approximate solution using numerical techniques is possible. Any numerical approach for the solution of such a system consists of three stages ([Gumm64], [DeMa68], [ENDM83], [Selb84]) as follows. Firstly, the space and time domains in which the solution is required are subdivided into subdomains in which the solution can be approximated with a desired accuracy. Secondly, the differential equations are approximated in each of the subdomains by algebraic equations which involve only values of the unknown variables within each of the subdomains. In this way one obtains a large but finite set of algebraic, in general nonlinear, equations with unknowns comprised of the approximations of the continuous unknown variables at discrete points. The third and final task to be carried out is the solution of that set of the algebraic equations for the unknown variables.

Each of these stages is discussed in more detail in the remainder of this section. The discussion is based on a number of sources which include: [FoWa60], [GrDu80], [SeSP80], [BlWi82], [EnDM83], [FiRB83], [Selb84], [SeRi84] and [Engl86].

5.3.1 Discretization in the Space Domain

There exist, basically, two classical methods for a partitioning (discretization) of the space domain and for the choice of functions to approximate the unknown variables

within the domains. These are the finite element method and the finite difference method. The finite difference method, which is regarded by the author to be conceptually simpler, is focused in this discussion.

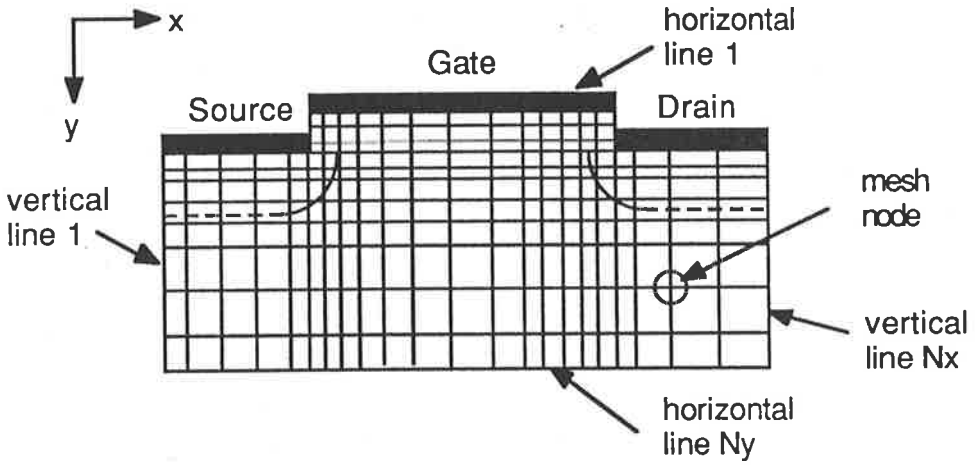


Figure 5.2: Finite difference mesh for a MOS transistor (two-dimensional case).

In the method of finite differences the spacial domain is partitioned into small areas by a set of lines (in general spaced nonuniformly) running parallel to a coordinate system, and the system of partial differential equations is discretized on the resulting rectangular system of nodes. Figure 5.2 shows a typical two-dimensional (for a three-dimensional mesh each node in Figure 5.2 becomes a line of nodes running in the depth direction) finite difference mesh formed by placing N_x vertical lines parallel to the y axis and N_y horizontal lines parallel to the x axis. There are thus a total $m = N_x \times N_y$ (or $m = N_x \times N_y \times N_z$ for a three-dimensional mesh) intersecting points (nodes) of mesh lines on which an approximate solution for the differential equations is sought. (The actual number of nodes is slightly smaller due to the space domain being composed of two different rectangular regions; one for the insulator and one for the silicon. This small difference will be neglected here to simplify the discussion.)

The most common approach is to discretize the equations using the five-point star discretization method in two space dimensions or the seven-point star method in three space dimensions. In this approach the approximation relates the value of an unknown at a given node to the values of the unknown at the four (two-dimensional case) or at the six (three-dimensional case) nearest neighbouring nodes. This is the simplest approach that provides acceptable accuracy and is capable of treating nonplanar interfaces [GrDu80]. An outline of the discretization procedure will be presented in what follows to demonstrate the nature of the resulting equations. In order to simplify the following discussion the system of basic semiconductor equations will be considered in

two dimensions. Some discussion of a three-dimensional case will be presented in the next chapter.

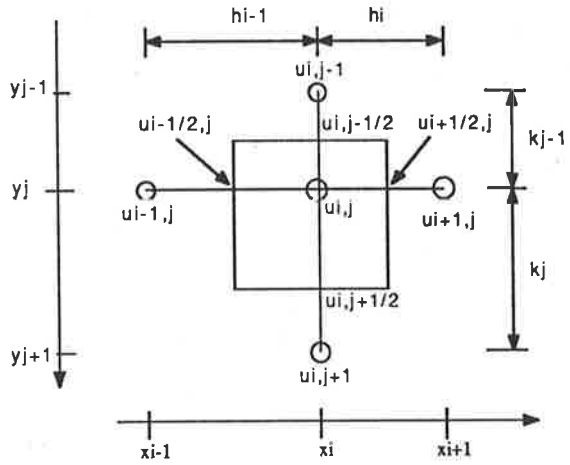


Figure 5.3: Finite difference method: adopted nomenclature.

The adopted nomenclature assuming a general unknown variable u is shown in Figure 5.3. The scaled Poisson equation (5.15) can be written for all inner nodes ($1 < i < N_x$, $1 < j < N_y$) of the space domain (with all terms involving higher derivatives neglected) as follows:

$$\frac{\left(\frac{\partial\psi}{\partial x}\right)_{i+1/2,j} - \left(\frac{\partial\psi}{\partial x}\right)_{i-1/2,j}}{\frac{h_i+h_{i-1}}{2}} + \frac{\left(\frac{\partial\psi}{\partial y}\right)_{i,j+1/2} - \left(\frac{\partial\psi}{\partial y}\right)_{i,j-1/2}}{\frac{k_j+k_{-1}}{2}} = n_{i,j} - p_{i,j} - N_{i,j} \tag{5.18}$$

and a similar treatment yields for the continuity equation for electrons:

$$\frac{(-J_{nx})_{i+1/2,j} - (-J_{nx})_{i-1/2,j}}{\frac{h_i+h_{i-1}}{2}} + \frac{(-J_{ny})_{i,j+1/2} - (-J_{ny})_{i,j-1/2}}{\frac{k_j+k_{j-1}}{2}} = R_{i,j}, \tag{5.19}$$

where J_{nx} and J_{ny} are the scaled electron current density components in x and y direction:

$$J_{nx} = \mu_n n\frac{\partial\psi}{\partial x} - \mu_n\frac{\partial n}{\partial x}, \quad J_{ny} = \mu_n n\frac{\partial\psi}{\partial y} - \mu_n\frac{\partial n}{\partial y}. \tag{5.20}$$

An analogous equations is obtained for the current continuity for holes, and this equations is omitted here for brevity.

In order to obtain the discretized equations the mid-interval values of the quantities $\partial\psi/\partial x, \partial\psi/\partial y, J_{nx}$ and J_{ny} must be replaced with an appropriate finite difference approximation. It is normally assumed that these quantities are constant within each interval. The following expressions are obtained for the electrostatic potential (with all terms involving higher order derivatives neglected):

$$\left(\frac{\partial\psi}{\partial x}\right)_{i+1/2,j} = \frac{\psi_{i+1,j} - \psi_{i,j}}{h_i}, \quad \left(\frac{\partial\psi}{\partial y}\right)_{i,j+1/2} = \frac{\psi_{i,j+1} - \psi_{i,j}}{k_j}. \tag{5.21}$$

The discretization of Poisson equation is completed by substituting (5.21) into expression (5.18):

$$\begin{aligned} & \psi_{i,j-1}\frac{2}{k_{j-1}(k_{j-1}+k_j)} + \psi_{i-1,j}\frac{2}{h_{i-1}(h_{i-1}+h_i)} \\ - \; & \psi_{i,j}\left[\frac{2}{h_{i-1}(h_{i-1}+h_i)} + \frac{2}{h_i(h_{i-1}+h_i)} + \frac{2}{k_{j-1}(k_{j-1}+k_j)} + \frac{2}{k_j(k_{j-1}+k_j)}\right] \\ + \; & \psi_{i+1,j}\frac{2}{h_i(h_{i-1}+h_i)} + \psi_{i,j+1}\frac{2}{k_j(k_{j-1}+k_j)} \\ = \; & (n_{i,j} - p_{i,j} - N_{i,j})\,. \end{aligned} \tag{5.22}$$

According to the above expression the value of the electrostatic potential at any internal node i,j is related to the values of the electrostatic potential at the four neighbouring nodes located at: $i,j-1$, $i-1,j$, $i+1,j$ and $i,j+1$.

The nature of the required calculation described by expression (5.22) can be expressed more clearly by rewriting that expression in a simplified form. Firstly, a simplified notation is introduced, according to which the nodes and the corresponding unknowns are numbered sequentially from left to right, top to bottom (Figure 5.2) with a single index $i = 1, \ldots, N_x N_y$. Secondly, a simplified notation is used to identify the four neighbouring nodes of a given node i. With this notation the node at $i,j-1$ is denoted i,n (the node to the north of node i), the node at $i-1,j$ is denoted i,w (the node to the west of node i), the node at $i,j+1$ is denoted i,s (the node to the south of node i) and the node at $i+1,j$ is denoted i,e (the node to the east of node i). Using the notation defined in this way it is possible to write expression (5.22) as:

$$a_{i,n}\psi_{i,n} + a_{i,w}\psi_{i,w} - a_{i,i}\psi_i + a_{i,s}\psi_{i,s} + a_{i,e}\psi_{i,e} = Q_i, \tag{5.23}$$

The coefficients $a_{i,i}$, $a_{i,n}$, $a_{i,w}$, $a_{i,s}$ and $a_{i,e}$ in expression (5.23) are given in terms of the mesh node spacings, h_i and k_j, and express the coupling of the unknown ψ_i, at any internal node i, to the unknowns at the four neighbouring nodes immediately to the north, $\psi_{i,n}$, west, $\psi_{i,w}$, south, $\psi_{i,s}$, and east, $\psi_{i,e}$. (A node i in a three-dimensional mesh will have two additional neighbours, positioned one on either side in the z direction). For example, according to this notation, the coefficient $a_{i,w}$ is given by:

$$a_{i,w} = \frac{2}{h_{i-1}(h_{i-1}+h_i)}. \tag{5.24}$$

Thus, once the node spacings are established, ie the values for all h_i and k_j are fixed, these coupling coefficients remain constant throughout a solution process of a given linear algebraic equation. The quantity Q_i in expression (5.23) is a known constant, and it is given by:

$$Q_i = n_i - p_i - N_i. \tag{5.25}$$

A similar expression to (5.23) is obtained for the unknowns belonging to the mesh nodes along the silicon-insulator interface. For a Dirichlet boundary condition, eg boundary A-B or C-D (Figure 5.1), the actual values of the unknown variables are specified, as described in Section 5.2.2. The discretization of a boundary condition specified by a Neumann condition is performed by the so called "mirror imaging" technique [FoWa60]. In this approach the value of the unknown at a particular boundary node is related to the values of the unknowns at only three (instead of four as is the case for any internal node unknown) neighbouring nodes. For example, the corresponding expression for a boundary node i belonging to the boundary A-H (Figure 5.1) does not include a term involving coefficient $a_{i,w}$, since there is no nodes to the left (west) of that boundary node.

The system of nonlinear algebraic equations, in the unknowns ψ_i, which results when expression (5.23) is written for all nodes of the mesh is usually written in the following matrix form to simplify notation:

$$A_\psi \vec{\psi} = \vec{Q}_\psi. \tag{5.26}$$

In the above matrix equation the vector $\vec{\psi}$ consists of the elements ψ_i for each node in the problem domain where ψ_i is unknown. The vector $\vec{Q}_\psi$ contains the elements Q_i given by expression (5.25), and it also accounts for Dirichlet boundary conditions, ie it also contains elements consisting of the product terms of the coupling coefficients (a) and the values of the electrostatic potential (ψ_i) involving mesh nodes for which the values of ψ are directly specified by Dirichlet boundary conditions. The form of the coefficient matrix A_ψ depends on the scheme of numbering of the nodes within the solution region but each row of A_ψ has a maximum of only five (two-dimensional problem) or seven (three-dimensional problem) nonzero elements, ie matrix A_ψ is a sparse matrix.

A similar treatment (to that described above for the Poisson equation) of the current continuity equations results in algebraic equations which become unstable [ScGu69], [EnDM83] when the voltage change between two adjacent mesh nodes exceeds $2kT/q$. A suitable scheme for the discretization of the current continuity equations was first proposed by Scharfetter and Gummel [ScGu69] and is now universally used in numerical semiconductor device simulations. These authors suggest that the current continuity expressions be treated as independent differential equations in n and p with J_n, J_p, μ_n, μ_p and ψ assumed constant between mesh nodes. The final result (a detailed discussion can be found in [ScGu69], [Slot73] and [Selb84]) for the continuity equation for electrons reads (the subscript n in the symbol for electron mobility is omitted for clarity):

$$n_{i,j-1}\left[\mu_{i,j-1/2}\frac{Ber(\psi_{i,j-1}-\psi_{i,j})}{0.5k_{j-1}(k_j+k_{j-1})}\right]+n_{i-1,j}\left[\mu_{i-1/2,j}\frac{Ber(\psi_{i-1,j}-\psi_{i,j})}{0.5h_{i-1}(h_i+h_{i-1})}\right]$$

$$
\begin{aligned}
&- \; n_{i,j}\left[\mu_{i+1/2,j}\frac{Ber(\psi_{i,j}-\psi_{i+1,j})}{0.5h_{i-1}(h_i+h_{i+1})}+\mu_{i-1/2,j}\frac{Ber(\psi_{i,j}-\psi_{i-1,j})}{0.5h_i(h_i+h_{i-1})}\right.\\
&+ \; \left.\mu_{i,j+1/2}\frac{Ber(\psi_{i,j}-\psi_{i,j+1})}{0.5k_{j-1}(k_j+k_{j+1})}+\mu_{i,j-1/2}\frac{Ber(\psi_{i,j}-\psi_{i,j-1})}{0.5k_j(k_j+k_{j-1})}\right]\\
&+ \; n_{i+1,j}\left[\mu_{i+1/2,j}\frac{Ber(\psi_{i+1,j}-\psi_{i,j})}{0.5h_i(h_i+h_{i+1})}\right]+n_{i,j+1}\left[\mu_{i,j+1/2}\frac{Ber(\psi_{i,j+1}-\psi_{i,j})}{0.5k_j(k_j+k_{j+1})}\right]\\
&= \; R_{i,j}, \qquad (5.27)
\end{aligned}
$$

where $Ber(x)$ is the Bernoulli function given by $Ber(x) = x/(e^x - 1)$. An analogous expression can be derived for the hole current continuity equation in discrete form. In matrix form expression (5.27) becomes:

$$A_n\vec{n} = \vec{Q}_n. \qquad (5.28)$$

In the case of a discretized continuity equation the coupling coefficients $a_{i,i}$, $a_{i,n}$, $a_{i,w}$, $a_{i,s}$, $a_{i,e}$, ie the elements of the matrix A_n, depend on the node spacings, h_i and k_j, as well as on the values of the electrostatic potential at node i and the four neighbouring nodes through the Bernoulli function and through the expression for carrier mobility.

In summary, the discretization procedure in the space domain yields at each grid point one algebraic equation for the electrostatic potential (ψ) one for the electron concentration (n) and one for the hole concentration (p). The value of any unknown at a given node i is expressed in terms of the unknowns at the nodes in the immediate neighbourhood only. The resulting system comprises simultaneous algebraic equations in the space domain if DC analysis is considered, or coupled ordinary differential equations in the time domain for the transient problem.

The discretization in the time domain is briefly considered in the following section.

5.3.2 Discretization in the Time Domain

Time discretization of the scaled system (5.15)-(5.17) must also be applied for the case when the boundary condition for the electrostatic potential becomes time dependent. The basic difficulty in the solution of the transient system is the requirement that the numerical method chosen be unconditionally stable. The semiconductor equations exhibit a property called stiffness, due to the existence of widely varying time constants [Gear71]. Therefore, explicit ordinary differential equations methods such as predictor-corrector Adams schemes are not feasible for the semiconductor problem unless very small time steps $\delta t^r = t^{r+1} - t^r$ are used.

In a simplistic approach, for example, one may consider generalization of the Gauss-Seidel/Jacobi iteration and repeatedly solving each of the basic equations using the

"best available" values for the unknown variables, except in the recombination term. This scheme can be written as presented in [Selb84] and [BCFR86] as:

$$\frac{n^{r+1}-n^r}{\delta t^r} = \nabla\cdot[\mu_n^r(\nabla n^{r+1} - n^{r+1}\,\nabla\psi^r)] - R(\psi^r,n^r,p^r), \tag{5.29}$$

$$\frac{p^{r+1}-p^r}{\delta t^r} = \nabla\cdot[\mu_p^r(\nabla p^{r+1} - p^{r+1}\,\nabla\psi^r)] - R(\psi^r,n^r,p^r), \tag{5.30}$$

$$\nabla\psi^{r+1} = n^{r+1} - p^{r+1} - N, \tag{5.31}$$

where δt^r is the temporal discretization parameter at the rth time step and the index r refers to the respective variable at time $t = t^r = t^{r-1} + \delta t^r$. All three equations given above are linear in their own variable. Thus, using this discretization method one has to solve three linear algebraic equations at each time step. At time $t = t^{r+1}$ both continuity equations depend only on the value of the electrostatic potential calculated at $t = t^r$ which is equivalent to a forward difference in time for this term. As was demonstrated by Mock this causes serious instability limitation [Mock83].

The simple solution to avoid any instability problems is to replace the potential terms in equations (5.29) and (5.30) with their values at time $t = t^{r+1}$. The resulting system is fully coupled and analogous to the steady state case:

$$\nabla\psi^{r+1} = n^{r+1} - p^{r+1} - N, \tag{5.32}$$

$$\frac{n^{r+1}-n^r}{\delta t^r} = \nabla\cdot[\mu_n^r(\nabla n^{r+1} - n^{r+1}\,\nabla\psi^{r+1})] - R(\psi^r,n^r,p^r), \tag{5.33}$$

$$\frac{p^{r+1}-p^r}{\delta t^r} = \nabla\cdot[\mu_p^r(\nabla p^{r+1} - p^{r+1}\,\nabla\psi^{r+1})] - R(\psi^r,n^r,p^r). \tag{5.34}$$

The analytical expressions for the carrier mobilities and the net recombination rate can be, in general, discretized at the rth time step in any discretization scheme used when the solution at the $(r+1)th$ time step is sought [EnDM83], [Selb84]. This implicit backward Euler method is unconditionally stable for arbitrarily large time steps δt [Selb84] and it has been used in many transient device simulation works. For this method a coupled nonlinear system has to be solved at each time step.

The Poisson equation discretized in time and space domains (for all i, j and using the double index notation) is then given by:

$$\begin{aligned}
&\psi_{i,j-1}^{r+1}\frac{2}{k_{j-1}(k_{j-1}+k_j)} + \psi_{i-1,j}^{r+1}\frac{2}{h_{i-1}(h_{i-1}+h_i)} \\
-\;&\psi_{i,j}^{r+1}\left[\frac{2}{h_{i-1}(h_{i-1}+h_i)} + \frac{2}{k_{j-1}(k_{j-1}+k_j)} + \frac{2}{h_i(h_{i-1}+h_i)} + \frac{2}{k_j(k_{j-1}+k_i)}\right] \\
+\;&\psi_{i+1,j}^{r+1}\frac{2}{h_i(h_{i-1}+h_i)} + \psi_{i,j+1}^{r+1}\frac{2}{k_j(k_{j-1}+k_j)} \\
=\;& n_{i,j}^{r+1} - p_{i,j}^{r+1} - N_{i,j}.
\end{aligned} \tag{5.35}$$

The discretized form of the continuity equation for electrons reads:

$$
\begin{aligned}
& n_{i,j-1}^{r+1}\left[\mu_{i,j-1/2}\frac{Ber(\psi_{i,j-1}^{r+1}-\psi_{i,j}^{r+1})}{k_{j-1}(k_j+k_{j-1})/2}\right]+n_{i-1,j}^{r+1}\left[\mu_{i-1/2,j}\frac{Ber(\psi_{i-1,j}^{r+1}-\psi_{i,j}^{r+1})}{h_i(h_i+h_{i-1})/2}\right] \\
- \; & n_{i,j}^{r+1}\left[+\mu_{i,j-1/2}\frac{Ber(\psi_{i,j}^{r+1}-\psi_{i,j-1}^{r+1})}{k_j(k_j+k_{j-1})/2}+\mu_{i-1/2,j}\frac{Ber(\psi_{i,j}^{r+1}-\psi_{i-1,j}^{r+1})}{h_i(h_i+h_{i-1})/2}\right. \\
+ \; & \left.\mu_{i+1/2,j}\frac{Ber(\psi_{i,j}^{r+1}-\psi_{i+1,j}^{r+1})}{h_i(h_i+h_{i-1})/2}+\mu_{i,j+1/2}\frac{Ber(\psi_{i,j}^{r+1}-\psi_{i,j+1}^{r+1})}{k_j(k_j+k_{j-1})/2}+\frac{1}{\delta t^r}\right] \\
+ \; & n_{i+1,j}^{r+1}\left[\mu_{i+1/2,j}\frac{Ber(\psi_{i+1,j}^{r+1}-\psi_{i,j}^{r+1})}{h_i(h_i+h_{i-1})/2}\right]+n_{i,j+1}^{r+1}\left[\mu_{i,j+1/2}\frac{Ber(\psi_{i,j+1}^{r+1}-\psi_{i,j}^{r+1})}{k_j(k_j+k_{j-1})/2}\right] \\
= \; & R_{i,j}^{r}-\frac{n_{i,j}^{r}}{\delta t^r}. \qquad (5.36)
\end{aligned}
$$

Summary

Spacial discretization of the basic semiconductor equations yields at each mesh node one discrete equation for Poisson equation and both continuity equations, respectively. In the case of a transient problem the discretized continuity equations still contain derivatives of the carrier densities with respect to time. In a general transient analysis problem space discretization and implicit time discretization yield at each instant of time a coupled nonlinear system of algebraic equations which can be written in a simplified form [EnDM83], [Selb84] as:

$$F_\psi(\vec{\psi},\vec{n},\vec{p})=0, \qquad (5.37)$$

$$F_n(\vec{\psi},\vec{n},\vec{p})=0, \qquad (5.38)$$

$$F_p(\vec{\psi},\vec{n},\vec{p})=0, \qquad (5.39)$$

where the nodal values of the unknown variables are condensed into vectors and where all elements of such vectors are evaluated at the time instant t^{r+1}.

In a static (DC) analysis a system of the same form is obtained at each bias point. The transient solution can be considered as a series of steady state solutions; one at each instance of time t^{r+1}. In this case the solutions of the previous time instant can be used as a starting solution to the next time instant. Thus, in both a DC and a transient analysis approximately $3N_x \times N_y \times N_z$ nonlinear equations need to be solved. The number of mesh nodes for a three dimensional problem is usually limited by the computing resources available [TMAS85] to between 10000 and 20000, however, for a mesh of $64 \times 64 \times 64$ nodes, the total number of nodes is ≈ 250000, giving the total number of equations of ≈ 800000. The numerical solution of such a system of equations is the most time consuming task during a numerical device simulation and presents a formidable problem even for a large supercomputer [FNPP85], [TMAS85]. Therefore, it

is very important to choose an efficient numerical algorithm, especially if more than one space dimension has to be considered [EnDM83]. A number of common approaches used to solve this large system of nonlinear equations are briefly described in the following section.

Having demonstrated the equivalence between the DC and transient problems, for simplicity only the DC case will be assumed from now on.

5.3.3 Solution Methods of the Nonlinear System

There are basically two approaches to the solution of the system (5.37)-(5.39) of matrix equations. Most simply, one treats the three equations separately by decoupling the equations and solving them sequentially. This approach was originally proposed by Gummel [Gumm64] with respect to the solution of the steady state problem in one dimension and a number of variations of this method have been developed to improve its computational efficiency [Bell72], [Suth80] and [ScSP81]. The second approach is to solve the complete coupled system simultaneously [BuCo80], [EnDM83], [Selb84].

5.3.3.1 Decoupled Method

In the Gummel method (also known as the decoupled or the sequential method) each of the systems (5.37)-(5.39) is treated separately by neglecting their mutual couplings and solving the three equations one after the other. This sequence is iteratively repeated until self-consistent values for all unknowns are obtained. This method is easily understood if a change of variables is first applied to the system (5.37)-(5.39). The set of unknown ψ, n, p are replaced by the set $\psi, \varphi_n, \varphi_p$. These two sets are related to each other by the Boltzmann relations (5.14), ie $\varphi_n = e^{-\phi_n}$ and $\varphi_p = e^{\phi_p}$. Using the set $\psi, \varphi_n, \varphi_p$ of the unknown variables the scaled semiconductor equations are given by:

$$\nabla^2\psi = \varphi_n e^{\psi} - \varphi_p e^{-\psi} - N, \tag{5.40}$$

$$\nabla \cdot (\mu_n e^{\psi} \nabla \varphi_n) - R(\psi, \varphi_n, \varphi_p) = 0, \tag{5.41}$$

$$\nabla \cdot (\mu_p e^{-\psi} \nabla \varphi_p) - R(\psi, \varphi_n, \varphi_p) = 0. \tag{5.42}$$

The solution is depicted in the form of a flow diagram in Figure 5.4 and it proceeds as follows. Given an estimate $(\psi^g, \varphi_n^g, \varphi_p^g)$ of the solution at Gummel iteration g the Poisson equation:

$$F_\psi(\psi^{g+1}, \varphi_n^g, \varphi_p^g) = -\nabla^2 \psi^{g+1} + \varphi_n^g e^{\psi^{g+1}} - \varphi_p^g e^{-\psi^{g+1}} - N = 0. \tag{5.43}$$

is solved subject to the given boundary conditions for the electrostatic potential, assuming that the quasi-Fermi levels (ϕ_n^g, ϕ_p^g) are known functions of position (the best estimates of the quasi-Fermi levels, and hence the variables φ_n^g and φ_p^g, are assumed to be correct and thus held constant) to obtain a new estimate of the potential distribution ψ^{g+1}. In the second step one of the continuity equations is solved (for example the continuity equation for electrons) using the new approximation for the electrostatic potential (ψ^{g+1}) and assuming constant quasi-Fermi levels $(\phi_p^g$, and hence $\varphi_p^g)$ for the other continuity equation (the continuity equation for holes in this case):

$$F_n(\psi^{g+1}, \varphi_n^{g+1}, \varphi_p^g) = \nabla \cdot (\mu_n e^{\psi^{g+1}} \nabla \varphi_n^{g+1}) - R(\psi^{g+1}, \varphi_n^g, \varphi_p^g) = 0. \tag{5.44}$$

In the third step the second continuity equation is solved under similar assumptions:

$$F_p(\psi^{g+1}, \varphi_n^{g+1}, \varphi_p^{g+1}) = \nabla \cdot (\mu_p e^{-\psi^{g+1}} \nabla \varphi_p^{g+1}) - R(\psi^{g+1}, \varphi_n^{g+1}, \varphi_p^g) = 0, \tag{5.45}$$

where equations (5.44) and (5.45) are two decoupled linear equations (assuming ψ^{g+1} constant) in φ_n^{g+1} and φ_p^{g+1}, respectively. The sequence thus described is performed repeatedly until self-consistent values of the desired accuracy for all unknown variables are obtained. The overall process is shown in Figure 5.4(a).

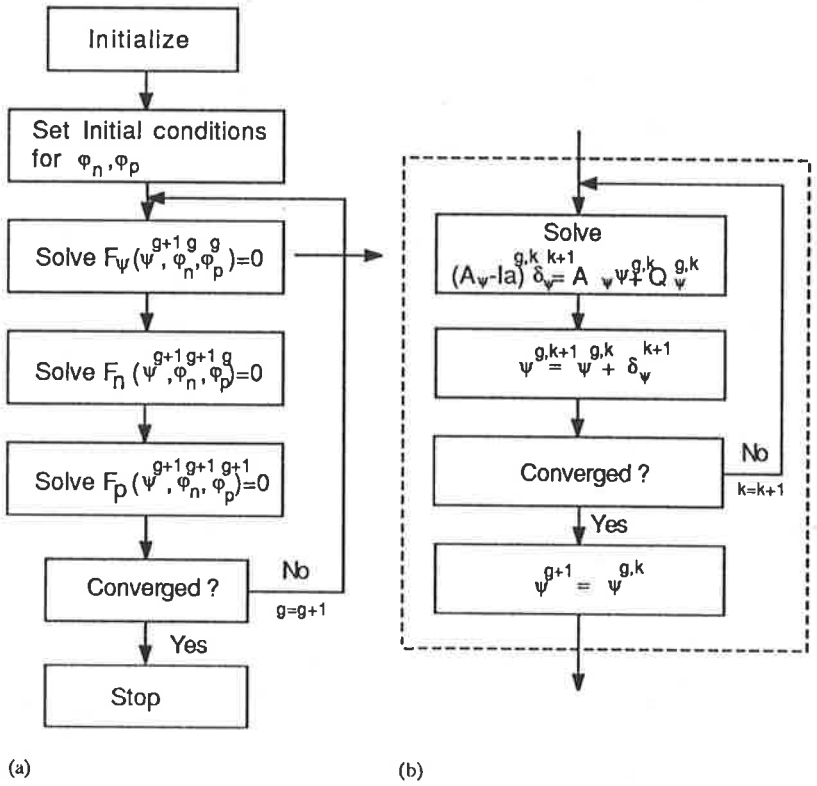


Figure 5.4: Gummel's solution method: (a) outer loop calculation (b) inner loop calculation for the linearized Poisson equation.

The Poisson equation (5.43) is nonlinear in ψ^{g+1} (while φ_n and φ_p are considered known and held constant) and therefore it itself has to be solved iteratively in each step. The nonlinearity is caused by the exponential relationship between ψ and n and p, respectively, via expressions (5.14). Because of its quadratic convergence properties the Newton method is normally chosen for the solution of (5.43). The Poisson equation is linearized by substituting:

$$\psi^{g+1} = \psi^g + \delta_\psi \tag{5.46}$$

into expression (5.43) and using the approximation $e^{\psi^g+\delta_\psi} \approx e^{\psi^g}(1+\delta_\psi)$. The resulting system is linear in δ_ψ and reads:

$$\nabla^2 \delta_\psi - (e^{\psi^g}\varphi_n^g + e^{-\psi^g}\varphi_p^g)\delta_\psi = -\nabla^2 \psi^g + \varphi_n^g e^{\psi^g} - \varphi_p^g e^{-\psi^g} - N, \tag{5.47}$$

or written in a simplified form:

$$\nabla^2 \delta_\psi - a^g \delta_\psi = -\nabla^2 \psi^g + \varphi_n^g e^{\psi^g} - \varphi_p^g e^{-\psi^g} - N, \tag{5.48}$$

where a^g is given by the term in brackets on the left hand side of expression (5.47). In discrete form (5.48) becomes:

$$\begin{aligned} [A_\psi - I\vec{a}^g]\vec{\delta}_\psi &= -A_\psi \vec{\psi}^g + \vec{Q}_\psi^g, \\ A^g \vec{\delta}_\psi &= \vec{b}^g. \end{aligned} \tag{5.49}$$

where I is the identity matrix of the same rank as the matrix A_ψ and the matrix A_ψ and the vector $\vec{Q}_\psi$ are formed as described previously (see text below expressions (5.22) and (5.26)).

Expression (5.49) constitutes a linear system of algebraic equations in the unknown vector $\vec{\delta}_\psi$. This expression can be solved for $\vec{\delta}_\psi$, since at iteration g of the overall solution process, estimates for all other quantities are available. This step constitutes one Newton iteration. Using this value for $\vec{\delta}_\psi$ the quantity $\vec{a}^g$ and the right hand side of expression (5.49) are updated and expression (5.49) is solved again to obtain a better estimate for $\vec{\psi}^{g+1}$. This procedure can be expressed as an iterative process as follows with iteration index k:

$$\begin{aligned} \vec{\delta}_\psi^{k+1} &= (A^{g,k})^{-1}\vec{b}^{g,k}, \\ \vec{\psi}^{g,k+1} &= \vec{\psi}^{g,k} + \vec{\delta}_\psi^{k+1}, \\ A^{g,k+1} &= A_\psi - I\vec{a}^{g,k+1}, \\ \vec{b}^{g,k+1} &= -A_\psi \vec{\psi}^{g,k+1} + \vec{Q}_\psi^{g,k+1}. \end{aligned} \tag{5.50}$$

Expression (5.50) is solved a sufficient number of times until $\vec{\delta}_\psi$ is sufficiently small, in which case:

$$\vec{\psi}^{g,k+1} = \vec{\psi}^{g,k} + \vec{\delta}_\psi^{k+1},$$

$$\approx \vec{\psi}^{g,k},$$
$$= \vec{\psi}^{g+1}. \tag{5.51}$$

This is graphically shown in Figure 5.4(b). Thus, the complete Gummel method is composed of two iteration loops; the inner iteration loop being the solution of the linearized Poisson's equation (iteration index k) and the outer (Gummel) loop comprising the solution of the complete system (iteration index g). If the set (ψ, n, p) of unknowns is used rather than the set $(\psi, \varphi_n, \varphi_p)$ then both continuity equations are also nonlinear in their unknowns and Newton linearization must be applied to all three equations. The major advantage of the decoupled solution method is that it requires a small computer memory and programming effort.

The convergence of this method is strongly dependent on the operating condition of the device, ie the mutual coupling between the equations [BuCo80], [EnDM83], [ShDa86]. In the case of a MOS transistor being biased below threshold, the dominant charge is the space charge of the ionized impurities in the depletion region. Since the concentrations of both carriers are very low, the continuity equations have little effect and the solution of the Poisson equations is essentially all that is required. The convergence is much slower when the device is biased in the linear or saturation regions. In such cases the inversion layer of free carriers results in strong coupling between the Poisson equation and the carrier continuity equations. As a result each solution of the continuity equations alter the carrier concentrations sufficiently to substantially alter the previous solution of the Poisson equation. This is why, in the original form of the decoupled solution method, expression (5.49) was only solved for δ_ψ once per outer iteration loop, ie $k = 1$ and the solution shown in the inner loop (Figure 5.4(b)) was only carried out once per outer loop solution. Using this approach a reduction of approximately 40% in the total number of matrix equation solutions is possible.

A modification of the original Gummel method, that has been shown to work well for the three-dimensional case and for the case where the three equations are strongly coupled together, is a simplification of the Step Solving method [ShDa86]. The equations are solved sequentially in two steps; the inner loop for two strongly coupled equations (either expressions (5.37) and (5.38) or expressions (5.37) and (5.39)) and the outer loop for all three equations, as shown in Figure 5.5.

The decoupled solution method is usually preferred for its simplicity and much smaller computational and memory requirements than the coupled solution method.

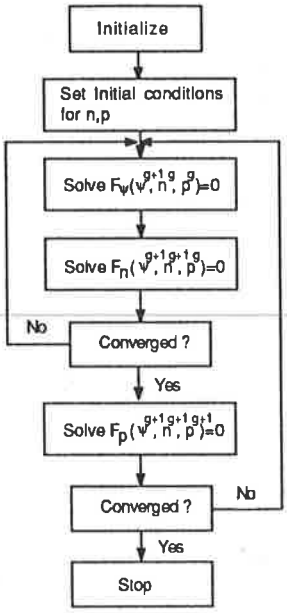


Figure 5.5: A simplified Step Solving method.

5.3.3.2 Simultaneous Method

The other approach of solving the nonlinear system of equations (5.37)–(5.38) is the simultaneous solution method, where the complete system is solved simultaneously. In this way, mutual coupling between equations are taken into account so that the rate of convergence is less dependent on the bias conditions [BuCo80].

Applying Newton's method to the complete nonlinear system of equations, it is required to solve a system of linear equations at each iteration step. Newton's method at the gth iteration is given by:

$$\begin{bmatrix} \frac{\partial F_\psi}{\partial \psi} & \frac{\partial F_\psi}{\partial n} & \frac{\partial F_\psi}{\partial p} \\ \frac{\partial F_n}{\partial \psi} & \frac{\partial F_n}{\partial n} & \frac{\partial F_n}{\partial p} \\ \frac{\partial F_p}{\partial \psi} & \frac{\partial F_p}{\partial n} & \frac{\partial F_p}{\partial p} \end{bmatrix}^g \begin{bmatrix} \delta_\psi^g \\ \delta_n^g \\ \delta_p^g \end{bmatrix} = \begin{bmatrix} -F_\psi(\psi^g, n^g, p^g) \\ -F_n(\psi^g, n^g, p^g) \\ -F_p(\psi^g, n^g, p^g) \end{bmatrix}. \tag{5.52}$$

Although this simultaneous Newton's method is advantageous from purely mathematical point of view, it is more involved with regard to the program structure, amount of computation and the storage requirement [EnDM83], [Selb84]. Due to these disadvantages the method is modified and the system is often solved by the Block-Newton method. Under the assumption that the Jacobian is definite, one can use a classical block iteration scheme (iteration index k) for the solution at gth Newton iteration step

[Selb84], [SeRi84]:

$$\begin{bmatrix} \frac{\partial F_\psi}{\partial \psi} & 0 & 0 \\ \frac{\partial F_n}{\partial \psi} & \frac{\partial F_n}{\partial n} & 0 \\ \frac{\partial F_p}{\partial \psi} & \frac{\partial F_p}{\partial n} & \frac{\partial F_p}{\partial p} \end{bmatrix}^g \begin{bmatrix} \delta_\psi^g \\ \delta_n^g \\ \delta_p^g \end{bmatrix}^{k+1}$$

$$= \begin{bmatrix} -F_\psi(\psi^g, n^g, p^g) \\ -F_n(\psi^g, n^g, p^g) \\ -F_p(\psi^g, n^g, p^g) \end{bmatrix} - \begin{bmatrix} 0 & \frac{\partial F_\psi}{\partial n} & \frac{\partial F_\psi}{\partial p} \\ 0 & 0 & \frac{\partial F_n}{\partial p} \\ 0 & 0 & 0 \end{bmatrix}^g \begin{bmatrix} \delta_\psi^g \\ \delta_n^g \\ \delta_p^g \end{bmatrix}^k . \tag{5.53}$$

Since the coefficient matrix of (5.53) is a block lower triangular, one can decouple the elimination process into three linear systems which have to be solved sequentially:

$$\frac{\partial F_\psi^g}{\partial \psi} \delta_\psi^{g,k+1} = -F(\psi^g, n^g, p^g) - \frac{\partial F_n}{\partial n}^g \delta_n^{g,k} - \frac{\partial F_p}{\partial p}^g \delta_p^{g,k} \tag{5.54}$$

$$\frac{\partial F_n^g}{\partial n} \delta_n^{g,k+1} = -F(\psi^g, n^g, p^g) - \frac{\partial F_\psi}{\partial \psi}^g \delta_\psi^{g,k+1} - \frac{\partial F_p}{\partial p}^g \delta_p^{g,k} \tag{5.55}$$

$$\frac{\partial F_p^g}{\partial p} \delta_p^{g,k+1} = -F(\psi^g, n^g, p^g) - \frac{\partial F_\psi}{\partial \psi}^g \delta_\psi^{g,k+1} - \frac{\partial F_n}{\partial n}^g \delta_n^{g,k+1} \tag{5.56}$$

This method has (like Gummel's method) the advantage that the equations can be solved sequentially.

5.4 Solution of Systems of Linear Algebraic Equations

For any of the linearization procedures described in Section 5.3.3, a large sparse system of linear algebraic equations has to be solved at each iteration of the Newton method. Because of the size of the system of equations, the solution can only be carried out using iterative techniques. The efficiency of a given method depends both on the computational work required to carry out one iteration step and the rate of convergence of this method since this determines the total number of iterations required to achieve a solution of desired accuracy. Thus, the method with the least computational requirements per iteration is not necessarily the fastest one.

A short discussion of the basic iterative methods giving their formulas, computational requirements and emphasizing their adaptability to parallel implementation is provided in the next section. Parallel implementation is important because it allows the solution process to be carried out on a parallel computer and, as discussed in Chapter 2, in order to make the best use of the available capabilities of VLSI technology, it is necessary to perform in parallel as many tasks and computations as possible. This material forms a

prelude to the discussion of the two most widely used solution methods for systems of this size, the conjugate gradient (CG) and the Chebychev's (CM) methods. Since the full mathematical treatment of the iterative solution methods is outside the scope of this work, the interested reader is referred to a number of excellent publications [FoWa60], [Varg62], [HaLY80], [Hage81], [Noye82].

5.4.1 Basic Iterative Methods

Consider the following system of linear algebraic equations:

$$A\vec{u} = \vec{b}, \tag{5.57}$$

where A is a given $m \times m$, symmetric, positive definite matrix, $\vec{b}$ is a given real m component vector, and it is required to solve for the unknown vector $\vec{u}$. For example, such a system is formed for the linearized Poisson equation (5.49) by putting $A = A_\psi - I\vec{a}^g$, $\vec{u} = \vec{\delta}_\psi$ and $\vec{b} = -A_\psi\vec{\psi}^g + \vec{Q}^g_\psi$. The main characteristic of (5.57) is that the matrix A is sparse, as discussed in Section 5.3.1 for the case of the matrix A_ψ. Iterative methods exploit this sparsity of such systems with two objectives. The first is to reduce the amount of storage required during the solution process of the system of equations and the second is to reduce the amount of computation needed to obtain a solution of adequate accuracy.

The basic form of most "classical" iterative methods is a splitting of the coefficient matrix A into a sum of two matrices M and $A - M$. The "splitting" matrix M is chosen in such a way that it can be considered to be an approximation to A but, unlike A, is readily invertible, ie it is a much simpler computational task to solve the linear system $M\vec{\delta_u} = \vec{d}$, than it is to solve (5.57). For example, M may be a diagonal, tri-diagonal or lower triangular matrix. The solution procedure is then based on the repeated solution of the following equation:

$$M\vec{u} = (M - A)\vec{u} + \vec{b}, \tag{5.58}$$

which equation leads to the following iterative process:

$$M\vec{u}^{k+1} = (M - A)\vec{u}^k + \vec{b}, \tag{5.59}$$

where k is the iteration index. Because of the choice of M this system can be solved very economically for each new approximation $\vec{u}^{k+1}$ to the solution vector $\vec{u}$. A family of iteration solution methods is thus formulated, each method being characterized by a particular choice of M.

Although M^{-1} is not usually found explicitly, (5.59) can be written as:

$$\begin{aligned} \vec{u}^{k+1} &= M^{-1}(M-A)\vec{u}^k + M^{-1}\vec{b} \\ &= G\vec{u}^k + \vec{c}, \end{aligned} \tag{5.60}$$

where $G = (I - M^{-1}A)$ is the iteration matrix and $\vec{c} = M^{-1}\vec{b}$. The method described by (5.60) is linear, since neither G nor $\vec{c}$ depend on $\vec{u}^k$ and stationary since neither G nor $\vec{c}$ depends on the iteration index k. The scheme (5.60) will converge if and only if condition (5.61) is satisfied:

$$\rho(G) = \rho(I - M^{-1}A) < 1, \tag{5.61}$$

where $\rho(G)$ is the spectral radius (the value of the largest eigenvalue of the matrix G) of the iteration matrix G [Varg62]. This condition holds when the matrix A is *positive definite* which is the case for a five or seven-point-star discretized partial differential equation [Selb84].

In the following subsections a number of the well known splittings of the coefficient matrix A and the resulting iterative solution methods are considered.

5.4.1.1 The Point-Jacobi Method

The point-Jacobi method is perhaps the simplest iterative method. In order that the method be formally applied it is necessary only that no diagonal element of A vanish. If the matrix A is expressed as a sum:

$$A = D - L - U, \tag{5.62}$$

where D is the diagonal matrix, whose elements equal the diagonal elements of the coefficient matrix A (ie $d_i = a_{i,i}$) and L and U are respectively strictly lower and upper triangular $m \times m$ matrices. For the point-Jacobi iteration method expression (5.60) becomes:

$$\vec{u}^{k+1} = D^{-1}(L+U)\vec{u}^k + D^{-1}\vec{b}, \tag{5.63}$$

where the splitting matrix $M = D$ and the corresponding iteration matrix $G_J = (I - D^{-1}A)$.

Expanding the above matrix equation, for a two-dimensional case, the form of the required algebraic manipulation becomes identified:

$$u_i^{k+1} = \frac{1}{a_{i,i}}\left(a_{i,n}u_{i,n}^k + a_{i,w}u_{i,w}^k + a_{i,s}u_{i,s}^k + a_{i,e}u_{i,e}^k + b_i\right) \quad \text{for all } i. \tag{5.64}$$

A similar equation is obtained for a three-dimensional case. In the above expression the elements $a_{i,n}$, $a_{i,w}$, $a_{i,s}$ and $a_{i,e}$ of the coefficient matrix A relate the unknown u_i, corresponding to a given node i, to the four unknowns $u_{i,n}$, $u_{i,w}$, $u_{i,s}$ and $u_{i,e}$ corresponding to the four neighbouring mesh nodes to the north, west, south and east, respectively, of node i, as described for the case of the discretized Poisson equation in Section 5.3.1. Thus, the calculations are "localized"in the sense that they involve only the unknowns at the mesh nodes immediately adjacent to a given mesh node i. Most commonly, the grid is traversed point by point from left to right in each row and from top to bottom, row by row. It is possible to normalize the initial matrix problem so that the diagonal elements ($a_{i,i}$) are unity [Varg62], [Colg82], and this eliminates the numerous division operations from the iterative process.

According to (5.64) a new estimate of a given unknown u_i is equal to a weighted average of the previous estimates of the four unknown $u_{i,n}$, $u_{i,w}$, $u_{i,s}$ and $u_{i,e}$ corresponding to the four neighbouring mesh nodes. Since the new estimates at any mesh node depend only on the old estimates it is possible to evaluate the new estimates for *all* elements of $\vec{u}$ in parallel. This is a very important feature of this iterative method, since it allows the iterative process to be efficiently carried out on an array processor [CFHR82], [HKSH83], [FoOt84], [Zakh84] which is attractive from the VLSI implementation point of view.

5.4.1.2 The Point-Gauss-Seidel Method

Rewriting expression (5.64), such that the latest estimates of the components of $\vec{u}$ are used as soon as they become available, results in the point-Gauss-Seidel iteration method:

$$\begin{aligned} \vec{u}^{k+1} &= D^{-1}(L\vec{u}^{k+1} + U\vec{u}^{k} + \vec{b}), \\ (D-L)\vec{u}^{k+1} &= U\vec{u}^{k} + \vec{b}, \\ \vec{u}^{k+1} &= (D-L)^{-1}U\vec{u}^{k} + (D-L)^{-1}\vec{b}, \end{aligned} \tag{5.65}$$

where the splitting matrix is given by $M = D - L$ and the iteration matrix is given by $G_{GS} = (D-L)^{-1}U = [I - (D-L)^{-1}A]$. In this case the matrix M consists of all the nonzero elements of the matrix A which lie on or below the main diagonal, the remaining elements being zero.

Expanding the above matrix equation identifies the form of the required algebraic manipulation:

$$u_i^{k+1} = \frac{1}{a_{i,i}}\left(a_{i,n}u_{i,n}^{k+1} + a_{i,w}u_{i,w}^{k+1} + a_{i,s}u_{i,s}^{k} + a_{i,e}u_{i,e}^{k} + b_i\right) \quad \text{for all } i. \tag{5.66}$$

According to (5.66), the new estimates for the unknowns corresponding to the mesh nodes which lie to the north and west of the mesh node i are used, as soon as they become available, to update the unknown u_i, when a two-dimensional grid is traversed from left to right, top to bottom. The point-Gauss-Seidel method converges about twice as fast as the point-Jacobi method, but it is sequential in nature and thus not suitable for an efficient array processor implementation.

5.4.1.3 The Point-Successive Overrelaxation (SOR) Method

The point-SOR method combines the point-Gauss-Seidel method with an acceleration procedure to enhance the rate of convergence. The new estimate u_i^{k+1} is calculated as a weighted average of a term ($\hat{u}_i^k$) obtained using the point-Gauss-Seidel method, ie expression (5.66), and the current estimate u_i^k according to the following expression:

$$u_i^{k+1} = \omega(\hat{u}_i^{k+1} - u_i{}^k) + u_i^k. \tag{5.67}$$

In matrix form the point-SOR method becomes:

$$\begin{aligned} \vec{u}^{k+1} &= \omega\left[D^{-1}(L\vec{u}^{k+1} + U\vec{u}^k + \vec{b}) - \vec{u}_k\right] + \vec{u}^k, \\ (D-\omega L)\vec{u}^{k+1} &= [\omega U + (1-\omega)D]\,\vec{u}^k + \omega\vec{b}, \\ \vec{u}^{k+1} &= (D-\omega L)^{-1}[\omega U + (1-\omega)D]\,\vec{u}^k + (D-\omega L)^{-1}\omega\vec{b}, \end{aligned} \tag{5.68}$$

where ω is the relaxation parameter. The splitting matrix is given by: $M = \frac{1}{\omega}D - L$, and the corresponding iteration matrix is given by:

$$\begin{aligned} G_{SOR} &= (D-\omega L)^{-1}[\omega U + (1-\omega)D], \\ &= \left[I - \left(\frac{1}{\omega}D - L\right)^{-1} A\right]. \end{aligned} \tag{5.69}$$

The point-SOR method may be visualized as being carried out in two steps: one computes the new approximation to $\vec{u}$ using the point-Gauss-Seidel method, followed by computing u_i^{k+1} according to (5.67). For $\omega = 1$ the point-SOR method reduces to the point-Gauss-Seidel method, ie $G_{SOR} = G_{GS}$.

The rate of convergence of the point-SOR method is strongly dependent on the choice of the relaxation parameter. For systems where the largest eigenvalue of the iteration matrix G_{SOR} is known, as is the case for example for uniform discretization grids, it is possible to calculate its value analytically. For a general problem (one involving a nonuniform grid), however, the value of the largest eigenvalue is not known and the optimum relaxation parameter, ie one that gives the fastest convergence, has to estimated iteratively. A possible algorithm (denoted here omega) as proposed in [Colg82] is presented below:

Algorithm: omega

1. Apply a few steps of the point-SOR method with $\omega = 1$ and then use the ratio of consecutive pseudo-residuals:

$$R_\delta^k = \frac{\|\vec{\delta_u}^k\|}{\|\vec{\delta_u}^{k-1}\|}, \tag{5.70}$$

as an estimate for $\rho(G_{SOR})$. In expression (5.70) $\vec{\delta_u}^k = \vec{u}^{k+1} - \vec{u}^k$ and

$$\|\vec{d}\| = (\vec{d}, \vec{d})^{1/2} = \left(\sum_{i=1}^{i=m} d^2\right)^{1/2} \tag{5.71}$$

is the 2-norm of a given m element vector $\vec{d}$;

2. Use this value and the current value of ω to calculate $\rho(G_J)$, ie the spectral radius of the corresponding Jacobi method iteration matrix, from:

$$\rho(G_J) = \frac{\rho(G_{SOR}) + \omega - 1}{\omega\sqrt{\rho(G_{SOR})}}; \tag{5.72}$$

3. A new estimate of ω is then given by:

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(G_J)}}. \tag{5.73}$$

Thus, in general the point-SOR method is complicated by the need to evaluate vector inner products in the calculations used to estimate an improved value of the relaxation parameter ω. The method in its original form is sequential in nature, ie a new estimate for u_i cannot be calculated without first calculating the new estimates for the two elements $u_{i,n}$ and $u_{i,w}$, however, it is possible with proper ordering of the unknowns to derive a parallel version of the point-SOR iterative method and thus making it more attractive for an array processor implementation.

5.4.1.4 Parallel Version of the Point-SOR Method

The sequential nature of the point-SOR and the point-Gauss-Seidel methods stem from the fact that when the grid is traversed from left to right, top to bottom, the values $u_{i,n}^{k+1}$ and $u_{i,w}^{k+1}$ must be known before u_i^{k+1} can be computed.

In the normal approach the unknowns are ordered by rows and columns as shown schematically in Figure 5.6(a). However, by ordering the grid points in what is known as a *red-black* form, as shown in Figure 5.6(b), both the SOR and the Gauss-Seidel methods

can be made suitable for parallel computation [Hage81], [Colg82]. This node ordering results in a corresponding partitioning of the unknown vector $\vec{u}$ into two subvectors: one containing elements corresponding to *red* nodes and the other containing elements corresponding to *black* nodes. Any component of the unknown vector belonging to the red group can then be expressed in terms of the components of the unknown vector belonging to the black group and any component of the unknown vector belonging to the black group depends only on the values of the components of the unknown vector belonging to the red group.

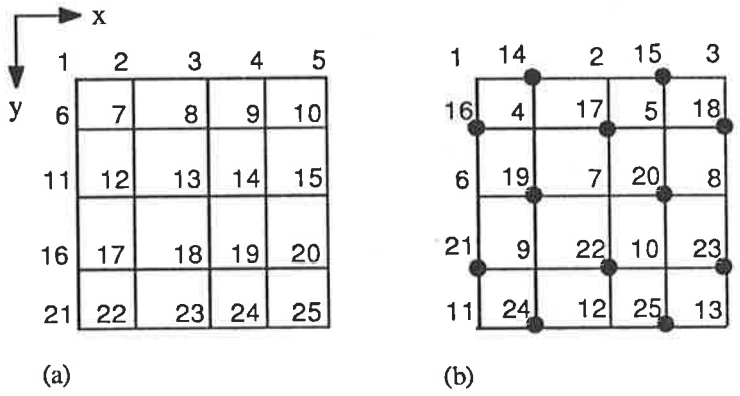


Figure 5.6: Node ordering schemes: (a) conventional, (b) red-black, where the unmarked nodes belong to the red and the nodes marked with black circles belong to the black group.

An SOR iteration is now performed in two stages: first all the red unknowns are updated (red stage) using the values of the black unknowns from the previous iteration step and then all the black unknowns are updated (black stage) using the new estimates of the red unknowns. A *red stage* corresponding to the node ordering of Figure 5.6(b) (where the total number of unknowns $m = 25$) is given by:

$$\hat{u}_i^{r,k+1} = \frac{1}{a_{i,i}}\left(a_{i,n}u_{i,n}^{b,k} + a_{i,w}u_{i,w}^{b,k} + a_{i,s}u_{i,s}^{b,k} + a_{i,e}u_{i,e}^{b,k} + b_i^r\right), \tag{5.74}$$

$$u_i^{r,k+1} = u_i^{r,k} + \omega(\hat{u}_i^{r,k+1} - u_i^{r,k}),\ i = 1, 2, \ldots, m/2 = 13. \tag{5.75}$$

As before, ie expression (5.64), the elements $a_{i,n}$, $a_{i,w}$, $a_{i,s}$ and $a_{i,e}$ of the coefficient matrix A relate the unknown u_i, corresponding to a given node i, to the four unknowns $u_{i,n}$, $u_{i,w}$, $u_{i,s}$ and $u_{i,e}$ corresponding to the four neighbouring nodes to the north, west, south and east, respectively, of the node i. However, in the case of expression (5.74) the unknown u_i belongs to the red group of unknowns and, thus, is denoted u_i^r, and the four unknowns corresponding to the four neighbouring nodes belong to the black group of unknowns and therefore a "b" is included in their superscripts.

The corresponding *black stage* is given by:

$$\hat{u}_i^{b,k+1} = \frac{1}{a_{i,i}}\left(a_{i,n}u_{i,n}^{r,k+1} + a_{i,w}u_{i,w}^{r,k+1} + a_{i,s}u_{i,s}^{r,k+1} + a_{i,e}u_{i,e}^{r,k+1} + b_i^b\right), \quad (5.76)$$

$$u_i^{b,k+1} = u_i^{b,k} + \omega(\hat{u}_i^{b,k+1} - u_i^{b,k}),\ i = m/2+1 = 14,\ldots,m = 25. \quad (5.77)$$

Each stage can be carried out in parallel and the two stages performed sequentially, these actions constituting one iteration step of the parallel point-SOR method.

5.4.1.5 Basic Block Iterative Methods

In the form presented in the previous sections, the basic iterative methods involve expressions that explicitly relate only *one* component of the approximate solution vector ($\vec{u}^{k+1}$) to the unknowns at the neighbouring nodes, hence the name *point*-Jacobi method, for example. In general, a faster rate of convergence is achieved if expressions that permit the estimates for a group of unknowns to be simultaneously determined are employed. Iterative methods that simultaneously update the values of the unknowns corresponding to an entire *line* of nodes are of special interest. Such methods use the relative ease with which tridiagonal matrices can be inverted.

For these methods the entire unknown vector ($\vec{u}$) is divided into subvectors ($\vec{u}_i$) containing the unknowns corresponding to a complete line i of nodes in the solution region, as shown in Figure 5.7. (Note that now the index i numbers lines of nodes in the xy plane, rather than the individual nodes, as was the case for the two-dimensional mesh of Figure 5.6). The individual lines are composed of nodes running in the depth direction. The known vector $\vec{b}$ and the coefficient matrix A are subdivided accordingly.

An example of a three-dimensional system of nodes with red-black line ordering is shown in Figure 5.7(a). In the figure there are 16 lines of nodes running in the depth (z) direction, with each line being composed of 4 nodes. The adopted notation corresponding to this partitioning of the unknowns is shown in part (b) of the figure. For example, the unknown subvector $\vec{u}_3$, which contains the unknowns assigned to node line 3, contains 4 elements, denoted u_1^3, u_2^3, u_3^3 and u_4^3. In a general case it is assumed that each line contains z nodes and there are $x \times y$ lines of nodes in the xy plane, ie the total number of unknowns is $m = xyz$. Consequently, each unknown subvector $\vec{u}_i$, corresponding to a given line i of nodes, contains z elements, denoted u_j^i, $j = 1,\ldots,z$.

Line iterative methods, analogous to the point iterative methods described in Section 5.4.1, when applied to this three-dimensional case, implicitly relate the unknowns corresponding to a *complete* line of nodes to the unknowns assigned to the four neighbouring lines of nodes. For example, the Jacobi method, expression (5.64), takes the

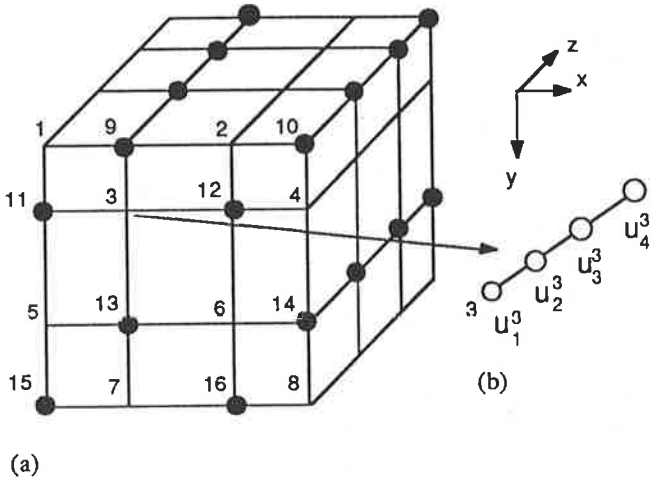


Figure 5.7: (a) A three-dimensional system of nodes, (b) adopted notation.

following form when the unknowns are grouped into lines:

$$\begin{aligned} A_{i,i}\vec{u}_i^{k+1} &= \left(A_{i,n}\vec{u}_{i,n}^{k} + A_{i,w}\vec{u}_{i,w}^{k} + A_{i,s}\vec{u}_{i,s}^{k} + A_{i,e}\vec{u}_{i,e}^{k} + \vec{b}_i\right) \\ A_{i,i}\vec{u}_i^{k+1} &= \vec{v}_i^{k}, \quad \text{for all } i, \end{aligned} \tag{5.78}$$

where $A_{i,i}$ is a tridiagonal $z \times z$ matrix, matrices $A_{i,n}$, $A_{i,e}$, $A_{i,e}$ and $A_{i,w}$ are block diagonal $z \times z$ matrices, $\vec{u}_i$ is a $z \times 1$ vector containing the unknowns corresponding to one line of nodes and $\vec{v}_i$ is a $z \times 1$ vector. Expression (5.78) applied to the example of Figure 5.7 can be written in explicit form for the unknowns corresponding to line $i = 3$ of nodes, where $\vec{u}_{3,n} = \vec{u}_9$, $\vec{u}_{3,w} = \vec{u}_{11}$, $\vec{u}_{3,s} = \vec{u}_{13}$ and $\vec{u}_{3,e} = \vec{u}_{12}$, as follows:

$$\begin{bmatrix} a_{1,1}^3 & a_{1,2}^3 & & \\ a_{2,1}^3 & a_{2,2}^3 & a_{2,3}^3 & \\ & a_{3,2}^3 & a_{3,3}^3 & a_{3,4}^3 \\ & & a_{4,3}^3 & a_{4,4}^3 \end{bmatrix} \begin{bmatrix} u_1^{3,k+1} \\ u_2^{3,k+1} \\ u_3^{3,k+1} \\ u_4^{3,k+1} \end{bmatrix} = \begin{bmatrix} v_1^{3,k} \\ v_2^{3,k} \\ v_3^{3,k} \\ v_4^{3,k} \end{bmatrix}. \tag{5.79}$$

In expression (5.79) $a_{p,q}^i$ represents the relationship between the unknown u_p^i to the unknown u_q^i of the subvector $\vec{u}_i$. The corresponding right hand side subvector is given by:

$$\begin{bmatrix} v_1^{3,k} \\ v_2^{3,k} \\ v_3^{3,k} \\ v_4^{3,k} \end{bmatrix} = \begin{bmatrix} a_{1,n}^3 & & & \\ & a_{2,n}^3 & & \\ & & a_{3,n}^3 & \\ & & & a_{4,n}^3 \end{bmatrix} \begin{bmatrix} u_1^{9,k} \\ u_2^{9,k} \\ u_3^{9,k} \\ u_4^{9,k} \end{bmatrix} + \begin{bmatrix} a_{1,w}^3 & & & \\ & a_{2,w}^3 & & \\ & & a_{3,w}^3 & \\ & & & a_{4,w}^3 \end{bmatrix} \begin{bmatrix} u_1^{11,k} \\ u_2^{11,k} \\ u_3^{11,k} \\ u_4^{11,k} \end{bmatrix}$$

$$+\begin{bmatrix} a^3_{1,s} & & & \\ & a^3_{2,s} & & \\ & & a^3_{3,s} & \\ & & & a^3_{4,s} \end{bmatrix}\begin{bmatrix} u_1^{13,k} \\ u_2^{13,k} \\ u_3^{13,k} \\ u_4^{13,k} \end{bmatrix}+\begin{bmatrix} a^3_{1,e} & & & \\ & a^3_{2,e} & & \\ & & a^3_{3,e} & \\ & & & a^3_{4,e} \end{bmatrix}\begin{bmatrix} u_1^{12,k} \\ u_2^{12,k} \\ u_3^{12,k} \\ u_4^{12,k} \end{bmatrix}+\begin{bmatrix} b_1^{3,k} \\ b_2^{3,k} \\ b_3^{3,k} \\ b_4^{3,k} \end{bmatrix}. \tag{5.80}$$

In the above matrix equation $a^3_{1,e}$, for example, represents the relationship between the first element (u_1^3) of the subvector $\vec{u}_3$, containing the unknowns corresponding to the line of interest, ie $i = 3$, to the first element (u_1^{12}) of the subvector ($\vec{u}_{12}$) containing the unknowns corresponding to the line of nodes immediately east (right) of line 3.

A line iterative method then involves, at each iteration step, solutions of i tridiagonal linear systems of equations such as (5.78). In general one iteration using a line iterative method is equivalent to a number of (k_l say) iterations of the corresponding point iterative method. If at each iteration the tridiagonal linear systems of equations can be solved at a time cost which is less than the cost of k_l iterations of the corresponding point iterative method, then a computational gain is achieved. Efficient solution methods of general tridiagonal linear systems are well known [Ston75], [Axel85] and one of them, based on a LU decomposition of the tridiagonal coefficient matrix will be described in more detail in Section 6.3.1.

5.4.2 Stopping Criterion

In general, the iteration process is terminated when a sufficiently accurate estimate of the unknown vector ($\vec{u}$) has been found. Ideally, one might wish to guarantee that after a sufficient number of iterations:

$$\frac{\| \vec{u}^{k+1} - \vec{u} \|}{\| \vec{u} \|} < \xi, \tag{5.81}$$

where $\xi = 10^{-6}$, for example, and $\vec{u}$ is the *exact* solution of (5.57). However, $\| \vec{u} \|_2$ cannot be measured precisely, since $\vec{u}$ is not known. It is possible to find the pseudo-residual $\vec{\delta_u}^k = \vec{u}^{k+1} - \vec{u}^k$ but this can only be used effectively if an accurate estimate of the spectral radius of the iteration matrix is available. The commonly used stopping test of the form:

$$\frac{\| \vec{\delta}_u^k \|}{\| \vec{u}^{k+1} \|} < \xi \tag{5.82}$$

does not, in general, guarantee the accuracy specified in expression (5.81) [Colg82], [SeRi84]. A stopping test which is very nearly equivalent to (5.81) is given by:

$$\frac{\| \vec{\delta}_u^k \|}{[1-\rho(G)] \| \vec{u}^k \|} < \xi. \tag{5.83}$$

Thus, it is necessary to calculate two vector inner products in order to carry out the stopping test given by (5.83). In addition, as discussed in Section 5.4.1.3, if the value of $\rho(G)$ is not known it must be estimated using the ratio of two successive pseudo-residuals (see algorithm **omega**) and this involves the calculation of an additional vector inner product.

5.4.3 Convergence Acceleration of Iterative Methods

Several methods have been developed for the purposes of acceleration of the convergence of basic iterative methods. The most widely used include the Chebychev acceleration (CM) and the conjugate gradient (CG) acceleration methods. These will be described in more detail in the following subsections.

5.4.3.1 Chebychev Acceleration Method

The Chebychev acceleration is a second degree semi-iterative method with respect to a basic iterative method given by expression (5.60). It is given by [HaLY80], [Hage81], [Colg82]:

$$\vec{u}^{k+1} = \rho^{k+1}(G\vec{u}^k + \vec{c}) + (1-\rho^{k+1})\vec{u}^{k-1}, \tag{5.84}$$

where the various constants are given by:

$$\begin{aligned} \rho^1 &= 1, \quad \rho^2 = \left(1 - \frac{1}{2}\sigma^2\right)^{-1}, \quad \rho^k = \left(1 - \frac{1}{4}\sigma^2\rho^k\right)^{-1} \quad k \geq 2, \\ \sigma &= (MG - mG)/(2 - MG - mG), \end{aligned} \tag{5.85}$$

and where MG and mG are, respectively, the largest and the smallest eigenvalues of the iteration matrix G. Any basic iterative method described in the previous section can be accelerated using the Chebychev acceleration method. The corresponding accelerated method is then similar to the SOR method in that it can be regarded as being carried out in two steps: the particular basic iterative method is used to determine $\hat{\vec{u}}^{k+1}$, where:

$$\hat{\vec{u}}^{k+1} = G\vec{u}^k + \vec{c} \tag{5.86}$$

which is then used in (5.84) to determine $\vec{u}^{k+1}$. Once the vector $\hat{\vec{u}}^{k+1}$ is known expression (5.84) becomes a vector equation involving vector-scalar multiplications and vector additions, all of which can be efficiently carried out in parallel on an array processor.

The main problem with the Chebychev acceleration method is that mG and MG must be known. Adaptive techniques similar to that described for the estimation of the relaxation factor ω also exist for the Chebychev method, where MG and mG are not known a priori. (In most cases a good estimate for MG is important but setting $mG = -MG$ is sufficient.)

5.4.3.2 Conjugate Gradient Acceleration Method

In 1952 Hestenes and Stiefel [HeSt52] proposed the conjugate gradient method for solving the system of linear algebraic equations $A\vec{u} = \vec{b}$. One of its important properties is that in the absence of round-off errors it will converge in at most m iterations, m being the order of A. For large problems, eg $m = 100000$, this rate of convergence is not acceptable. For this reason this method is used most commonly as a means of accelerating the rate of convergence of some basic iterative method, given by expression (5.60) and characterized by the splitting matrix M. The CG method applied to the basic iterative method is called the preconditioned conjugate gradient method (PCG) and it is given as follows.

Algorithm: PCG

Let $\vec{u}^0$ be a given starting approximation to the solution vector $\vec{u}$.

1. Initialize: Compute the residual;

$$\vec{r}^0 = \vec{b} - A\vec{u}^0; \tag{5.87}$$

Solve the linear system:

$$M\hat{\vec{r}}^0 = \vec{r}^0 \tag{5.88}$$

for $\hat{\vec{r}}^0$ and set $\vec{p}^0 = \hat{\vec{r}}^0$.

Thereafter, for $k = 1, 2 \ldots$ iteratively compute steps (2) to (4) until the given stopping criterion is satisfied.

2. Compute:

$$\alpha^k = \frac{(\hat{\vec{r}}^k, \vec{r}^k)}{(\vec{p}^k, A\vec{p}^k)}, \tag{5.89}$$

$$\vec{u}^{k+1} = \vec{u}^k + \alpha^k \vec{p}^k, \tag{5.90}$$

$$\vec{r}^{k+1} = \vec{r}^k - \alpha^k A\vec{p}^k; \tag{5.91}$$

3. Solve the linear system:

$$M\hat{\vec{r}}^{k+1} = \vec{r}^{k+1} \quad \text{for } \hat{\vec{r}}^{k+1}; \tag{5.92}$$

4. Compute:

$$\beta^k = \frac{(\vec{r}^{k+1}, \hat{\vec{r}}^{k+1})}{(\vec{r}^k, \hat{\vec{r}}^k)}, \tag{5.93}$$

$$\vec{p}^{k+1} = \hat{\vec{r}}^{k+1} + \beta^k \vec{p}^k, \tag{5.94}$$

where for any two vectors, $\vec{x}$ and $\vec{y}$, $(\vec{x}, \vec{y})$ denotes their inner product.

Each iteration of the preconditioned conjugate gradient method requires two vector inner products, three scalar-vector multiplications, three vector additions, a matrix-vector multiplication and a solution to the linear system (5.92).

Preconditioning, ie the choice of the splitting matrix M in (5.92), can be interpreted as an *inner iteration* to the outer iteration comprising the overall solution process of the conjugate gradient accelerated method [DuGR79], [HaLY80]. For example, for the coefficient matrix A with the main diagonal scaled to unity, such that $A = I - G$ the Jacobi iteration is of the form:

$$\vec{u}^{k+1} = G\vec{u}^k + \vec{b}, \tag{5.95}$$

for which:

$$\hat{\vec{r}}^k = \vec{u}^{k+1} - \vec{u}^k = G\vec{u}^k - \vec{u}^{k+1} + \vec{b}, \tag{5.96}$$

$$= \vec{b} - A\vec{u}^k,$$

$$= \vec{r}^k. \tag{5.97}$$

Comparing the above expression with (5.92) one gets $M^{-1} = I$, which corresponds to a preconditioning obtained by simply to scaling the linear system such that the elements on the main diagonal of A are equal to one. Grouping two successive Jacobi iteration into one preconditioning step one gets:

$$\vec{u}^{k+2} = G^2\vec{u}^k + G\vec{b} + \vec{b}, \tag{5.98}$$

for which:

$$\hat{\vec{r}}^k = \vec{u}^{k+2} - \vec{u}^k = G^2\vec{u}^k - \vec{u}^{k+2} + G\vec{b} + \vec{b},$$

$$= (I + G)(\vec{b} - A\vec{u}^k),$$

$$= (I + G)\vec{r}^k \tag{5.99}$$

and in this case $M^{-1} = (I + G)$, which constitutes a better approximation to A^{-1}, than $M^{-1} = I$.

The most popular preconditioning method has been the incomplete Cholesky decomposition in which the splitting matrix M is written in the form $M = LL^T$, where the matrix L is a lower triangular matrix [Kers77], [MevV77], [Gust78], [Mant80], [MevV81]. The solution to the linear system $M\hat{\vec{r}}^{k+1} = \vec{r}^{k+1}$ is then obtained by forward and back substitutions which together with the incomplete factorization process are inherently recursive in nature (previous components of $\hat{\vec{r}}^{k+1}$ must be known in the solution process before the next component can be calculated) and thus not applicable to vector or parallel computations.

In conclusion it should be noted that with the advent of vector and parallel computers preconditioning methods which allow the solution process to be carried out on a vector or a parallel computer have become of special interest. Such new preconditioning methods may require many more iterations than the classical ICCG (conjugate gradient with incomplete Cholesky decomposition used as a preconditioner) but they still require less cpu time on a vector or parallel computer since the computation at each iteration step is performed in a much shorter time [DuGR79], [Vors82], [Adam85], [Nash85], [Seag86].

5.4.4 Systems of Linear Equations with a Nonsymmetric Matrix A

The condition that the coefficient matrix A be symmetric is not fulfilled for the matrices arising from the discrete approximations to the continuity equations if the sets (ψ, n, p) or (ψ, ϕ_n, ϕ_p) of unknowns are used. Furthermore, the coefficient matrix of the set of equations (5.52) formulated for the coupled solution method is also nonsymmetric [TMAS85]. Therefore, before the Chebychev or the conjugate gradient acceleration methods may be applied, the system to be solved must be transformed into a symmetric positive definite form. Strictly speaking it is necessary to find a nonsingular matrix W, such that the matrix $W(I - G)W^{-1}$ is symmetric positive definite, where G is the iteration matrix of the corresponding basic iterative method, (5.60), to be accelerated [Hage81], [Colg82].

However, a form of the Chebychev acceleration method which can be applied directly to iterative methods involving nonsymmetric matrices also have been developed, for example in [Mant77], [Mant78], [Vors81]. Furthermore, the general form of the conjugate gradient method, the biconjugate gradient method (BCG) [Flet75], [AxGu79], [EiES83], [Saad84], or the conjugate residual (CR) method [AxGu79], [EiES83] can be

directly used to solve a linear system involving a nonsymmetric coefficient matrix A. In [TMAS85] both of these methods have been successfully applied to simulate MOS transistor characteristics using the numerical solution of the basic semiconductor equations in three dimensions.

The iterative solution techniques which can be directly used to solve nonsymmetric systems of linear equations only involve the arithmetic manipulations that have been identified in Section 5.4.3, but in some cases require additional storage. Thus, the discussions of this and the next chapter which mainly concentrate on symmetric linear systems of algebraic equations are also applicable to the more general nonsymmetric linear systems.

5.5 Summary

An accurate simulation of small geometry MOS transistors can only be achieved with a numerical solution of the basic semiconductor equations. In this chapter these partial differential equations have been reviewed. It has been shown how space and time discretization lead to sets of algebraic, nonlinear equations. The solution of such equations is the most time consuming process in the simulation of MOS transistors using numerical methods.

Because of the size and the nonlinear nature of the algebraic equations, the solution is generally carried out using iterative techniques. The best, from the convergence point of view, solution methods involve red-black partitioning of the unknowns and the application of a convergence acceleration techniques, such as the Chebychev method or the conjugate gradient method, to a basic iterative method, eg Jacobi, Gauss-Seidel or SOR.

The calculations involved in solving the algebraic equations by an iterative solution method are very regular and localized, making them very suitable for implementation on two-dimensional array processors. However, a calculation of the general stopping test, an adaptive estimation of the spectral radius of the iteration matrix, needed for the calculation of the optimum relaxation parameter for the SOR method, and of the acceleration parameters for the Chebychev method, as well as the calculations of iteration parameters for the conjugate gradient method, all require calculations of vector inner products. Such calculations are not efficiently carried out on a two-dimensional array processor.

The following chapter examines a possible architecture of a parallel computer, developed

in this work, which can efficiently perform the various calculations identified in this chapter.

Chapter 6

The TIME Machine Architecture

6.1 Introduction

As discussed in Chapter 5, the problem of numerical device simulation involves repetitive solutions of large, sparse systems of linear algebraic equations of the form $A\vec{u} = \vec{b}$. These tasks account for the major part of the computing resources used; both in terms of cpu cycles and storage. Therefore, the efficient solution of such systems of equations is a key factor in cost effective numerical device simulation process. The numerical algorithms normally used involve a high degree of parallelism, making them attractive for VLSI implementation. The current VLSI technology offers a cost effective means of performing all the structured arithmetic manipulations inherent in the solution process in a time efficient manner. The main problem is to devise a hardware architecture that can efficiently exploit that parallelism and at the same time satisfies the constraints that VLSI design imposes upon it.

The architecture of an array processor for an efficient solution of systems of algebraic linear equations, arising from the finite difference discretization of second order partial differential equations, is presented in this chapter. The main objective of this chapter is not to give a detailed design specification of the complete architecture (a task well outside the scope of this work) but rather present *free form* descriptions of the array processor and the individual processing element, in order to demonstrate how the advantages of VLSI have been exploited, and how the most important design decisions involving the various design tradeoffs (discussed in Chapter 2) have been made.

The numerical solution of partial differential equations was the principal motivation for the development of the earliest parallel processing machines [Zakh84]. The architectural requirements of such machines are well understood and many different machines

have been proposed and designed, eg ILLIAC IV [BBKK68], PEPE [EvTr73], CLIP4 [Duff76], MPP [Fung77], ASPRO [Meil81], DAP [CFHR82], PACS [HKSH83]. Most of the hardware architectures which have been developed over the years are based around a two-dimensional array of identical processors with connections only to nearest neighbours, as first proposed by Unger [Unge58].

The question of how the solution to a system of linear algebraic equations can be carried out on such an architecture, using the point-SOR method with red-black node ordering, has been answered many times. Answers can be found for example in [Scho75], [AdOr82], [CFHR82], [HKSH83], [FoOt84]. However, such discussions concentrate on a high architectural level, and no attention is given to the data manipulations within individual processing elements. In addition, no consideration is given of the computation and communication costs involved in performing the stopping test or in calculating the optimum relaxation parameter.

Similar assumptions to those made in the cited references are made in this discussion, but the abovementioned issues of how to perform the calculations for the stopping test and the relaxation parameters are also addressed. A modification to the conventional two-dimensional array of processing elements is then presented, which diminishes the communication costs involved in the calculation of the stopping test and the acceleration parameters. It is also demonstrated how the Chebychev or conjugate gradient acceleration methods can be carried out. Computations carried out by individual processing elements are also considered and a description of a single chip processing element, termed here the Mathematical Processor (mP), is subsequently presented. A complete specification of the processor is well outside the scope of this work, as previously pointed out, and instead the Mathematical Processor is specified in sufficient detail to enable a complete design, in the symbolic domain, of its two most complex and difficult MCM level modules to be carried out. The functional specifications for these modules are given in the final parts of this chapter. These two modules are, as mentioned previously, the Floating Point Adder and the Floating Point Multiplier, the novel architectural designs of which are the subjects of the next two chapters.

6.2 The Two-Dimensional Array Architecture

Figure 6.1 shows a block diagram of the basic parallel computer consisting of a two-dimensional array of $x \times x$ identical processing elements (PE), where the PEs occupy points in the plane with integer coordinates j, h $(j, h = 1 \ldots x)$ and two input-output buffers located along two opposite sides of the array. Each internal PE_i (where the

index i is given by $i = (j-1)x + h$ and it numbers the individual PEs in each row of the array from left to right starting from the top row of the array) is directly connected to its four nearest neighbours in the horizontal and vertical directions. With reference to Figure 6.1, the PEs above, to the left, below and to the right of a given PE_i are denoted $PE_{i,n}$ (north neighbour), $PE_{i,w}$ (west neighbour), $PE_{i,s}$ (south neighbour) and $PE_{i,e}$ (east neighbour), respectively. The PEs occupying the left hand side column of the array ($h = 1$) do not have west neighbours and the PEs occupying the right hand side column of the array ($h = x$) do not have east neighbours. Such PEs will be termed *boundary* PEs. Similarly the PEs occupying the top row and the bottom row do not have north or south neighbours, respectively, and instead they are connected to the input-output buffers. Such PEs will be termed *input-output* PEs (note that these PEs also belong to the group of boundary PEs).

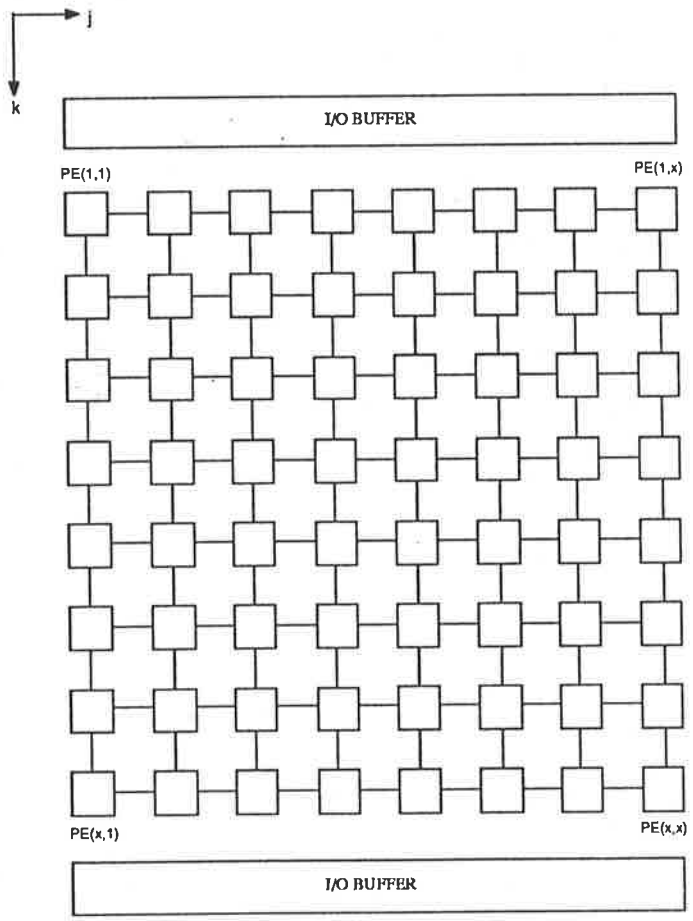


Figure 6.1: Two-dimensional array of identical processing elements.

The logical and physical architectures of the individual PE will be discussed in some detail in Section 6.4. At this stage it is sufficient to regard each PE as a single chip microprocessor, capable of arithmetic and logic operations containing an on-chip memory for program and data storage, and four input-output (I/O) ports for communications

with neighbouring PEs. All internal PEs execute the same code, with the boundary PEs containing modified code subject to the relevant boundary condition. There is a common clock and PEs containing instructions to execute, execute them simultaneously. The communications between PEs are carried out in "data driven" mode [Snyd84], [Eshr85]: reads wait until the arrival of data, writes take place immediately unless they would cause "buffer overflow" of the receiving PE, in which case they idle until space is available.

Two input-output buffers interface the array processor to a general purpose host computer. The host computer is responsible for user interface and for carrying out all the data manipulation tasks other than those performed by the array processor. The function of the array processor is to carry out the solution of the partial differential equations subject to a set of boundary conditions. This task is broken down, as discussed in Chapter 5, into repetitive solutions of sets of linear algebraic equations of the form $A\vec{u} = \vec{b}$. The input-output buffers are used to load the array processor with the necessary data and to unload the final solution. The specific inputs required by the array processor are the elements of the coefficient matrix, A_ψ, which defines the structure of the problem space domain discretization mesh, the net impurity concentration, N_i, at each node of the discretized space domain and the boundary conditions. The way in which these data are distributed between the individual PEs and the way in which the calculations are carried out will be discussed in the following section.

The design of a suitable communication channel between the host computer and the array processor is a problem of its own and is not addressed in this thesis. It is recognized, however, that any excessive communication with the host is a source of a possible communication bottleneck. Such communication would normally be only limited to loading the array processor with the necessary data and to unloading the final solution, as mentioned above. In a typical problem a solution is required for a number of transistor bias conditions in a DC analysis or a number of time steps in an transient analysis. Thus, the array processor needs to be loaded only once with the required data, and all that is necessary in subsequent calculations is to update the boundary conditions. This would only involve communication between the host computer and the boundary PEs.

6.2.1 Two-Dimensional Problem

The solution of a system of linear algebraic equations arising from discretization of a partial differential equation in two dimensions is considered first to illustrate the general concepts [Scho75], [Snyd84]. The solution method used in this discussion is the parallel

version of the point-SOR method, outlined in Section 5.4.1.4. It is assumed that the array processor is about to begin a new iteration in the solution process.

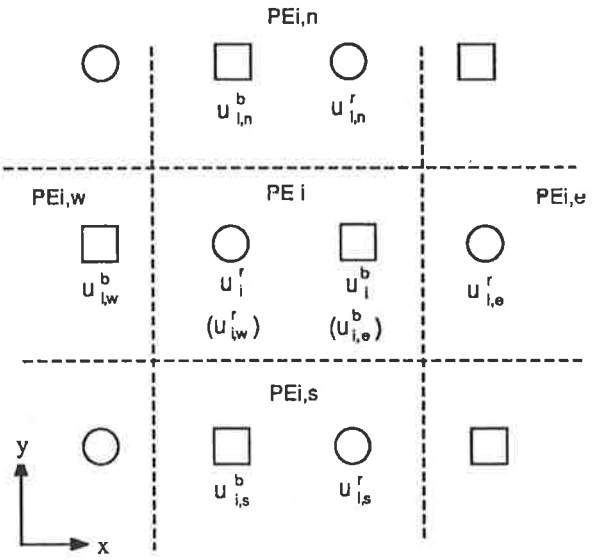


Figure 6.2: Node partitioning for a two-dimensional solution.

The mesh nodes are coloured red-back and each PE solves for the unknowns corresponding to one black and one red node, for example (Figure 6.2). With such a mapping of the unknowns all PEs taking part in the solution process are active on every black and every red iteration step. In general, it is necessary that each PE carry out the solution for an equal number of unknowns belonging to both the red and the black groups of unknowns. It is assumed that there is a sufficient number of PEs within the array processor available for such a grid partitioning to be possible.

With reference to Figure 6.2, PE_i evaluates expressions (5.74)–(5.75), again given below, to find a new approximation $u_i^{r,k+1}$ to its red unknown (it is assumed that the diagonal of the coefficient matrix A has been scaled to unity):

$$\hat{u}_i^{r,k+1} = b_i^r + a_{i,n}u_{i,n}^{b,k} + a_{i,w}u_{i,w}^{b,k} + a_{i,s}u_{i,s}^{b,k} + a_{i,e}u_{i,e}^{b,k}, \tag{6.1}$$

$$u_i^{r,k+1} = (1-\omega)u_i^{r,k} + \omega\hat{u}_i^{r,k+1}, \tag{6.2}$$

and the corresponding expressions to find a new approximation $u_{i,b}^{k+1}$ to its black unknown. Since the calculations involved in updating red unknowns (red stage) are equivalent to the calculations involved in updating black unknowns (black stage), only the red stage calculations will be considered. Each PE_i stores in its internal data memory that part of the coefficient matrix (A) that describes the coupling of the unknowns being solved for within that PE and the unknowns at the neighbouring nodes, and the corresponding part of the right hands side vector ($\vec{b_i}$). The only quantities which are present in expressions (6.1)-(6.2) and which are not stored in the local data memory of PE_i, for

the node partitioning of Figure 6.2, are $u_{i,n}^{b,k}$, $u_{i,w}^{b,k}$ and $u_{i,s}^{b,k}$. These three quantities must be obtained from the neighbouring PEs.

The sequence of arithmetic operations described by expressions (6.1)-(6.2) can be partitioned into two phases performed iteratively, as described by algorithm **SOR 2D** given below:

Algorithm: SOR 2D

1. Phase 1: Each PE_i evaluates $\hat{u}_i^{r,k+1}$ according to expression (6.1). PE_i communicates with its neighbours to obtain the missing quantities needed for this calculation.

2. Phase 2: Each PE_i evaluates $u_i^{r,k+1}$ according to expression (6.2). No inter-processor communication is necessary.

In a similar fashion the black step of the iteration process is performed. This time each PE_i does not need to communicate with its west neighbour to obtain $u_{i,w}^{r,k+1}$ which is the red unknown $u_i^{r,k+1}$ evaluated in the preceding red stage, and stored in its local memory.

The fastest execution of a given task by the PE would be one in which all functional blocks within that PE are utilized at a 100% duty cycle, performing a nonredundant and algorithmically efficient encoding of the task. For an efficient computation, especially during Phase 1, the architecture of the PE should allow for the arithmetic computations to be carried out in parallel with inter-processor communication. More specifically, each PE should be able to communicate with its west neighbour, for example, to receive the value $u_{i,w}^{b,k}$ when computing the product $a_{i,n}u_{i,n}^{b,k}$. This implies that the east neighbour will request the value $u_i^{b,k}$ to be communicated to it at the same time, and thus the PE_i should also be able to communicate with its east neighbour when performing that calculation.

Assuming that the logical architecture of an individual PE allows the degree of parallelism specified above, that any data manipulation operation takes a single instruction cycle and that only two simultaneous data memory accesses are allowed (a three port data memory is excluded at this stage on the basis of design complexity and the large silicon area required for its implementation) the sequence of operations performed by the various functional blocks within the PE according to Phase 1 and Phase 2 of algorithm **SOR 2D** are performed with the highest degree of parallelism as given below ("rd" stands for memory, or an I/O port read and "wt" stands for memory or an I/O

port write):

cycle	Multiplier	Adder	I/O Ports	Data RAM
0		$u_i^{b,k} += \omega \hat{u}_{i,b}^{k}$		rd b_i^r
1	$a_{i,e}u_{i,e}^{b,k}$	$\hat{u}_i^{r,k+1} = b_i^r$	rd $u_{i,n}^{b,k}$ north	rd $a_{i,e}$
			wt $u_i^{b,k}$ south	wt $u_i^{b,k}$
2	$a_{i,n}u_{i,n}^{b,k}$	$\hat{u}_i^{r,k+1} += (a_{i,e}u_{i,e}^{b,k})$	rd $u_{i,w}^{b,k}$ west	rd $a_{i,n}$
			wt $u_i^{b,k}$ east	
3	$a_{i,w}u_{i,w}^{b,k}$	$\hat{u}_i^{r,k+1} += (a_{i,n}u_{i,n}^{b,k})$	rd $u_{i,s}^{b,k}$ south	rd $a_{i,w}$
			wt $u_i^{b,k}$ north	
4	$a_{i,s}u_{i,s}^{b,k}$	$\hat{u}_i^{r,k+1} += (a_{i,w}u_{i,w}^{b,k})$		rd $a_{i,s}$
5	$(1-\omega)u_i^{r,k}$	$\hat{u}_i^{r,k+1} += (a_{i,s}u_{i,s}^{b,k})$		rd $u_i^{r,k}$
6	$\omega \hat{u}_i^{r,k+1}$	$u_i^{r,k+1} += (1-\omega)u_i^{r,k}$		
7		$u_i^{r,k+1} += \omega \hat{u}_i^{r,k+1}$		rd b_i^b

Clock cycle 0 constitutes the last step in the previous black stage in which the new estimate for u_i^b have been calculated. Clock cycles 1-7 constitute a new red stage of the point-SOR iterative method in which a new estimate for u_i^r is calculated. Figure 6.3 depicts the data manipulation performed by PE_i for cycles 1-7. It is assumed that the relaxation factor ω and the quantity $(1-\omega)$ have been precalculated and stored in two separate data register having a direct connection to the Multiplier (see cycles 5 and 6).

From the above discussion it can be concluded that each PE need contain a Multiplier, an Adder, four I/O Ports, a double port Data Memory, presumable implemented as a RAM (random access memory) and a number of registers (reg) that can be accessed by the Multiplier, Adder, Data RAM or any of the I/O Ports. The Adder is utilized at every clock cycle during the calculations described above, and the Multiplier is used for six out of the seven clock cycles needed to update one unknown. In addition, by providing a number of storage registers only two simultaneous Data RAM accesses are ever required.

In this section it has been shown that an evaluation of equations (6.1)-(6.2) needs data stored internally in a given PE_i or data stored only in PEs that are adjacent to the PE_i in the array. This implies that a two-dimensional mesh connection of PEs well corresponds

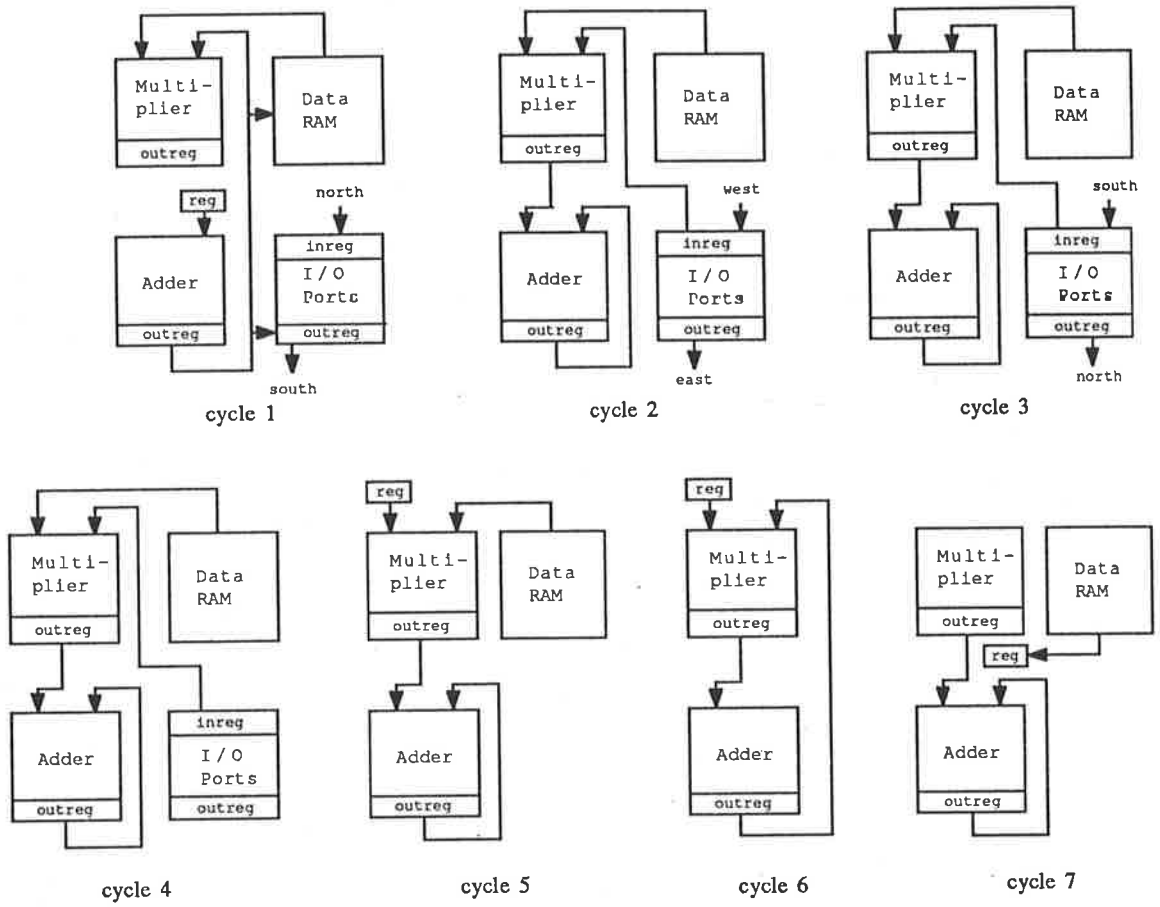


Figure 6.3: Data manipulation within PE_i for Phase 1 and Phase 2 of algorithm **SOR 2D** (red stage).

to the nature of the problem domain. In addition the data handling elements which need to be included in the PE have been identified. The various calculations and data manipulations required of the PE can be carried out with a high degree of parallelism, with most functional blocks within the PE doing useful work on every clock cycle. The following section deals with the calculations required for the evaluation of the stopping test and the relaxation parameter (ω), aspects of the overall solution process often neglected in literature.

6.2.2 Evaluation of a Vector Inner Product

As discussed in Chapter 5, the following operations: evaluation of the stopping test, estimation of a new relaxation parameter ω, estimation of the spectral radius of the iteration matrix needed for the calculation of Chebychev parameters, and the calculations of iteration parameters for the conjugate gradient method, all require calculations of a number of vector inner products. In the following it will be shown that the computations

of vector inner products can potentially become the performance limiting operations in iterative solution algorithms carried out on a two-dimensional array processor.

Consider a general case of evaluating the inner product of two vectors $\vec{a} = (a_1, \ldots, a_m)$ and $\vec{b} = (b_1, \ldots, b_m)$ on a square array of $x \times x$ of PEs, where $n_p = x^2$ is the total number of PEs within the array. Each PE_i, where $i = 1, \ldots, x^2$, stores in its local memory the subvectors $\vec{a}_i = (a_1^i, a_2^i, \ldots, a_z^i)$ and $\vec{b}_i = (b_1^i, b_2^i, \ldots, b_z^i)$, each containing z elements. The vector inner product $\nu = (\vec{a}, \vec{b}) = \sum_{k=1}^{k=m} a_k b_k$ is computed in two phases, as described by algorithm **VIP2D** set out below:

Algorithm: VIP2D

1. Phase 1: In each PE_i compute the partial inner vector product $\nu_i' = (\vec{a_i}, \vec{b_i}) = \sum_{k=1}^{k=z} a_k^i b_k^i$.
2. Phase 2: Compute the SUM operator over the elements ν_i', ie $\nu = \text{SUM} = \sum_{i=1}^{i=n_p} \nu_i'$.

Phase 1 of the above process is carried out simultaneously within each PE, with no inter-processor communications being necessary. Assuming that the operations of addition and multiplication each take one clock cycle, the total time cost of the operations in Phase 1 is approximately z clock cycles, for two z element subvectors. The main problem, however, is to efficiently carry out Phase 2 of the algorithm, since this involves performing summations across the processor array.

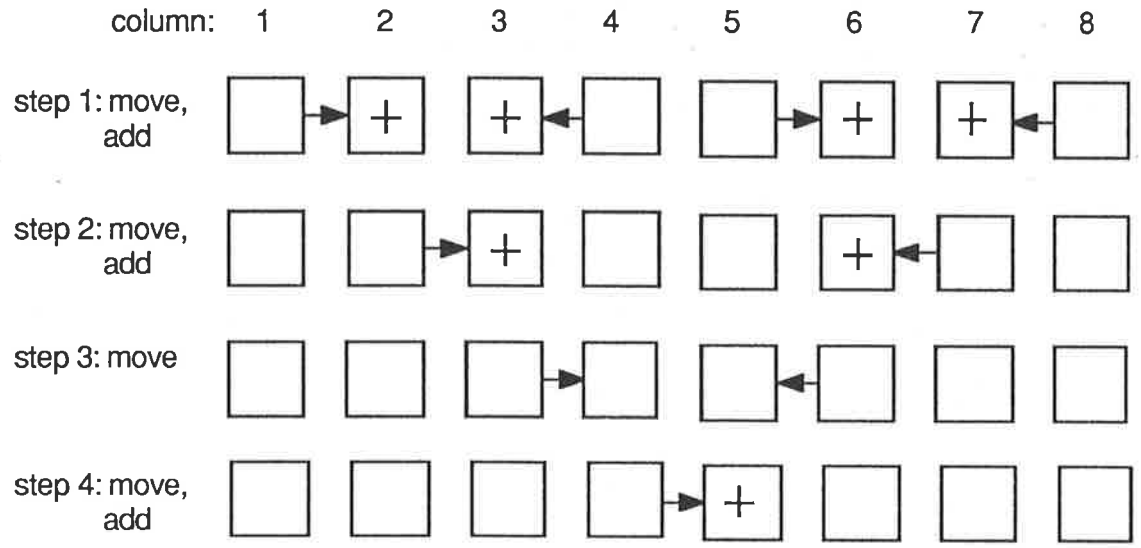


Figure 6.4: Efficient evaluation of a vector inner product.

According to [Park87], the cost of the SUM operator on a square array of x^2 PEs, each PE storing in its local memory one number, is $\log_2(n_p)$ additions and $2(x-1)$

move operations, where a move operation is defined as being a datum transfer between two neighbouring PEs. This result shows that it is important to consider not only the computation complexity of an algorithm but also its communications complexity. In this case the communication complexity is larger than the computation complexity. The result given in [Park87] may be improved upon. In Figure 6.4 it is shown how the SUM operator can be evaluated on a two-dimensional processor array at a cost of $\log_2(n_p)$ additions and x move operations. In the figure the sequence of operations carried out in a single row of the array is shown, where $x = 8$. The cost associated with this process is $0.5\log_2(n_p) = 3$ additions and $0.5x = 4$ move operations. At the end of this sequence PEs in column 5 of each row contain in their local memories the sums of ν_i' across the respective rows of the array. This sequence is then repeated across the PEs in column 5, with the final result of the SUM operator becoming available in PE positioned in row 5 and column 5 of the array.

Using the scheme described above the SUM operator is evaluated on an 8×8 array of PEs at a cost of $\log_2 64 = 6$ additions and 8 move operations. It can be easily shown that in general this cost is $\log_2 n_p$ additions and x move operations. The communication complexity has been approximately halved over the result given in [Park87], however, the communication complexity is still larger than the computation complexity and it increases linearly with x.

Assuming that the cost of a move operation is one clock cycle, the cost of the operations of Phase 2 of algorithm **VIP2D** is approximately $2\log_2(x)+x$ clock cycles. The required vector inner product (ν) may be made available in each PE at an additional cost of x clock cycles, ie this step involves inter-processor move operations only. The total time taken to evaluate a vector inner product, with both vectors containing $m = x^2z$ elements and with the vectors partitioned between the PEs as defined above, and to make the result available in all PEs is approximately $2[\log_2(x) + x] + z$ clock cycles. In addition, during such a calculation the PEs cannot perform any other operations associated with the solution process.

For example, for an array processor composed of 4096 PEs ($x = 64$) the cost of calculating a vector inner product and making the result available in each PE is 142 clock cycles (for $z = 2$, ie one red and one black unknown). The cost of evaluating the stopping test is 78 clock cycles, since the result does not need to be made available in all PEs, but instead it is communicated to the host computer. The above results show that, for a modest size array processor, the estimation of new relaxation parameters for the point-SOR method is equivalent to approximately 10 iterations of the point-SOR method, carried out as discussed above (it is assumed that the vector inner product

calculations are pipelined and therefore the cost of computing two vector inner products is essentially the same, for the purposes of this discussion, as the cost of one vector inner product computation). Performing this operation every 10 iterations, for example, approximately doubles the length of time needed to solve a system of linear algebraic equations using this method. From the above it may be concluded that the vector inner product calculation also become the performance limiting factor in methods such as the conjugate gradient method, where two vector inner products (apart from eigenvalue estimation needed for the stopping test) must be evaluated at *every* iteration.

It has become evident from the above discussion that a time efficient calculation of the vector inner product is very important as it determines, to a large degree, the speed with which linear systems of equations can be solved on a two-dimensional array processor using iterative solution methods. In the next section a modification of the two-dimensional array processor architecture (one of the contributions of this thesis) is proposed. The modified architecture is called here the TIME Machine and it can perform the calculation just described in $O(\log_2 x^2)$ time. This is the fastest possible way of evaluating the sum of x^2 numbers [Kuck81], ie the numbers are added together in a binary tree sequence.

6.3 The TIME Machine Architecture

Consider a 4×4 array of PEs. A sum of sixteen numbers (ν'_i, $i = 1, \ldots, 16$) stored one per PE can be evaluated by this group of PEs in $\log_2 16 = 4$ phases, each phase taking two clock cycles. A slightly different sequence of inter-processor communications to that suggested by the algorithm **VIP 2D** is shown by a set of diagrams given in Figure 6.5.

During the first clock cycle of Phase 1 (part (a) of Figure 6.5) the PEs in columns 1 and 4 communicate their values to PEs in columns 2 and 3, respectively. During the second clock cycle of Phase 1 the PEs in columns 2 and 3 evaluate the sums of the values initially present in their local data memories and the values communicated to them in clock cycle 1, eg the PE in column 2 and row 3 evaluates the sum $\nu'_{10} + \nu'_9$.

At the beginning of Phase 2 (part (b) of the figure) the PEs in row 1 and columns 2 and 3 communicate the results of the calculations of Phase 1 to their respective neighbouring PEs in row 2 and columns 2 and 3. Simultaneously, the PEs in row 4 and columns 2 and 3 communicate the results of the calculations of Phase 1 to their respective neighbouring PEs in row 3 and columns 2 and 3. During the fourth clock cycle the four PEs in grid positions (2,2), (2,3) (3,2) and (3,3) evaluate the sums of the results of the calculation

carried out in Phase 1 and the values communicated to them at the beginning of Phase 2, eg the PE in row 3 and column 2 evaluates the sum $(\nu'_{10} + \nu'_9) + (\nu'_{13} + \nu'_{14})$.

Following the last phase of this process (see parts (c) and (d) of Figure 6.5) the PE in row 3 and column 3 contains the required sum, ie $\sum_{i=1}^{i=16} \nu'_i$.

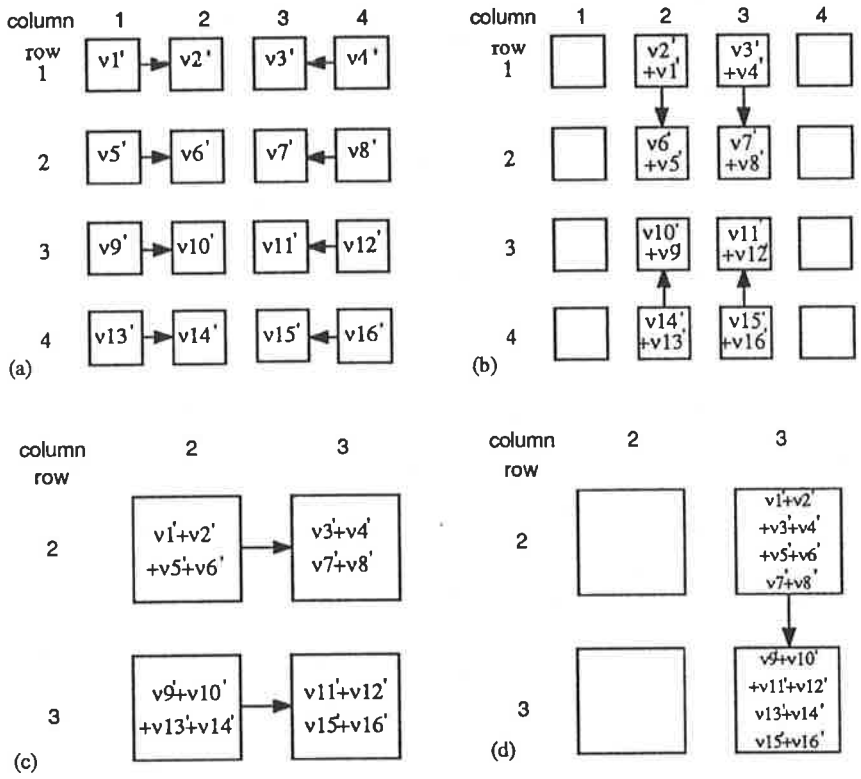


Figure 6.5: Summation over a 4×4 group of PEs: (a) Phase 1, (b) Phase 2, (c) Phase 3, (d) Phase 4.

If a much larger array processor (eg 64×64) of PEs is subdivided into groups of 4×4 PEs, then the calculation described above can be simultaneously performed by each group within the array at a cost of 4 additions and 4 move operations, ie 8 clock cycles. The results of such calculations consist of $64^2/16 = 256$ partial results which need to be added together to form the final sum, which constitutes the required vector inner product. The objective is to sum these numbers as quickly as possible.

Note that the calculations described above for a group of 4×4 PEs were carried out as if though the individual PEs were connected as a binary tree, with some PEs performing the functions of both parent nodes and children nodes. This effective binary tree of PEs is completed by introducing additional PEs, as shown in Figure 6.6 for the case of a two-dimensional array of 8×8 PEs. The additional PEs and the additional communication channels make it possible to evaluate the sum of the partial results, computed by the

individual groups of 4×4 PEs, at a cost of $\log_2(x^2/16)$ additions and $\log_2(x^2/16)$ move operations. In the case of the 64×64 array of PEs, considered above, the 256 partial results can be summed by this modified array processor in 16 clock cycles.

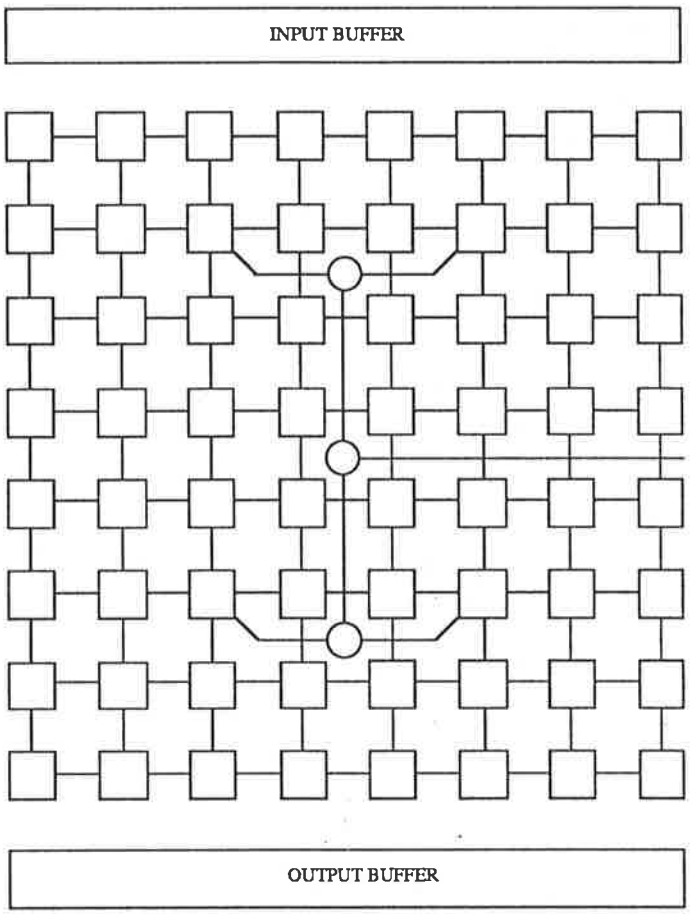


Figure 6.6: TIME Machine architecture: the circles denote the additional PEs connected as a binary tree (squares and circles denote identical PEs).

The time to evaluate the sum of 64^2 numbers (one number per PE) is then $2\log_2 64^2 = 24$ clock cycles. It takes an additional $\log_2 64^2 = 12$ clock cycles to make the final result available in each PE. Thus, the process of vector inner product calculation takes $3\log_2(x^2) + z = 38$ $(x = 64,\ z = 2)$ clock cycles on the modified array processor compared to $2[\log_2(x) + x] + z = 142$ clock cycles on the conventional two-dimensional array processor. In addition, once the computations are transferred to the PEs connected as the binary tree, the PEs connected as the two-dimensional mesh can proceed to perform all the tasks of a new iteration step which do not require the result of the vector inner product calculation. Therefore, the effective cost of evaluating a vector inner product and making the result available in all PEs is only the cost of computations and interprocessor communications within a 4×4 group of PEs, ie $3\log_2 16 = 12$ clock cycles,

which is equivalent to 0.86 of one iteration of the point-SOR method.

This modified array processor is termed here the TIME Machine. The word TIME has been derived from "a binary Tree In a two-dimensional MEsh of processing elements". The additional number of PEs, required to modify an original two-dimensional array processor composed of x^2 PEs is given by:

$$\sum_{i=5}^{\log x^2} x^2 2^{-i} = x^2(1/16 - 1/x^2). \tag{6.3}$$

This corresponds in the limit to 6.25% of the number of PEs in the original two-dimensional array processor.

In general, trees serve to distribute and collect data [Snyd84], [Uhr84], while arrays execute highly parallel, relatively local processes [BBKK68], [Uhr84], [Zakh84]. The TIME Machine architecture combines the main advantages of these two most widely used structures, making it suitable for performing tasks which are efficiently performed both on the binary tree and two-dimensional array processor architectures.

The main disadvantage of this new architecture is that the distance between the PEs connected as the binary tree increases with x. This is not, however, a problem of large proportions since only a small fraction of the PEs is affected. The solution for very large arrays in which the communication time may dominate could be either to increase the time for communication between the affected PEs, or to provide proportionally larger off-chip drivers. The other costs associated with the TIME Machine is that some of the PEs need to be connected to five, rather than only four, other PEs.

The above discussion focused on those parts of a possible solution process that require global communications between individual PEs, such as a calculation of the vector inner product required by any iterative solution method. In Section 6.2.1 an outline of the calculation process carried out by an individual PE, as part of the overall point-SOR method, was presented. As a result, an initial PE architecture specification was identified. In the following section a similar discussion for the solution of a three-dimensional problem on the TIME Machine is presented, with the objective of identifying the required PE architecture in more detail.

6.3.1 Three-Dimensional Problem

The linear system of equations of interest is given by expression (5.57), as before. The nodes in a three-dimensional mesh are ordered by lines in the red-black manner, as shown in Figure 5.7. A possible node partitioning for such a node ordering is depicted

in Figure 6.7, with each PE being required to solve for the unknowns corresponding to one red and one black line of nodes, and each line containing z nodes (in general each PE solves for the unknowns on an equal number of red and black lines of nodes). With this partitioning the solution process can be effectively carried out using a line iterative method which gives a faster rate of convergence over the corresponding point iterative method, as discussed in Section 5.4.1.5. (Only the computations performed by a given PE_i for the red stage of a single iteration will be considered.) The line-Jacobi iterative method applied to such a system is given by:

$$\vec{v}_i^{r,k} = \vec{b}_i^r + A_{i,n}\vec{u}_{i,n}^{b,k} + A_{i,w}\vec{u}_{i,w}^{b,k} + A_{i,s}\vec{u}_{i,s}^{b,k} + A_{i,e}\vec{u}_{i,e}^{b,k}, \tag{6.4}$$

$$A_{i,i}\hat{\vec{u}}_i^{r,k+1} = \vec{v}_i^{r,k}. \tag{6.5}$$

This requires at each iteration of the linear equation solution process, apart from other calculations to be discussed shortly, a solution of a tridiagonal system of linear algebraic equations (one per PE) given by (6.5). Normally the convergence of a line iterative method, such as that given by the above expressions, is improved by applying either the Chebychev or the conjugate gradient acceleration, as discussed in Section 5.4.3.

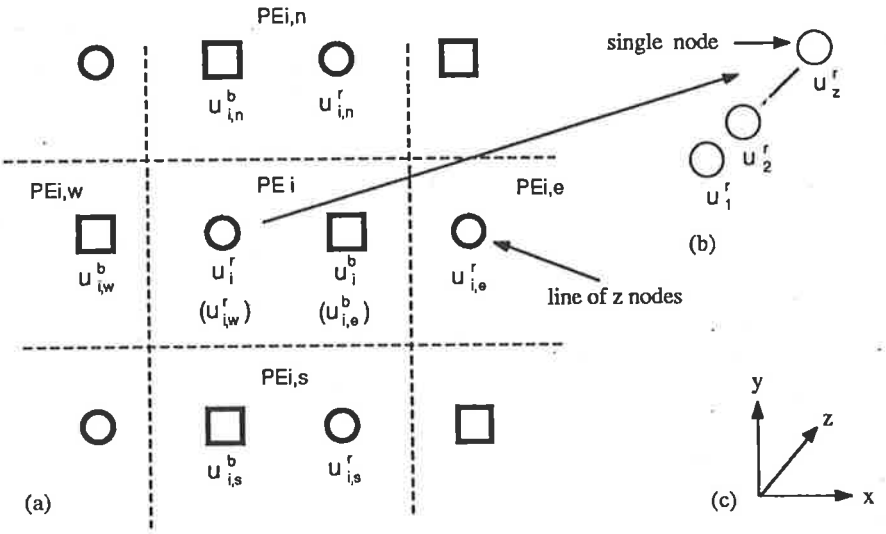


Figure 6.7: (a) Line partitioning for a three-dimensional space discretization, (b) adopted nomenclature, (c) coordinate system.

One iteration step of the complete accelerated iterative method, necessary to update the unknowns corresponding to a red line of nodes, can then be subdivided into three phases shown below:

Algorithm: AC-3D

1. Phase 1: assemble the new right hand side vector $\vec{v}_i^{r,k}$ according to expression (6.4);
2. Phase 2: solve the tridiagonal system of equations given by expression (6.5) for $\hat{\vec{u}}_i^{r,k+1}$;
3. Phase 3: apply either the Chebychev or the CG acceleration to obtain the new estimate $\vec{u}_i^{r,k+1}$.

In the above it is assumed that the unknown vector $\vec{u}$, the coefficient matrix A and the known right hand side vector $\vec{b}$ have been partitioned and distributed between the individual PEs so that each PE can carry out its part of the overall solution process, ie PE_i stores in its local memory $\vec{u}_i^r$, $\vec{u}_i^b$, $\vec{b}_i^r$, $\vec{b}_i^b$, $A_{i,i}$, $A_{i,n}$, $A_{i,w}$, $A_{i,s}$ and $A_{i,e}$.

In an initialization phase of the solution process the LU decomposition of the tridiagonal coefficient matrix $A_{i,i}$ is carried out. The decomposition is necessary for an efficient solution of the tridiagonal system of linear equations carried out in Phase 2 of the above algorithm. Two matrices L_i and U_i are found, such that $A_{i,i} = L_i U_i$, where L_i is a lower bidiagonal matrix and U_i is an upper bidiagonal matrix. The LU decomposition process is not discussed here because the linear first order recurrence relations which must be solved to obtain the decomposition are very similar to the linear recurrence relations involved in the forward and backward substitutions of the actual solution process, which will be discussed in detail in the later part of this section. The result of the decomposition process is shown in Figure 6.8. The elements of matrix L_i and matrix U_i are calculated according to the following relations:

$$e_1^i = 1/a_{1,1}^i, \tag{6.6}$$

$$e_j^i = 1/(a_{j,j}^i - a_{j-1,j}^i w_{j-1}), \tag{6.7}$$

$$w_j^i = a_{j-1,j}^i e_j^i, \quad \text{for } j = 2, \ldots, z. \tag{6.8}$$

Each of the solution stages of the iterative process will now be discussed in more detail. Since the discussion of this section refers only to a single equation evaluated within a single PE the subscript i will be omitted for clarity from now on.

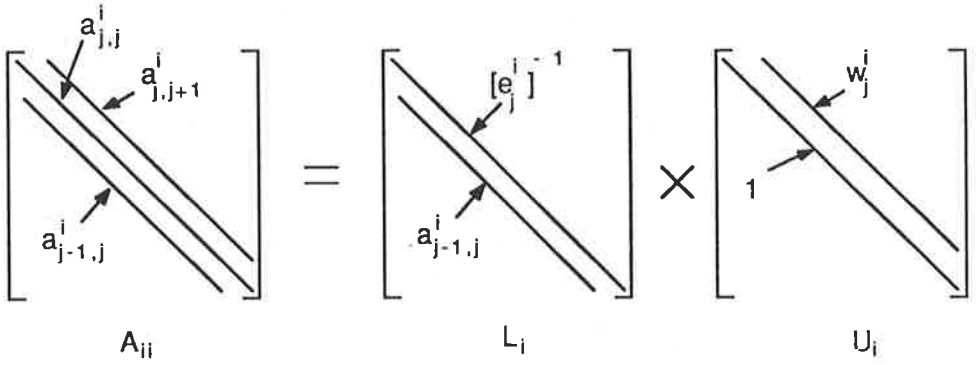


Figure 6.8: *LU* decomposition of a tridiagonal coefficient matrix $A_{i,i}$.

6.3.1.1 Phase 1 of algorithm AC-3D

The required calculations in the evaluation of the right hand side vector ($\vec{v}_i^{r,k}$) given by (6.4) are as follows:

$$v_j^{r,k} = b_j^r + a_{j,n}u_{j,n}^{b,k} + a_{j,w}u_{j,w}^{b,k} + a_{j,s}u_{j,s}^{b,k} + a_{j,e}u_{j,e}^{b,k}, \quad j = 1, \ldots z. \tag{6.9}$$

This is equivalent to performing z times the calculations described by (6.1). A possible sequence of operations for each PE, coded for maximum parallelism (the same assumptions about the PE logical architecture are made as those made in in Section 6.2.1), is as follows:

cycle	Multiplier	Adder	I/O Ports	Data RAM
1				rd b_j^r
				rd $u_j^{r,k}$
2	$a_{j,e}u_{j,e}^{b,k}$	$v_j^{r,k} = b_j^r$	rd $u_{j,n}^{b,k}$ north	rd $a_{j,e}$
			wt $u_j^{r,k}$ south	
3	$a_{j,n}u_{j,n}^{b,k}$	$v_j^{r,k} += a_{j,e}u_{j,e}^{b,k}$	rd $u_{j,w}^{b,k}$ west	rd $a_{j,n}$
			wt $u_j^{r,k}$ east	
4	$a_{j,w}u_{j,w}^{b,k}$	$v_j^{r,k} += a_{j,n}u_{j,n}^{b,k}$	rd $u_{j,s}^{b,k}$ south	rd $a_{j,w}$
			wt $u_j^{r,k}$ north	
5	$a_{j,s}u_{j,s}^{b,k}$	$v_j^{r,k} += a_{j,w}u_{j,w}^{b,k}$		rd $a_{j,s}$
				rd b_{j+1}^r
6	$a_{j+1,e}u_{j+1,e}^{b,k}$	$v_j^{r,k} += a_{j,s}u_{j,s}^{b,k}$	rd $u_{n,j+1}^{b,k}$ north	rd $a_{j+1,e}$
			wt $u_{j+1}^{r,k}$ south	
7	$a_{j+1,n}u_{j+1,n}^{b,k}$	$v_{j+1}^{r,k} =$	rd $u_{w,j+1}^{b,k}$ west	rd $a_{j+1,n}$
		$b_{j+1}^r - a_{j+1,e}u_{j+1,e}^{b,k}$	wt $u_{j+1}^{r,k}$ east	wt $v_j^{r,k}$

The above process is very regular. Cycles $1, 2, 3$ comprise an initialization stage, after which a program loop is set up involving steps $4, 5, 6, 7$. Subvectors $\vec{v}_i^{r,k}$ are formed in all PEs in approximately $4z$ clock cycles. A typical cycle of the above process includes: a Data RAM to Multiplier datum transfer, an I/O Port register to Multiplier datum transfer, a datum transfer between the output register of the Adder and one of its inputs, an I/O Port read and write, a multiplication and an addition. This gives an approximate measure of communication complexity within a single PE. A maximum of two simultaneous accesses to the data memory and two simultaneous input-output port accesses are required in any one clock cycle. In order for the above computations to require two, rather than three simultaneous memory accesses at any one cycle, it is necessary to include input and output registers within each of the four I/O Ports. In this way the required datum ($u_j^{r,k}$ in this case) to be broadcast to the neighbouring PEs is written into the output registers of each I/O Port during cycle 1 and no additional data memory reads for this particular datum are necessary following cycle 1. In addition, an external storage register (denoted reg in Figure 6.9) having direct connection to the Adder is provided. This register is used in cycle 5 to store the value b_j^r to be used by the Adder during cycle 7. Without this external register three data memory accesses would have been required during cycle 7: two reads to get b_j^r and $a_{j+1,n}$ and one write to store $v_j^{r,k}$.

A diagram showing the required data transfers between the various modules for cycles 4 to 7 is given in Figure 6.9.

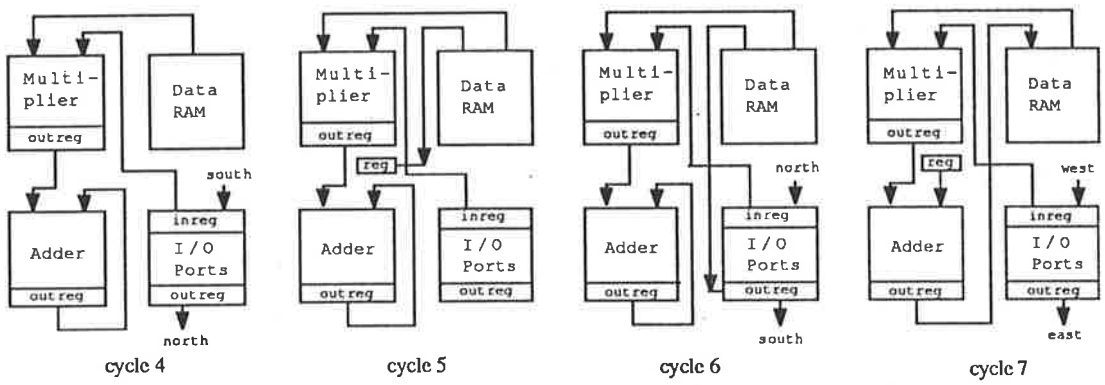


Figure 6.9: Data transfers for cycles 4 to 7 of Phase 1 of algorithm **AC-3D**.

6.3.1.2 Phase 2 of algorithm AC-3D

The solution of (6.5) based on the LU decomposition of the coefficient matrix $A_{i,i}$ is carried out in two stages. First, the equation $L_i\vec{g}_i = \vec{v}_i^{r,k}$ is solved for the vector

$\vec{g}_i$ (forward substitution), followed by the calculation of the required vector $\hat{\vec{u}}_i^{r,k}$ from $U_i\hat{\vec{u}}_i^{r,k} = \vec{g}_i$ (backward substitution).

1. The forward substitution is carried out within each PE as follows:

$$\begin{aligned} g_1 &= v_1^{r,k}e_1, \\ g_j &= (v_j^{r,k} - a_{j,j-1}g_{j-1})e_j \quad j = 2,\ldots,z. \end{aligned} \tag{6.10}$$

The sequence of operations coded for maximum parallelism is given below and the corresponding data transfers between the various data handling elements are shown in Figure 6.10.

cycle	Multiplier	Adder	Data RAM
1	$g_1 = v_1^{r,k}e_1$		rd e_1
			rd $v^{r,k}$
2	$a_{j,j-1}g_{j-1}$		rd $a_{j,j-1}$
			wt g_1
3		$v_j^{r,k} - a_{j,j-1}g_{j-1}$	rd $v_j^{r,k}$
4	$g_j = (v_i^{r,k} - a_{j,j-1}g_{j-1})e_i$		rd e_j

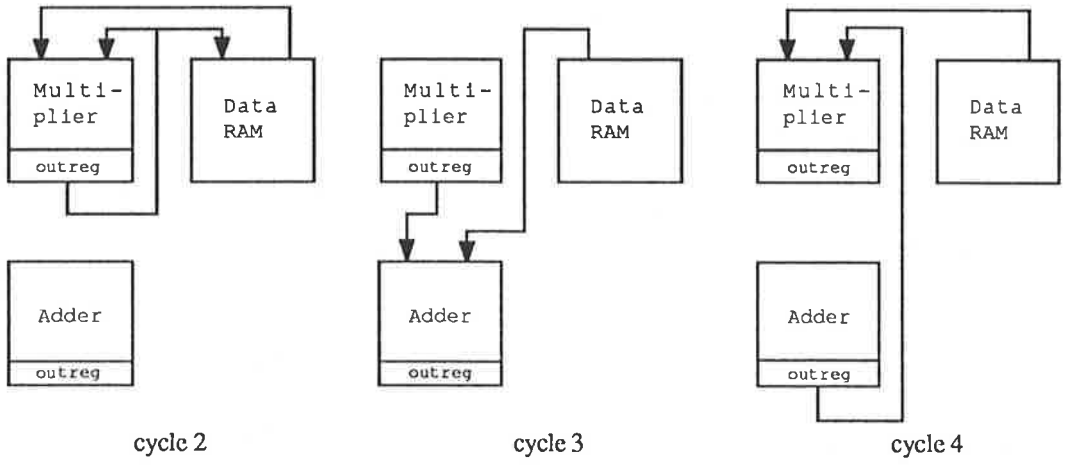


Figure 6.10: Forward Substitution: required data transfers within each PE for cycles 2 to 4.

This process is very regular, with cycles 2 to 4 being executed repetitively for $j = 2,\ldots,z$. No inter-processor communication is necessary and the forward substitution process proceeds simultaneously in each PE, at the cost of $3z$ clock cycles.

2. The back substitution is carried out within each PE as follows:

$$\begin{aligned} \hat{u}_z^{r,k} &= g_z, \\ \hat{u}_j^{r,k} &= g_j - w_j \hat{u}_{j+1}^{r,k}, \quad j = z-1, \ldots, 1. \end{aligned} \tag{6.11}$$

The sequence of operations coded for maximum parallelism is given below and the corresponding data transfers are shown in Figure 6.11.

cycle	Multiplier	Adder	Data RAM
1	$w_j \hat{u}_{j+1}^{r,k}$		rd w_j
			rd $\hat{u}_{j+1}^{r,k}$
2		$\hat{u}_j^{r,k} = g_j - w_j \hat{u}_{j+1}^{r,k}$	rd g_j
3	$w_{j-1} \hat{u}_j^{r,k}$		rd w_{j-1}
			wt $\hat{u}_j^k$
4		$\hat{u}_{j-1}^{r,k} = g_{j-1} - w_{j-1} u_j^k$	rd g_{j-1}

In this case the program loop consists of cycles 2 and 3. No inter-processor communication is necessary and the back substitution step proceeds simultaneously in each PE, at the cost of $2z$ clock cycles.

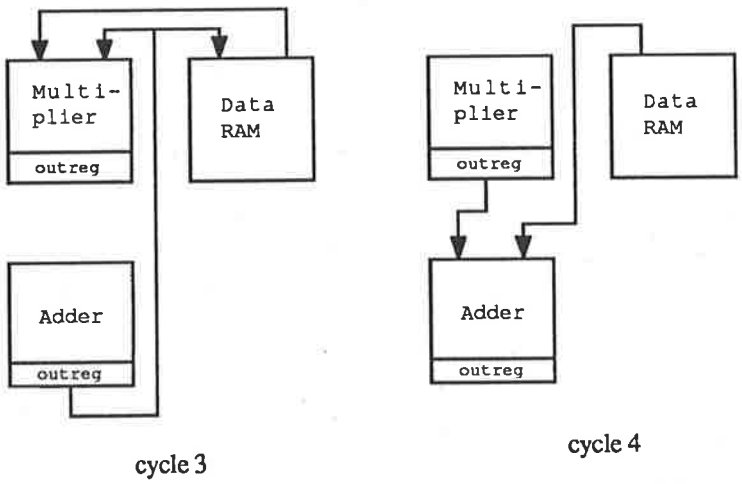


Figure 6.11: Backward substitution: required data transfers within each PE for cycles 3 and 4.

The last phase of the iteration step (Phase 3) involves a calculation of the new estimate $\vec{u}_i^{r,k+1}$ from $\hat{\vec{u}}_i^{r,k+1}$, using either the Chebychev acceleration or the conjugate gradient acceleration method. Each of these will now be discussed in turn.

6.3.1.3 Chebychev Acceleration

The relation between $\vec{u}_i^{r,k+1}$ and $\hat{\vec{u}}_i^{r,k+1}$ according to this method, and with a red-black line partitioning is given by [Hage81], [Colg82], [Axel85]:

$$\vec{u}_i^{r,k+1} = \vec{u}_i^{r,k} + \rho^{r,k+1}(\hat{\vec{u}}_i^{r,k+1} - \vec{u}_i^{r,k}), \tag{6.12}$$

where $\rho^{r,k+1}$ is the acceleration parameter used in a red stage, and for $k > 1$ it is given by:

$$\rho^{r,k+1} = \left(1 - \frac{1}{4}MG^2\rho^{b,k}\right)^{-1}, \tag{6.13}$$

where $\rho^{b,k}$ is the corresponding black stage acceleration parameter, with $\rho^{r,1} = 1$ and $\rho^{b,1} = 2/(2 - MG^2)$.

The sequence of operations for PE_i performing the operations given by (6.12) on its subvector is given as follows (for $j = 1, \ldots, z$):

cycle	Multiplier	Adder	Data RAM
1			rd ρ^{k+1}
2		$\hat{u}_j^{r,k+1} - u_j^{r,k}$	rd $\hat{u}_j^{r,k+1}$
			rd $u_j^{r,k}$
3	$\rho^{r,k+1}(\hat{u}_j^{r,k+1} - u_j^{r,k})$	$\hat{u}_{j+1}^{r,k+1} - u_{j+1}^{r,k}$	rd $\hat{u}_{j+1}^{r,k+1}$
			rd $u_{j+1}^{r,k}$
4	$\rho^{r,k+1}(\hat{u}_{j+1}^{r,k+1} - u_{j+1}^{r,k})$	$u_j^{r,k} + \rho^{r,k+1}(\hat{u}_j^{r,k+1} - u_i^{r,k})$	rd $\hat{u}_{j+2}^{r,k+1}$
			rd $u_{j+2}^{r,k}$
5		$u_{j+1}^{r,k} + \rho^{r,k+1}(\hat{u}_{j+1}^{r,k+1} - u_{j+1}^{r,k})$	wt $u_{j+1}^{r,k}$
			rd $u_{j+1}^{r,k}$
6		$\hat{u}_{j+2}^{r,k+1} - u_{j+2}^{r,k}$	wt $u_{j+1}^{r,k+1}$

Each processor updates its subvector of unknowns at a cost of $2z$ clock cycles. A program loop can again be established and it involves cycles 3 to 6. The data transfer and arithmetic operations performed in cycles 2 to 5 are depicted in Figure 6.12

In a general case, as pointed out previously, the largest eigenvalue MG of the corresponding iteration matrix has to be estimated adaptively. Normally the process described by

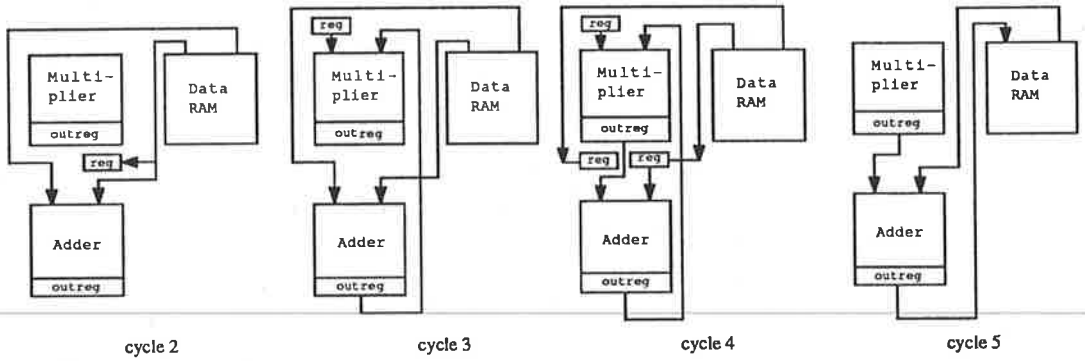


Figure 6.12: Chebychev Acceleration: required data transfers for cycles 2 to 5.

(6.12) and (6.13) is carried out for a number of iterations with the best available estimate for MG, before a new estimate for MG is calculated, which calculation involves a vector inner product computation. The required vector inner product is calculated in two stages, as described in Section 6.3. The important point here is that as soon as the summing process is transferred to the PEs forming the binary tree the mesh connected PEs can commence a new iteration step, since a number of calculations can be performed before the new estimate for MG is required. This is because following an evaluation of a new estimate for MG a new sequence of iterations is also commenced, ie k is set to 1, and $\rho^{r,1} = 1$. Consequently, the complete first red stage of a new sequence of iterations can be evaluated before the next estimate for MG becomes necessary. In addition the parameter $\rho^{b,1}$, which requires the next estimate for MG, is not used until Phase 3 of the **AC-3D** algorithm. Therefore, in addition to performing the first red stage, the calculations involved in Phase 1 and Phase 2 of the **AC-3D** algorithm for the first black stage of a new sequence of iterations can also be carried out. By overlapping the vector inner product calculations with the remainder of the calculations the effective costs of inner vector product calculations are greatly reduced on the TIME Machine for the Chebychev acceleration method.

6.3.1.4 Conjugate Gradient (CG) Acceleration

An efficient evaluation of vector inner products is more critical when the CG acceleration method is applied, firstly because two such operations are necessary in every iteration step to evaluate the acceleration parameters α^k and β^k, and secondly because it is not possible to perform any other work in parallel with the calculation of the vector inner products as the results of these operations must be available before the next task of the CG algorithm can be performed. This becomes apparent when the calculation involved

in one step of the conjugate gradient method is examined:

$$\vec{s}^k = A\vec{p}^k, \tag{6.14}$$

$$\alpha^k = \frac{(\hat{\vec{r}}^k, \vec{r}^k)}{(\vec{p}^k, \vec{s}^k)}, \tag{6.15}$$

$$\vec{u}^{k+1} = \vec{u}^k + \alpha^k \vec{p}^k, \tag{6.16}$$

$$\vec{r}^{k+1} = \vec{r}^k - \alpha^k \vec{s}^k, \tag{6.17}$$

$$M\hat{\vec{r}}^{k+1} = \vec{r}^{k+1}, \tag{6.18}$$

$$\beta^k = \frac{(\hat{\vec{r}}^{k+1}, \vec{r}^{k+1})}{(\hat{\vec{r}}^k, \vec{r}^k)}, \tag{6.19}$$

$$\vec{p}^{k+1} = \vec{r}^{k+1} + \beta^k \vec{p}^k. \tag{6.20}$$

Expression (6.14) is a matrix-vector multiplication. This operation involves communication between PE_i and its neighbouring PEs to obtain the values $p^k_{j,n}$, $p^k_{j,w}$, $p^k_{j,s}$, $p^k_{j,e}$. All PE_i simultaneously evaluate the following expression, for $j = 1, \ldots, z$:

$$s^k_j = a_{j-1,j}p^k_{j-1} + a_{j,j}p^k_j + a_{j,j+1}p^k_{j+1} + a_{j,n}p^k_{j,n} + a_{j,w}p^k_{j,w} + a_{j,s}p^k_{j,s} + a_{j,e}p^k_{j,e}. \tag{6.21}$$

This operation is equivalent to Phase 1 of the iteration process given by algorithm **AC-3D** and for a subvector $\vec{p}^k_j$ with z elements it takes approximately $7z$ clock cycles.

Expression (6.15) involves a calculation of the vector inner product $(\vec{p}^k, \vec{s}^k)$. This task is performed as described in Section 6.3. Briefly, each PE calculates its contribution (ν_i) of the total sum at a cost of z clock cycles:

$$\nu_i = p^k_1 s^k_1 + p^k_2 s^k_2 + p^k_3 s^k_3 + \cdots + p^k_z s^k_z. \tag{6.22}$$

These components are then summed over all PEs at a cost of $2\log_2 n_p$ clock cycles. The acceleration parameter α^k is calculated by the PE at the root of the binary tree of PEs, and then distributed to all mesh connected PEs at a cost of $\log_2 n_p$ clock cycles.

During this process no other work can be carried out, since the result of this calculation, ie α^k, is required in the next step of the complete sequence. A similar comment applies to the calculation of β^k, which involves the calculation of the vector inner product of vectors $\hat{\vec{r}}^{k+1}$ and $\vec{r}^{k+1}$.

Expressions (6.16), (6.17) and (6.20) are evaluated by each PE independently at a cost of $3z/2$ clock cycles per expression if the previous assumptions about the PE logical architecture are observed. The operation sequence carried out in the evaluation of expression (6.16) is given as follows:

cycle	Multiplier	Adder	Data RAM
1	$\alpha^k p_j^k$		rd p_j^k
			rd α^k
2	$\alpha^k p_{j+1}^k$	$u_j^{k+1} = u_j^k + \alpha^k p_j^k$	rd p_{j+1}^k
			rd u_j^k
3		$u_{j+1}^{k+1} = u_{j+1}^k + \alpha^k p_{j+1}^k$	rd u_{j+1}^k
			wt u_j^{k+1}
4	$\alpha^k p_{j+2}^k$		rd p_{j+2}^k
			wt u_{j+1}^{k+1}
5	$\alpha^k p_{j+3}^k$	$u_{j+2}^{k+1} = u_{j+2}^k + \alpha^k p_{j+2}^k$	rd p_{j+3}^k
			rd u_{j+2}^{k+1}

No inter-processor communications are necessary to carry out these computations. The calculations shown for cycles 2, 3 and 4 are repeated until all terms have been calculated, ie a short program loop is established. Figure 6.13 shows the data manipulation in each PE for cycles 2, 3 and 4.

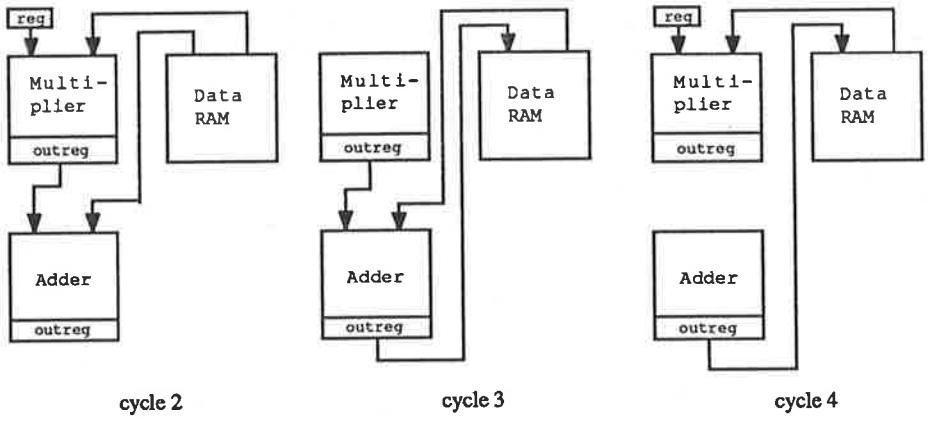


Figure 6.13: Step 3 of the CG method, cycles 2, 3 and 4.

The calculations described by expression (6.18) are those needed to carry out Phase 2 of the overall iteration process, with the block matrix $A_{i,i} = M_{i,i}$, the subvector $\vec{u}_j^{r,k} = \vec{u}_j^k$ and the subvector $\vec{b}_j^r = \vec{r}_j^k$.

6.4 The Mathematical Processor (mP)

In the previous sections the main computational tasks that need to be performed by the individual PE of the TIME Machine have been identified. In addition, some aspects of the PE logical architecture have been specified. In this section the main aspects of the logical architecture of the PE (named here the Mathematical Processor) and its transformation to the physical architecture are described.

6.4.1 Design Overview

The Mathematical Processor (Figure 6.14) is a single chip, programmable processor whose internal architecture resembles that of typical digital signal processing (DSP) chips [HKMT83], [HKBB85], [ADSP86], [Moto86], [AlFe87].

Keeping the processor cell on a single chip allows the functional blocks of the mP to operate together without paying the penalty of off-chip communication. Furthermore, since each chip contains only one mP, implementation of the TIME Machine architecture and other mP interconnection schemes, such as the "Omega" network [Lawr75] or the "Perfect Shuffle" network [Ston71], can easily be achieved by custom inter-chip wiring on PC (printed circuit) boards.

The applications for which the mP is intended involve execution of regular sequences of operations which can be efficiently coded with short program loops, such as those identified in Section 6.3.1. As a result, the on-chip program memory requirements are likely to be quite small, eg 100 to 200 words. Making the mP a programmable device, however, means that all mPs (with the exception of the boundary mPs and the binary tree connected mPs) within the TIME Machine will contain the identical program code. In this respect some of the architectural density has been traded for an additional flexibility in the programming of different applications (perhaps involving different mPs to execute different sequences of operations).

The logical architectures of many DSP chips are optimized for the sum of products computations [HKMT83], [HKBB85], [AlFe87] needed for such signal processing tasks as convolution or correlation [RaGo75], and with some exceptions, eg the TFB chip (described in Appendix A) whose architecture also supports array environment operation, for stand alone operation. The mP on the other hand is mainly required to work as an element in the highly parallel multiprocessor system having the TIME topology, but it also has the capability for stand alone operation for less demanding applications.

The operations of addition (subtraction is equivalent to addition) and multiplication are the main functions of this chip, since these operations occur with the greatest frequency in the various algorithms discussed in Chapter 5. Other mathematical operations, such as division and square root, and functions, such as logarithmic and exponential, are also required for numerical device simulations, eg some analytical models for carrier mobility involve square root calculations, and therefore must also be supported by the mP. These more complex operations and functions can be evaluated using numerical methods: Newton-Raphson for division and square root calculations and Taylor series expansion or Chebychev polynomials for the logarithmic, exponential and other functions, such as trigonometric or hyperbolic. Numerical evaluations of functions are broken down into series of additions and multiplications, as discussed for example in [Lanc60], [Clen62] [LyCY65], and [Fike68]. Consequently, it is of great importance that the operations of addition and multiplication be performed by the mP as fast as possible, for best performance to be achieved. Furthermore, it is necessary that addition and multiplication be performed using floating point data formats. The motivation for floating point arithmetic comes from the necessity to provide a high dynamic range for data in both the numerical solution of partial differential equations and other engineering applications, such as digital signal processing [HKMT83], [FoOt84], [Snyd84], [ShDa86].

As a result of the above requirements a full hardware implementation of these two *basic* operations is provided within the mP architecture in the form of a floating point arithmetic unit containing a floating point adder (denoted here the Floating Point Adder) and a floating point multiplier (denoted here the Floating Point Multiplier), both of which are described in full detail in the next two chapters. The main objective in the design of these two functional blocks is to incorporate a sufficient level of architectural innovation to allow the mP system to perform these two most frequent arithmetic operations (addition and multiplication) as fast as possible. In addition, the high arithmetic performance required of the mP dictates that these operations be performed in a single instruction cycle. Single cycle operation and the consequent absence of pipeline delays within the floating point arithmetic unit make it possible to use the result of one calculation as the input operand for the very next calculation. This is a crucial feature for implementing algorithms with tight data feedback loops, eg forward or backward substitutions and other recursive algorithms.

The floating point data widths are dictated by the requirements of the numerical algorithms which require a data width of at least 32 bits [ShDa86]. Presently there are more than 20 different floating point formats [WaFl82] in use by various computer manufacturers. This situation which prohibits data portability produced by numerical software was the main motivation for the setting up in 1978 of an IEEE (Computer Society)

committee to standardize floating point arithmetic. Specifically, the concern was to devise rounding schemes for maximum precision and to specify exceptional conditions and what to do in each case, ie overflow, underflow etc. In 1985 such a standard was published [IEEE85] and there are indications that most of the computer and microprocessor manufacturers will closely adhere to its rules and recommendations, eg Intel 8087 floating point co-processor, the set of floating point chips released by Weitek and the new signal processing chip TMS 320-C30 released by Texas Instruments all implement the IEEE Floating Point Standard.

Although the data representation specified by the IEEE Floating Point Standard is not the most suitable for hardware realization (two's complements mantissa representation would be preferred to signed magnitude representation) it was decided to adhere, in the design work reported in this thesis, to the IEEE Floating Point Standard representation for the reasons of compatability with other arithmetic chips which are quickly entering the market, and to provide the best numerical performance by using data rounding specified by the Standard.

6.4.2 Logical Architecture

The main objective in specifying the logical architecture of the mP was to provide sufficient data routing flexibility to ensure that the two most complex modules viz the Floating Point Adder and the Floating Point Multiplier could operate in parallel as much as possible during the execution of the various computational tasks described in Sections 6.2.1 and 6.3.1. The major data transfer operations and the required data communication channels, within the mP and between mPs, have been identified in these two sections and depicted in the series of figures, 6.3, 6.9, 6.10, 6.11, 6.12 and 6.13.

A block diagram showing the proposed logical architecture of the mP is given in Figure 6.14. The logical architecture is separated into processing and control sections. For clarity (and because its design is not considered here), the control section, envisaged as a program memory, decoder, program controller and a read only memory (ROM) for storing the constants and the microcode necessary for the evaluation of mathematical functions using numerical methods, is not shown in the figure. Apart from control signals to drive the data processing elements and status flags from the data processing elements back to the control, the only other connection between control and data processing sections which is provided is that to down-load data, such as values of fixed constants, from the program memory to the data processing elements.

The two main data handling elements include the Floating Point Adder and the Floating

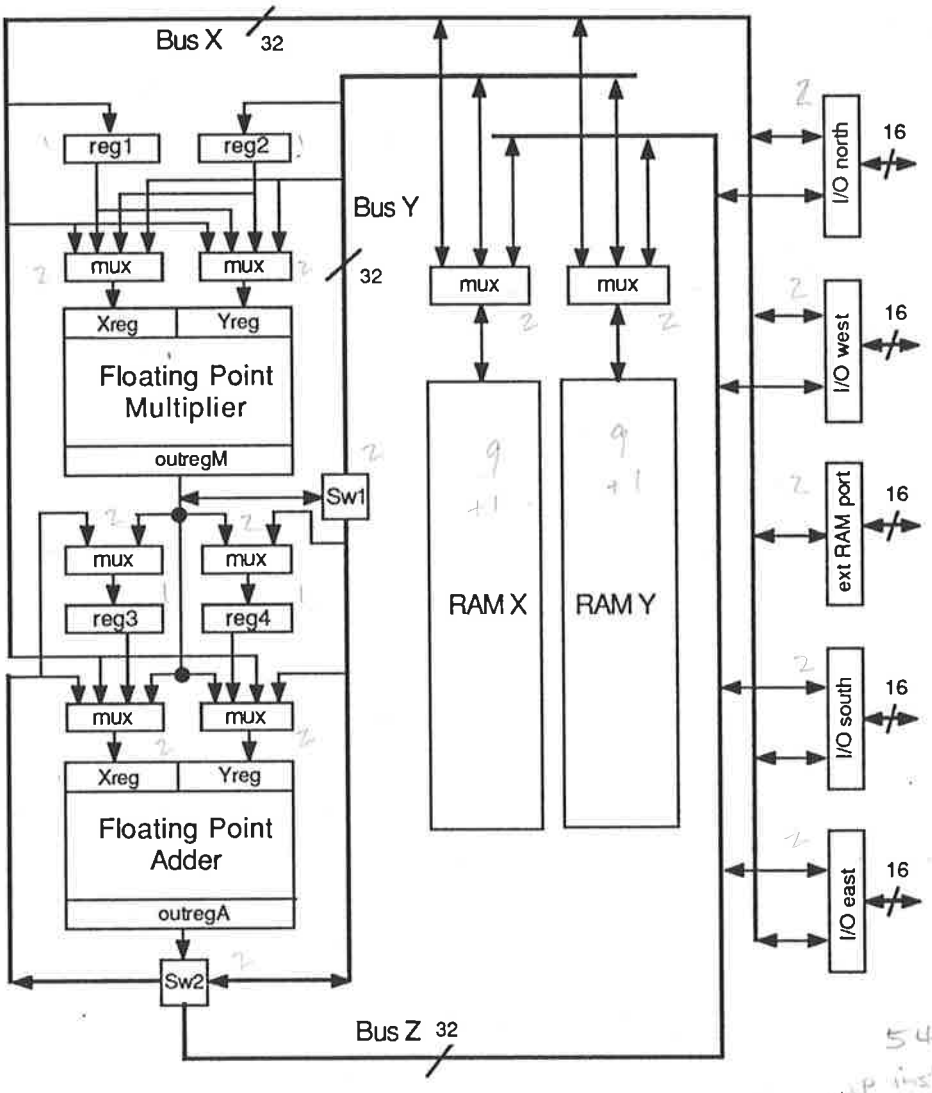


Figure 6.14: Mathematical Processor: the logical architecture.

Point Multiplier. Both functional blocks have been implemented with static CMOS combinatorial logic to achieve the shortest possible latency. In order to maintain high degree of internal parallelism, ie to have both the Adder and the Multiplier perform useful work on every clock cycle, it would be necessary, in the most general case, to supply two new data to and collect a new result from both the Adder and the Multiplier during every cycle of the system clock (the system in this case being the mP). This, however, would result in a very complex and expensive designs of the data memory, the control section, and the data buses on the chip. The analysis of the various algorithms indicates that in most cases a maximum of three different simultaneous data transactions are sufficient in any clock cycle, with at least one datum obtained from an I/O Port.

As a result of the above considerations three 32-bit buses (X, Y and Z) are included on the chip. The limitation imposed by this number of buses seems to be small and is alleviated by the ability of each of the functional blocks to hold its inputs and outputs

over more than one clock cycle, and by the provision of four general data storage registers: reg1, reg2, reg3 and reg4, in addition to the main data memory (RAM X and RAM Y). The manner in which the buses are connected to the data processing elements facilitates the parallel loading of two new operands and the unloading of one result, or the parallel loading of three new operands to the floating point unit, in one instruction cycle. All buses may be used to load the floating point unit with operands, with buses Y and Z also used for unloading the results.

In order to increase the flexibility of the data buses two reconfigurable bus switches (Sw1 and Sw2) are provided. The switches operate under program control and are used for the setting up of data feedback paths from the output registers of the Adder (outregA) and the Multiplier (outregM) to the input registers of these two functional blocks. Such *active* data channels allow an intermediate result of a calculation generated during one clock cycle to be used as an input operand to the Adder or the Multiplier in the very next clock cycle. The two bus switches are identical and their possible configurations are shown in Figure 6.15. Note that only three terminals of Sw1 are used and the physical orientation of Sw1 is the mirror reflection of Sw2 about the horizontal axis. The orientation of Sw2 being as shown in Figure 6.15.

With the use of these switches it is possible to configure buses Y and Z to provide any data path identified in Section 6.3.1. For example, in order to establish the required data paths for cycle 3 of the Chebychev acceleration method shown in Figure 6.12, Sw1 is set as shown in Figure 6.15(b) and Sw2 is set as shown in Figure 6.15(c). With this setting of the bus switches the required data feedback path from the output register of the Adder to the right hand side input of the Multiplier is established via bus Y, the data path between the data memory and the left hand side input of the Adder is established via bus Z, and finally the data path between the data memory and the right hand side input of the Adder is provided via bus X.

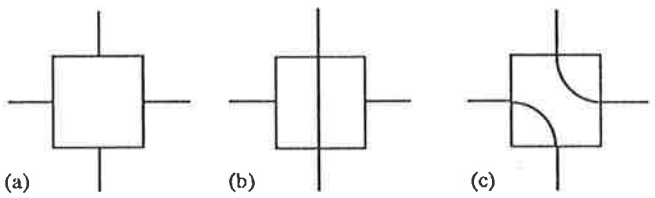


Figure 6.15: Possible switch configurations (a) off state, (b) connection of two opposite terminals, (c) connection of the two pairs of adjacent terminals.

The double port data memory, which is required on-chip, is implemented as two single port memory blocks, RAM X and RAM Y. Both memory blocks have read and write ac-

cess to the three system buses. This arrangement allows two simultaneous data memory accesses to be carried out in one clock cycle, which satisfies the on-chip communication requirements identified in Section 6.3.1.

The data memory requirements of the numerical methods, studied in Chapter 5, dictate that the data storage capacity of the mP be as large as possible, thus the size (in terms of the number of words) of the on-chip data memory is fully determined by the available silicon area. However, it is more than likely that the data storage requirements of some problems will exceed the data storage made available on the chip, irrespective of the size of the provided data memory. For this reason an external data memory port (ext RAM port) is included as part of the mP logical architecture, making it possible for the on-chip data memory to be extended by an addition of an extra memory chip directly connected to the mP. In order to simplify the data communication to the external data memory, it is envisaged that data transfers will be carried out using the "burst mode". The address of the first data word is made available to the external data memory via the external RAM port for only the first clock cycle. The data is then transferred at a rate of one word per clock cycle and it is the responsibility of the external data memory control hardware to provide incrementing addresses to the external data memory for every clock cycle in which the burst-request signal is active. This operation being carried out independently of any other activity within the mP.

Efficient implementation of the numerical algorithms discussed in Section 6.3.1 and many signal processing algorithms requires wide input-output ports for inter-processor communication. Four separate I/O Ports are accommodated within the mP to meet the high data bandwidth requirements and facilitate efficient array operation. These functional blocks perform data input and output operations under program control, relieving the central control unit of the handshaking and sequencing tasks. Each I/O Port has two 32-bit data registers. Data is read from or written to these registers under the control of the program, but the actual communication with the other mPs is independently handled by the I/O Ports.

Both synchronous and asynchronous I/O modes are supported. Synchronous mode of data transfer is mainly intended for communication between two mPs, while asynchronous operation is primarily intended for communication with foreign devices.

In order to keep the system clock period short inter-chip communication is overlapped with instruction execution. Thus, a value which is computed in a given clock cycle can be used by the same mP in the very next cycle, but not until the cycle after that on a neighbouring mP. One data word can be transmitted or received via any of the five ports in one clock cycle with a complete 32-bit word being transferred in two 16-bit half-

words on two phases of the system clock. This communication scheme also allows the inter-chip data paths to be only 16-bit wide, greatly reducing the number of necessary pins which are a scarce resource in VLSI technology [RaMa81], [Snyd84]. (Similar I/O Ports were included in the TFB chip and their designs were carried out by the author of this thesis. Some details can be found in Appendix A.)

Another means of removing "housekeeping" tasks from the control unit is employed in the data memory. Two independent pointers address each memory block (RAM X and RAM Y) and each pointer can automatically increment in a number of ways. This allows the programmer to set up regular data structures and the recursive execution of the program utilizes the auto-incrementing capability to select the required data.

The provision of separate data and program memories on the mP chip enables the fetching of information from data and program memories in parallel. Thus, instruction execution overlaps the fetch cycle of the next instruction. Some of the initial requirements placed on the instruction set include:

1. Arithmetic instructions: add, subtract, multiply, divide, exponential, logarithmic, trigonometric.

2. Data transfer instructions: data transfers between the on-chip data memory blocks, the I/O Ports and the internal storage registers and the floating. point unit.

3. Program flow control instructions: conditional and unconditional jumps, conditional and unconditional subroutine calls.

As stated at the beginning of this section the design of the control unit is not considered in this work. However, for maximum on-chip parallelism it is envisaged that each instruction word will include a sufficient number of fields to specify control signals for all data handling elements. The width of the program word cannot be determined at this stage, however, as it is highly dependent on the physical implementation of the logical architecture.

The clocking strategy is based on a three phase nonoverlapping clock, primarily to ease the communication problems associated with the highly interconnected parallel data processing elements included on the chip. This clocking strategy is based on the clocking strategy developed for the TFB chip which is briefly described in Appendix A. As far as the data handling elements are concerned, Phase 1 of the system clock is reserved for passing information on the three buses and for evaluation of any conditional expressions that may be exist at that program step, while Phases 2 and 3 are primarily operation

cycles; ie multiplication, addition or communication off-chip.

Because of the complexity of the problem considered in this work, it is difficult to estimate, at this stage, how fast the TIME Machine can carry out a solution to the basic semiconductor equations. An initial estimation of the "raw computing power" of the TIME Machine and the mP system is therefore only attempted herein. Circuit simulation results, obtained using the FACTS program and presented in Chapters 7 and 8, indicate that the maximum delay of the floating point unit is $66ns$, for the $Orbit^+$ fabrication process (the delay of the Floating Point Adder is $66ns$ and that of the Floating Point Multiplier is $59ns$, thus the delay of the Floating Point Adder determines the period of the system clock). According to these results the minimum period of the three phase clock of the mP system, assuming that Phases 2 and 3 and their separation period take $66ns$ and allowing an additional $34ns$ for Phase 1 and its separation to Phases 2 and 3, can be set at approximately $100ns$. This corresponds to a maximum performance of the mP system of 10^6 floating point additions and 10^6 floating point multiplications, or 22 MFLOPS (million floating point operations per second). Therefore, the TIME Machine composed of 4096 mPs connected as the two-dimensional array could, theoretically, deliver a maximum computing power of approximately 8200 MFLOPS.

6.4.3 Transformation to Physical Architecture

The transformation of the proposed logical architecture into a physical architecture was performed with the aims of maximizing the circuits's regularity, planarity and hierarchical partitioning while minimizing the inter-module communication costs, as discussed in Chapter 2. The design is targeted at the 1-micrometer double metal p-well CMOS process ($Orbit^+$), discussed in Section 4.6.

The floorplan of the Mathematical Processor system is shown in Figure 6.16. The second layer of metallization is mainly used for global power distribution and the three data buses. This allows for the buses to be placed *on top* of the active elements (with a maximum of two bus tracks per bit slice of the Multiplier, Adder, RAM X and RAM Y modules), thus to a large degree minimizing the silicon area occupied by the interconnect.

In order to determine whether a single chip implementation of the proposed logical architecture is feasible it is necessary to approximate the sizes of the various functional blocks. The critical elements are the Floating Point Adder and the Floating Point Multiplier modules. The silicon area of the Floating Point Multiplier was initially

estimated using previous designs carried out by the author of this thesis and the results presented in the literature describing designs of fixed point multiplier circuits:

1. 8×8 bit modified Booth's algorithm integer multiplier implemented during the course of this work using a 5-micrometer CMOS process occupies $6.1 \times 10^{-6} m^2$;
2. 10×10 bit integer multiplier realized with a 2.5-micrometer MESFET process occupies $2.4 \times 10^{-6} m^2$ [HaRP82];
3. 8×8 bit integer multiplier realized with a 1.5-micrometer NMOS process occupies $0.35 \times 10^{-6} m^2$ [LeGS87];
4. 64×64 bit multiplier realized with a 1.2-micrometer CMOS process occupies $18.5 \times 10^{-6} m^2$ [TIIK85].

From these data the area of a 24×24 bit integer multiplier (the single precision floating point word format specified by the IEEE Floating Point Standard contains a 24-bit mantissa, Section 6.4.4) implemented using the $Orbit^+$ process is estimated to be $2.2 \times 10^{-6} m^2$, by applying the simple scaling principle. It is estimated that the additional circuitry needed to provide floating point capability will occupy an additional 20% of this area giving a total of $2.64 \times 10^{-6} m^2$ or a square with $1.62 \times 10^{-3} m$ on the side.

It is more difficult to estimate the area of the Floating Point Adder module, since the circuitry needed to convert an integer adder to a floating point adder occupies most of the total silicon area. Thus, it was initially assumed that the Floating Point Adder module occupied the same area as the Floating Point Multiplier module.

The actual areas of the two functional blocks which were completely designed and implemented using the symbolic style of design (as discussed in Chapters 7 and 8) are $(1.30 \times 10^{-3} m) \times (1.40 \times 10^{-3} m) = 1.82 \times 10^{-6} m^2$ for the Floating Point Adder and $(1.78 \times 10^{-3} m) \times (1.63 \times 10^{-3} m) = 2.90 \times 10^{-6} m^2$ for the Floating Point Multiplier. Thus, the original estimation of the silicon area occupied by the Floating Point Multiplier module was reasonably accurate.

The silicon area occupied by the 16-bit I/O Port module is estimated from the actual design of a similar functional block carried out by the author in connection with the TFB project to be $(0.55 \times 10^{-3} m) \times (0.15 \times 10^{-3} m) = 0.075 \times 10^{-6} m^2$.

The area occupied by the data memory is estimated on the basis of the following literature results:

1. $64K$ SRAM (static random access memory), 2-micrometer CMOS process, six

transistor cell, arranged as two blocks of 256×128 bits with an access time of $80ns$ occupies an area $(5.92 \times 10^{-3}m) \times (7.49 \times 10^{-3}m)$ [OHYM82];

2. $64K$ SRAM, 2-micrometer CMOS double poly process, four transistor with polysilicon resistive load cell, arranged as two blocks of 256×128 bits with an access time of $65ns$ occupies an area of $(4.87 \times 10^{-3}m) \times (7.22 \times 10^{-3}m)$ [UIMI82];

3. $256K$ SRAM, 1.3-micrometer CMOS double poly process, four transistor with polysilicon resistive load cell, arranged as four block of 512×128 bits with an access time of $45ns$ occupies an area $(4.98 \times 10^{-3}m) \times (9.16 \times 10^{-3}m)$ [YTNM85];

4. $256K$ SRAM, 1.3-micrometer CMOS double poly process, four transistor with polysilicon resistive load cell, arranged as four blocks of 512×128 bits with an access time of $45ns$ occupies an area $(5.41 \times 10^{-3}m) \times (9.18 \times 10^{-3}m)$ [SAIW85].

If each of RAM X and RAM Y is designed as a SRAM arranged as 512 words by 32 bits then it is estimated, from the above results, that each of RAM X and RAM Y will occupy an approximate area of $(1.2 \times 10^{-3}m) \times (7.2 \times 10^{-3}m) = 8.6 \times 10^{-6}m^2$, assuming the $Orbit^+$ fabrication process design rules.

Assuming that a microprogram word is 40 bits long and the the program memory contains 256 words, then a SRAM implementation of this function block will occupy an area of approximately $(1.5 \times 10^{-3}m) \times (3.6 \times 10^{-3}m) = 5.4 \times 10^{-6}m^2$. At this stage of the overall design process it is difficult to accurately estimate the silicon area occupied by the remaining parts of the control section. It is assumed, however, that the free space which is present between the data processing elements will be sufficient for the placement of the instruction decoder logic and the ROM which is required to support numerical evaluations of mathematical functions, and for the routing of the control signals.

With these area estimations of the various data handling elements the total chip area of the mP is approximately $10^{-2}m \times 10^{-2}m$ which is well within the capabilities of the present day technology. According to [PaSe80], [FKMD84] and [Snyd84] any near-future improvements in circuit densities can be used more effectively by increasing the functionality and the on-chip memory of the mP system rather than by implementing more mPs on a single chip.

Having found a suitable transformation to the physical architecture which conserves all the features of the logical architecture, the physical architecture of the various functional blocks may be specified more completely, to allow different groups of designers to work on various functional blocks of the mP system within known functional and interfacing

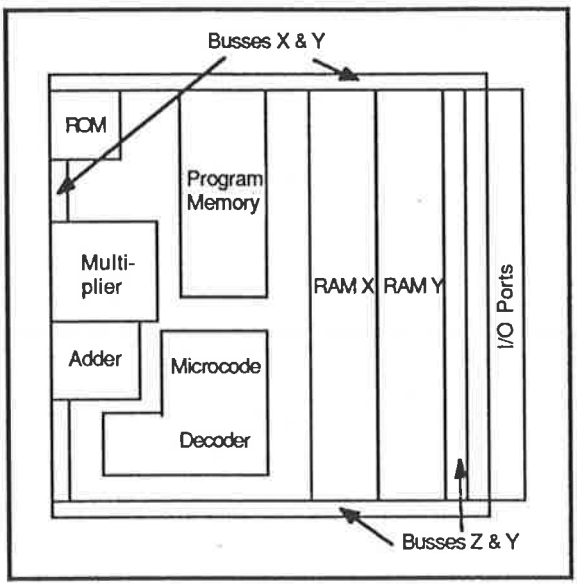


Figure 6.16: Mathematical Processor: floorplan.

constraints. Of main interest in this work are the Floating Point Adder and Multiplier modules, which are considered to be (apart from the control section) the most complex elements of the mP system. The main specification aspects of these two modules are summarized below:

1. Floating Point Adder: This module is to perform the floating point addition (subtraction) of two 32-bit operands according to the specifications of the IEEE Floating Point Standard. The operands are to be latched into the two adder input registers (Xreg and Yreg) on Phase 2 of the clock, and the result is to be latched into the adder output register (outregA) on Phase 3 of the clock. Since this operation is likely to determine the overall system clock frequency, (the length and the separation of Phases 2 and 3 of the clock) it is required that it be completed in the shortest time possible with the target technology.

2. Floating Point Multiplier: This module is to perform the floating point multiplication of two 32-bit operands according to the specifications of the IEEE Floating Point Standard. The operands are to be latched into the two multiplier input registers (Xreg and Yreg) on Phase 2 of the clock, and the result is to be latched into the multiplier output register (outregM) on Phase 3 of the clock. Since this operation is also likely to determine the overall system clock frequency, it is required that it be completed in the shortest time possible with the target technology.

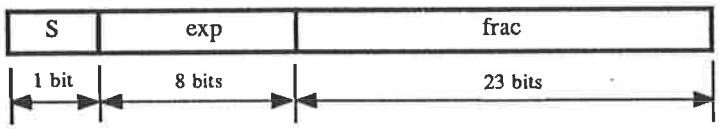
The aspects of the IEEE Floating Point Standard, which are implemented by and influence the design of each of these two modules, are described in the following section.

S	exp	$frac$	Quantity.
0	0	0	+zero
1	0	0	−zero
0	255	0	$+\infty$
1	255	0	$-\infty$
0	0	Not zero	+Denormalized numbers
1	0	Not zero	−Denormalized numbers
X	255	Not zero	NAN (not a number)

Table 6.1: Reserved operands.

6.4.4 Floating Point Arithmetic

The single precision IEEE Floating Point Standard format is presented in Figure 6.17 [Coon80], [WaFl82], [IEEE85]. A normalized number (A) is represented by a sign bit (S) which is positioned at the left hand most end of the word (bit position 32), by an 8-bit exponent (exp) with a positive bias of 127 occupying bit positions 24 to 31 and a 23-bit fraction ($frac$) occupying bit positions 1 to 23, which fraction together with an implied leading 1 (hidden 1) and an implied binary point to the right of the hidden 1 yields the mantissa field $man = 1.frac$. Thus, normalized nonzero single precision numbers can range between $2^{-126} \times 1.0$ and $2^{127} \times (2 - 2^{-23})$ inclusive. Table 6.1 gives the Standard's representations of the reserved operands; zero, infinity, denormalized and NAN (not a number).



S = sign bit.
exp = biased exponent; E = true exponent = exp-bias.
frac = fraction; man = true mantissa = 1.frac.

$$A = \text{normalized number} = (-1)^S * 2^E * man.$$

Figure 6.17: Single Precision IEEE Standard Format.

There are four rounding modes specified by the Standard:

1. RN: unbiased rounding to nearest (in case of a tie round to even);
2. RZ: round toward zero (truncate);
3. RM: round toward minus infinity; and

$manr_{n,4}$	G_1	$G_2 + S$	Condition.	Rnd
DC	0	0	Exact result	DC
DC	0	1	Mantissa is rounded correctly.	DC
0	1	0	Mantissa is even. No rounding required.	0
1	1	0	Mantissa is odd. Round to nearest even.	1
DC	1	1	Round to nearest.	1

Table 6.2: The value of the bit *Rnd* to be added to $manr_n$ in the position of $G_1 = manr_{n,3}$ in order to obtain *unbiased rounding to nearest.*

4. RP: round toward plus infinity.

Only the first rounding mode is mandatory and the other three are optional.

The final output delivered by both the Floating Point Adder and the Floating Point Multiplier modules is rounded according to the RN mode. However, it is recognized that with the hardware provided the implementation of the remaining rounding modes is possible with some modifications to the design.

In order to perform this rounding scheme correctly use is made of a quantity called the "sticky bit". The "sticky bit" is a 1-bit field that identifies whether a shifting (to the right) operation has resulted in a 1 being shifted out of the second guard bit. The first (G_1) and the second (G_2) guard bits are two additional bits immediately to the right of the least significant bit of the mantissa, with the first guard bit being more significant than the second guard bit. Thus, the extended mantissa (man) contains 27 bits: 24 being the mantissa field (bit positions man_4 to man_{27}), with man_4 being the least significant mantissa bit, plus the two guard bits ($G_1 = man_3$, $G_2 = man_2$) and the "sticky bit" ($S = man_1$).

The value of bit *Rnd* to be added to a normalized resultant mantissa ($manr_n$) in the position of the first guard bit in order to obtain *unbiased rounding to nearest* (RN) is given in Table 6.2. (A normalized resultant mantissa is the result of mantissa addition, mantissa subtraction or mantissa multiplication following its normalization.)

The other relevant piece of information is the specification of responses to exceptional conditions. The procedures specified by the Standard to be followed in cases of overflow or underflow are rather complex from the point of view of hardware implementation. For example, the Standard requires that any underflowed result be denormalized before being rounded and delivered as the final result. This is called *gradual underflow.*

It was decided to adopt a simpler approach in which the hardware detects the excep-

tions, generates constants and sets appropriate flags (the flags are also required by the Standard). The desired action to be taken can then be implemented by the software. In particular, in a case of a result overflow the *overflow* flag (*OVF*) is set and the Standard's representation of infinity with the appropriate sign is delivered as the final result. Such a procedure is, in fact, in accordance with the Standard for the case of the overflow trap being disabled.

Implementation of gradual underflow would further complicate what are already complex functional blocks and would require the hardware to recognize denormalized numbers as valid operands. Since it is much faster and more economical to generate a zero output on underflow, such an approach has been adopted, ie on result underflow the *underflow* flags (*UNF*) is set and the Standard's representation of zero with the appropriate sign is delivered as the final result. Another reason that supports this decision has been given by Fraley [WaFl82], who states that there is no sufficient documented need for *gradual underflow.*

Thus, the adoption of only *normalized numbers* as legal operands is the only major departure from the requirements of the Standard. The invalid operands termed *not a number* are treated as infinity and any denormalized operands are treated as zeros by the floating point unit of the Mathematical Processor system.

In addition the *invalid operand* flags, the *inexact result* flag (*Inx*) and the *zero result* flag (*Z*) are also implemented in both functional blocks of the floating point unit. The invalid operand flags indicate that one or both of the operands are not valid, ie infinity, denormalized or NAN. The *Inx* flag indicates that a round-off error has taken place and the purpose of this flag is to allow integer calculations with a floating point unit.

6.5 Summary

In this chapter the main computational and communication requirements of common algorithms used to solve linear systems of equations have been examined. A two-dimensional array processor is well suited to performing the necessary calculations involved in such solution methods as SOR with red-black node partitioning, Chebychev or conjugate gradient. The calculations can be arranged as short repetitive program loops, with all PEs carrying out such computations simultaneously, and in most cases the nearest neighbour interconnection scheme is sufficient.

The main problem with the two-dimensional array of PEs is the large communication cost involved in evaluating vector inner product, which operation require a sum of num-

bers, stored one per PE, to be evaluated. It has been shown in this chapter that the cost of this operation is $\log_2 x$ additions and x move operations on a two-dimensional square array, with the total number of PEs being x^2, and that the large communication cost will be the performance limiting factor in the SOR method and the Chebychev acceleration method, where vector inner product calculations are necessary for an adaptive estimation of the spectral radius of the iteration matrix (needed for the calculation of iteration parameters). The communication cost is even more severe in the conjugate gradient acceleration method, where two vector inner product calculations are required at each iteration.

The TIME Machine architecture solves the above communication "bottleneck" by enabling the above calculations to be carried out at a cost of $\log_2 x^2$ additions and $\log_2 x^2$ move operations, which is the lowest possible time cost. This substantial improvement in performance has been achieved by introducing a number of (6.25% of the initial PEs in the array) additional PEs embedded within the original two-dimensional array, and connected as the binary tree. This architecture combines the characteristics of the two-dimensional and binary tree structures, which greatly extends its range of applications.

Consideration of the tasks which need to be carried out by the individual PE, resulted in a proposal of a logical architecture of the Mathematical Processor. It has been shown that the processor may be implemented as a single chip (with $10^{-2}m$ on a side) device using the hypothetical 1-micrometer *Orbit*$^+$ process. The main architectural characteristics of the Mathematical Processor include the reconfigurable bus switches which allow for a number of data transfer paths to be actively provided on the chip, the Floating Point Adder and the Floating Point Multiplier. The performance of the mP is determined by the time delays within the floating point unit. Therefore, large amount of effort, during the course of this project, has been directed towards developing fast architectures for this complex functional element. The complete and novel implementations of the Floating Point Adder and Multiplier are the subjects of the next two chapters.

Chapter 7

The Floating Point Adder

7.1 Introduction

A novel VLSI realization of floating point addition, the Floating Point Adder (FPA) system, is described in this chapter. This functional block constitutes one of the most complex MCM level modules within the Mathematical Processor architecture.

A number of floating point adder chips have been made available in the recent years [WaMc82], [WMCS82], [HKMT83], [WoLO83], [WiWi84], [WLWW84], [GaOx84], [SiOC86], [WiFa86]. All of them have been realized as a direct silicon mapping of the conventional floating point addition algorithm, described in [Flor63], [Hwan79], [Kuck81] and [WaFl82], as commonly performed with software. The throughput of such circuits is typically improved by introducing three or four stages of pipelining.

The main objective in this work is to develop a fast floating point adder architecture by making a more efficient utilization of silicon, and without necessarily resorting to pipelining. The conventional algorithm for floating point addition is analysed and the sequence of operations is modified, in order to introduce more parallelism into this time consuming operation. A silicon mapping of this new form of the floating point algorithm results in a microelectronics circuit in which more functional elements operate concurrently, giving an improved performance over the conventional hardware realizations.

FACTS simulation results are presented to illustrate the operation of the Floating Point Adder system, to predict its maximum latency for the $Orbit^{+}$ process implementation and to compare its performance with the corresponding circuit performing the operation of floating point addition according to the conventional algorithm.

7.2 Free Form Specification

The FPA system performs the operation of normalized floating point addition (subtraction) on two single precision floating point format, normalized operands in one cycle of the Mathematical Processor system clock. The floating point format of the normalized operands used is according to the IEEE Standard for Floating Point Arithmetic [IEEE85], as discussed in Chapter 6.

A diagram which shows the arrangement of input and output ports is given in Figure 7.1. The input and the output registers which are necessary for this asynchronous system to operate in the synchronous environment of the Mathematical Processor are not shown. The sign, exponent and fractional parts of the operand A are denoted by SA, $expA$ and $fracA$, respectively and the sign, exponent and fractional parts of the operand B are denoted by SB, $expB$ and $fracB$, respectively. The opcode input (OP) determines whether B is to be added to or subtracted from A. The sign, exponent and fractional parts of the result (R) are denoted by SR, $expR$ and $fracR$, respectively. The output flag $invA$ indicates that A is not a valid operand and the output flag $invB$ indicates that B is not a valid operand. The remaining output flags are as defined in Section 6.4.4. (Note that in the text a full use is made of the available fonts, whereas the type of fonts available for the construction of figures was rather limited. As a result subscripts and superscripts do not appear in figures but the symbols representing a given quantity are identical to the corresponding symbols in the text.)

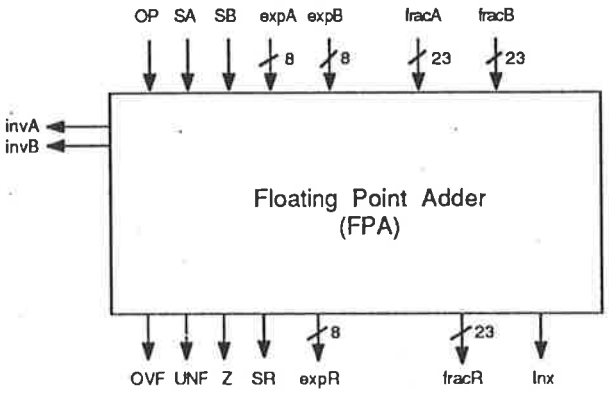


Figure 7.1: The Floating Point Adder system.

7.3 Floating Point Addition Algorithm

The freedom that VLSI technology presents to the circuit designer makes it possible to introduce a degree of, so far unutilized, parallelism into the process of floating point addition, thus, increasing the speed with which it can be executed.

The parallelism is at the module level. This concept will now be defined in more detail. A simple n-bit ripple carry adder, composed of n instances of 1-bit full adder modules, is considered to be a parallel structure from the point of view of applying the inputs, ie all inputs are presented to the adder in parallel. However, not all of its elements perform useful work at any one instant of time. Consider for example, the special case of the carry bit propagating from the least significant bit position to the most significant bit position. Any full adder module can only complete its operation if the neighbouring full adder module, at the bit position with significance one less than the cell in question, has generated its carry bit output. The need for the carry signal to propagate across the chain of modules results in only one full adder module generating its stable output at any one time. Thus, the operation of the ripple carry adder at the module level is serial for such a combination of input.

Increasing the parallelism at the module level results in more modules producing their final results in parallel and independently of the remaining modules. For example if the inputs to the ripple carry adder are such that no carry bits are generated, then all full adder modules complete their operation in the same time. This is a case of 100% module parallelism.

In the subsections to follow, the conventional form of the floating point addition algorithm is introduced and analysed. A modification is then presented that results in an algorithm in which more tasks which were originally performed sequentially are now carried out concurrently, making it possible for the hardware realization to operate with a higher degree of parallelism at the module level.

7.3.1 The Conventional Form

The usual form of the algorithm for normalized floating point addition is presented below (all tasks combined together are normally carried out concurrently by various modules), and a block diagram showing its conventional hardware realization (denoted here the CFPA system) is given in Figure 7.2.

1. The operand exponents are compared to determine the intermediate value of the resultant exponent (the larger exponent). The positive exponent difference is computed to establish the alignment shift distance (*shift*) by which the mantissa of the smaller operand must be unnormalized (shifted right). The implicit bits are generated and invalid operands detected. The effective mantissa operation is determined.

2. The required n-bit mantissa is unnormalized and the resulting $(n + shift)$-bit word is rounded to an $(n + 3)$-bit word according to the RN rounding algorithm.

3. One of the mantissas is complemented if the effective mantissa operation to be carried out is a subtraction.

4. The two mantissas are added (or subtracted) producing at most an $n + 4$ bit result. The result is composed of 1 bit for possible overflow, n bits of the format, the first and the second guard bits and the "sticky bit". (The two guard bits and the "sticky bit" are defined in Section 6.4.4.)

5. The result of mantissa subtraction is complemented if it is negative.

6. The position of the leading bit in the resultant mantissa is determined. The case of computed zero which cannot be normalized is detected.

7. The resultant mantissa is normalized if necessary. The normalization shift distance is encoded into a two's complement number.

8. The intermediate value of the resultant exponent is adjusted to compensate for the normalization shift. The normalized resultant mantissa is examined to determine whether a rounding operation is necessary.

9. The normalized resultant mantissa is rounded, if necessary, according to the RN rounding algorithm.

10. The result of the rounding operation is tested for overflow and normalized if required. The resultant exponent is incremented if the rounding operation resulted in an overflow.

11. Finally, the resultant exponent is tested for both overflow and underflow and if either has occurred a constant is generated for the final result.

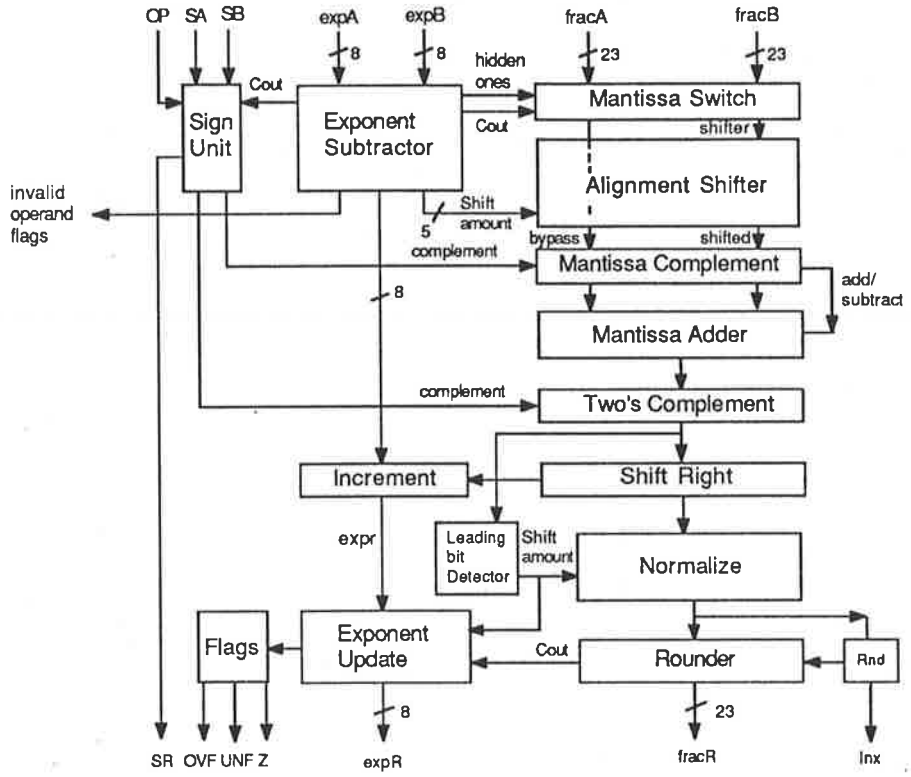


Figure 7.2: A block diagram of the conventional hardware realization of the usual floating point addition algorithm, after [WMCS82].

7.3.2 An Analysis of Normalized Floating Point Addition

In this section some of the properties of normalized floating point addition, which may simplify the design of the hardware, are highlighted. The general case of two n-bit operand mantissas is assumed. In the convention used in this thesis $i = 1$ indicates the position of the "sticky bit", $i = 2$ corresponds to the first guard bit, $i = 3$ corresponds to the second guard bit, $i = 4$ corresponds to the least significant bit and $i = 27$ corresponds to the most significant bit of a the mantissa field of a floating point word. Bit position $i = 28$ corresponds to the most significant bit of an overflowed mantissa sum.

7.3.2.1 Mantissa Addition

The normalized operand mantissa (man_n) is in the range $1 \leq man_n < 2$ and the unnormalized operand mantissa (man_{un}) is in the range $0 \leq man_{un} < 2$, thus, the range of a mantissa sum is $1 \leq manr_{sum} = man_n + man_{un} < 4$, ie the operation of mantissa addition may overflow irrespective of whether one of the operand mantissas has or has not been unnormalized. The overflowed mantissa sum is normalized by

shifting it one place to the right. When there is no mantissa overflow the mantissa sum is available in normalized form.

An additional case, where a mantissa overflow can occur, is during the rounding operation. However, the case of mantissa addition overflow and the case of mantissa rounding overflow are mutually exclusive. This can be demonstrated as follows. If an overflow occurs during mantissa addition then the overflowed mantissa sum before it is normalized ($manr_{sum,un,max}$) has a maximum value given by:

$$\begin{aligned} manr_{sum,un,max} &= [(2^n - 1) + (2^n - 1)]2^{-(n-1)}, \\ &= [2^{n+1} - 2]2^{-(n-1)}. \end{aligned} \tag{7.1}$$

(The factor $2^{-(n-1)}$ shifts the binary point to the position required by the Standard.) The maximum possible value of a normalized mantissa sum ($manr_{sum,n,max}$) which requires rounding is given by the result of (7.1) shifted by one place to the right:

$$\begin{aligned} manr_{sum,n,max} &= manr_{sum,un,max}/2, \\ &= [2^n - 1]2^{-(n-1)}. \end{aligned} \tag{7.2}$$

It is now necessary to show that the minimum value of a normalized mantissa sum which requires rounding and which produces an overflow on rounding is smaller than the result given in (7.2). In order for the operation of rounding to overflow the number being rounded, ie $manr_{sum,n}$, must have the value given by:

$$manr_{sum,n} = [2^n - 1 + 2^{-1}]2^{-(n-1)}, \tag{7.3}$$

which is indeed greater than the value given in expression (7.1).

The only special case which needs to be considered when performing mantissa addition is the addition of two zeros which results in the mantissa sum = 0. There is no leading bit when such a result is generated and this condition must be flagged and a constant generated for the fractional field of the result.

7.3.2.2 Mantissa Subtraction

The result of mantissa subtraction ($manr_{diff}$) is in a range given by $0 \leq manr_{diff} < 2$. Thus, the operation of mantissa subtraction will never overflow and as a consequence a mantissa difference will never require a shift to the right to normalize it.

The absolute value of the smallest possible mantissa difference ($man_{diff,min}$) which is greater than zero is given by:

$$manr_{diff,min} = [(2^{n-1}) - (2^n - 1)/2]2^{-(n-1)} = 2^{-1}2^{-(n-1)}. \tag{7.4}$$

According to expression (7.4) the leading bit lies in a bit position between the most significant bit (bit position 27) and the first guard bit (bit position 3) of a mantissa difference, and will never have the significance equal to the significance of the second guard bit (bit position 2) or the "sticky bit" (bit position 1).

If both operand mantissas are normalized, prior to their subtraction, then the amount of normalization shift (*shift-normalize*) which may need to be applied to the mantissa difference is given by:

$$n \leq \textit{shift-normalize} \leq 0. \tag{7.5}$$

No rounding is ever required in such cases, since both guard bits in the mantissa difference are zero.

If one of the operand mantissas has been unnormalized by one place, prior to mantissa subtraction, then the normalization shift distance is again given by (7.5). However, the rounding operation may also be required. In fact, the rounding condition will only be generated following mantissa subtraction if the normalization shift distance is either zero or one. This result follows since in order to generate a rounding condition the first guard bit of the mantissa difference must be = 1, but when the normalization shift distance is more than one place a zero is shifted into the bit position of the first guard bit.

In the cases where one of the operand mantissas is unnormalized by two or more places, the normalization shift distance is always at most by one place. This can be shown as follows. The minimum difference ($manr_{diff,min,2}$) between a normalized mantissa and a mantissa which has been unnormalized by two places is given by:

$$\begin{aligned} manr_{diff,min,2} &= \left|2^{n-1} - (2^n - 1)2^{-2}\right| 2^{-(n-1)}, \\ &= \left|2 \times 2^{n-2} - 2^{n-2} + 2^{-2}\right| 2^{-(n-1)}, \\ &= \left|2^{n-2} + 2^{-2}\right| 2^{-(n-1)}, \end{aligned} \tag{7.6}$$

which needs a shift of only one place to the left for its normalization.

From the above discussion it can be concluded that mantissa rounding may overflow in some cases and it can only do so if the normalization shift distance is one. It suffices to show that if the normalization shift distance is zero then a rounding operation does not overflow and if it is one then rounding may overflow.

In order for the operation of mantissa rounding to overflow in the case of the normalization shift distance being zero, the mantissa difference must have a value given by expression (7.3). This, however, is not possible since the maximum value of the

mantissa difference is given by the maximum value of a normalized mantissa:

$$manr_{diff,max} = (2^n - 1)2^{-(n-1)}. \tag{7.7}$$

The result of mantissa subtraction may be both positive or negative. If the result is negative and the subtraction is performed using either one's or two's complement notation then a corresponding conversion to signed magnitude notation is necessary. This is depicted in flowchart form in Figure 7.3. In this respect following the Standard

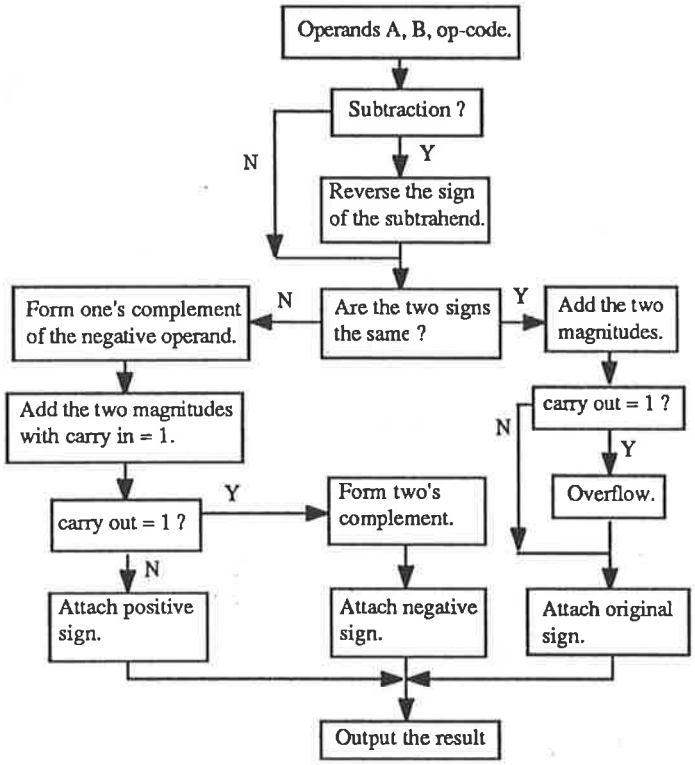


Figure 7.3: Addition-subtraction in signed magnitude notation.

complicates the design of the hardware and two's complement representation of the mantissa field would have been more desirable.

7.3.3 The Modified Form

Analysing the conventional form of the algorithm, from the point of view of the order in which the operations must be carried, it becomes evident that most of the initial steps must be performed sequentially. In particular:

1. Before the mantissas can be aligned the positive difference of the exponents must be established; and

2. The effective mantissa operation must be determined before one of the mantissas can be complemented.

Some of the remaining steps of the algorithm, however, may be performed in parallel. The *new* form of the normalized floating point addition algorithm, developed in this research work, is depicted in Figure 7.4 and the main modifications are specified below:

1. Position of the leading bit is determined in parallel with the evaluation of mantissa sum or difference. This is done by examining the two operand mantissas in their aligned form rather than the resultant mantissa itself.

2. The mantissa rounding condition is detected in parallel with the operation of mantissa normalization.

3. A possible mantissa overflow as a result of mantissa rounding is anticipated ahead. This allows for the operations of exponent adjustment and flag generation to be carried out in parallel with the mantissa rounding operation.

4. Finally the operations of rounding and complementing a negative mantissa difference are also carried out in parallel.

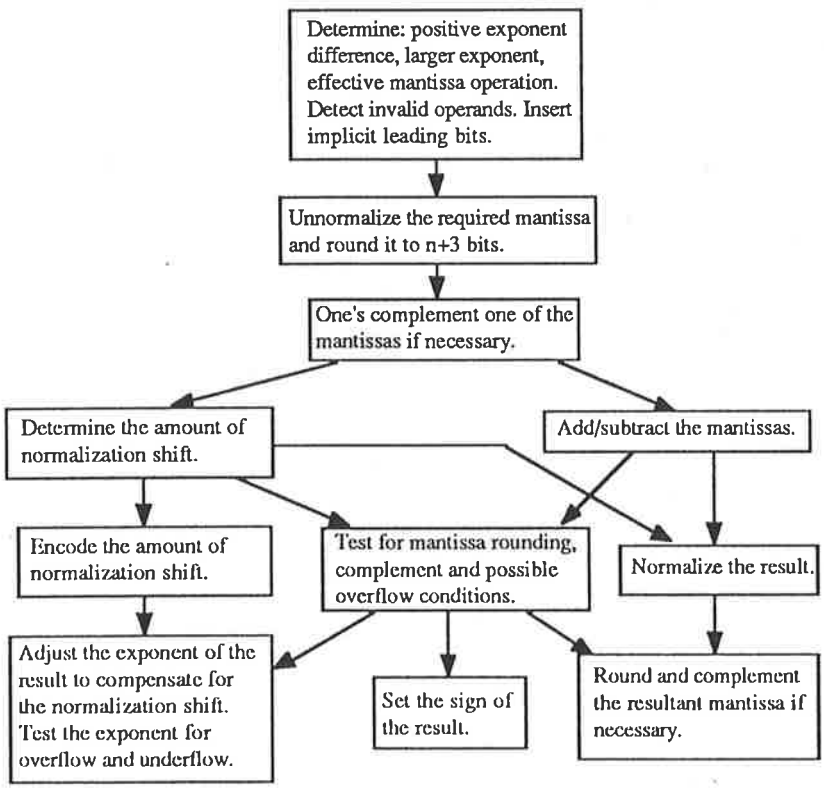


Figure 7.4: The new algorithm for floating point addition.

As a result of these modifications, the proposed architecture operates with a larger degree of parallelism which improves its performance over the conventional hardware realization of floating point addition. The evaluation of the new architecture from the point of view of silicon area requirements and performance is presented in Section 7.8, following a detailed discussion of its physical implementation.

7.4 Functional Specification

A full translation of the system specification in free form (which was given in Section 7.2) to a formal system specification in the functional simulator language ISSL [ISD88] was performed on a scaled system. Apart from using five bits for the mantissa and three bits for the exponent all other aspects of the system have been formally described. Using a reduced system representation enables an exhaustive module based validation of the proposed architecture to be carried out economically, ie only the parts of the system which are functionally different need to be verified. The ISSL description which represents the formal functional description of the scaled FPA system for the case of addition (the code simulating the subtraction is similar but much larger and is not presented) is given in Appendix C. This code constitutes the Specified Function Definition module of Figure 2.8 for the module based validation of the FPA system.

7.5 System Partitioning

The design of the FPA system is exposed in this section following the path of the hierarchical system decomposition. The block diagram of the system is first presented at the first level of decomposition. The block diagrams of the three composition modules identified in this way, namely the Align Unit, the Operate, and the Output Stage are then described in terms of modules which contain one level of hierarchical decomposition and are composed of functional modules only (with the exception of two modules, the Exponent Input Stage and the Exponent Output Stage, which contain two levels of hierarchical decomposition). Following the discussion of the validation of the partitioning and the formulation of the floorplan for the complete FPA system, each composition module is described in terms of its floorplan and logic realization of its functional modules.

The eventual floorplan of the FPA system is given in Figure 7.10. There is a close correspondence between the block diagram of Figure 7.5 and the floorplan diagram of

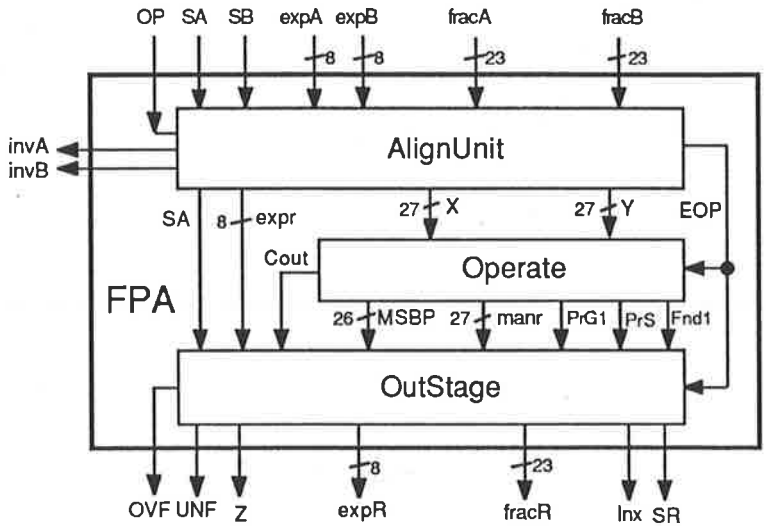


Figure 7.5: Floating Point Adder: first level of decomposition.

the FPA system and indeed this is true for most composition modules. This similarity is a byproduct of the design process in which the constraints for generating a floorplan for a given module and for producing a clear block diagram (one in which the crossover of signal lines in kept to a minimum) are similar.

In the first level of decomposition the FPA system is subdivided into three modules: the Align Unit (AlignUnit), the Operate (Operate) and the Output Stage (OutStage), as shown in Figure 7.5. The inputs of the Align Unit consist of the three fields (sign, exponent and fraction) of the two operands and the opcode (OP), as defined earlier. The outputs generated by this composition module are as follows: the invalid operand flags ($invA$ and $invB$), the intermediate value of the resultant exponent ($expr$), the effective mantissa operation control signal (EOP), and the aligned operand mantissas (X and Y). The output X corresponds to one's complement form of the normalized mantissa, while the output Y corresponds to the one's complement form of the mantissa which passes unchanged through an unnormalizing shifter module. The Align Unit is specified in turn as a composition of five modules, shown in Figure 7.6 and described below.

1. *Sign Unit (SignUnit)*: determines the effective operation, indicated by the control signal EOP, from the signs of the two operands and the opcode (OP), and generates two control signals, CX and CY, specifying which mantissa, if any, is to be negated using one's complement notation.

2. *Exponent Input Stage (ExpInp)*: checks for invalid operands and generates invalid operand flags ($invA$ and $invB$) as required, generates the hidden mantissa bits ($hidA$ and $hidB$), determines the alignment shift distance ($shift$), determines the

intermediate value of the resultant exponent (*expr*), and generates a control signal $C_{o,ex}$ specifying which mantissa is to be unnormalized. The control signal $C_{o,ex}$ is also necessary for the generation of the control signals CX and CY within the Sign Unit.

3. *Mantissa Switch (ManSw)*: operates under the control of the Exponent Input Stage and selects the operand mantissa (*shifter*) which may need to be unnormalized, or generates the constant zero for the mantissa of the operand which is the Standard's representation of zero or infinity, or which is denormalized or a NAN.

4. *Alignment Shifter (Shifter)*: operates under the control of the Exponent Input Stage to perform the specified alignment shift on the selected mantissa (*shifter*), producing an unnormalized mantissa (*shifted*), augmented by three extra bits: the first and second guard bits and the "sticky bit".

5. *Mantissa Complementer (ManComp)*: operates under the control of the Sign Unit and negates, if necessary, (using the one's complement notation) the selected operand mantissa.

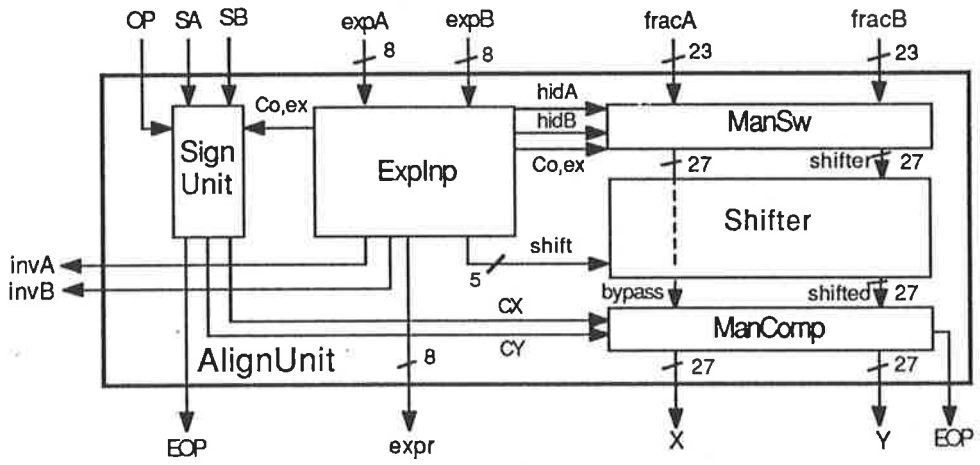


Figure 7.6: Align Unit: first level of decomposition.

The module Operate, operates on the outputs X, Y of the Align Unit and determines mantissa sum or difference (*manr*) according to the control signal EOP, in addition, it generates a control signal $MSBP$ specifying the position of the leading bit in the resultant mantissa, the control signals Pr_{G_1} and Pr_S used to anticipate mantissa overflow as a result of mantissa rounding or complementing, and a signal Fnd_1 indicating that $manr = 0$. It is further subdivided into three modules, as shown in Figure 7.7, described below.

1. *Mantissa Adder (ManAdder)*: calculates the sum or difference ($manr$) of the aligned mantissas, according to the control signal EOP. The carry bit out (C_{out}) from the most significant bit position indicates mantissa sum overflow or a positive mantissa difference. The signals C, G, P and K which are also generated by this module will be defined in the section describing the implementation of this module.

2. *Most Significant Bit Position Finder (MSBPF)*: determines the position of the leading bit (indicated by the signal $MSBP$) in the mantissa sum or difference, and generates the signal Fnd_1 indicating that the output of the Mantissa Adder is a zero; in which case there is no leading bit. The outputs Fnd and PrF are defined in the section describing the implementation of this module.

3. *Propagater*: generates two 1-bit control signals (Pr_{G_1} and Pr_S) which are used to determine whether the operations of mantissa rounding or complementing result in mantissa overflow.

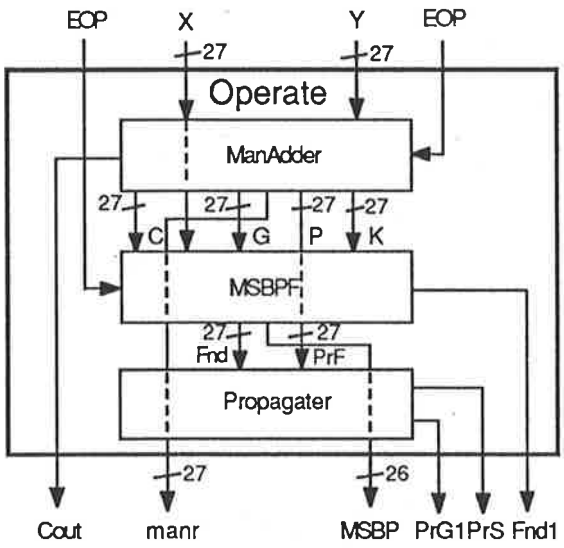


Figure 7.7: Operate: first level of decomposition.

The final functional block is the Output Stage. It is specified as a composition of five modules described below and shown in Figure 7.8.

1. *Mantissa Normalizer (Normalizer)*: generates the signal $manr_n$ which is the normalized version of the result of mantissa addition or subtraction ($manr$), according to the control signals $MSBP$ and C_{out}.

2. *Shift Distance Encoder (PLA2)*: encodes the normalization shift distance ($MSBP$) into a two's complement 5-bit integer (adj) to be subtracted from the intermediate value of the resultant exponent ($expr$).

3. *Special Case Detector (ManFlags)*: detects possible mantissa overflow or mantissa zero result and generates appropriate control signals. The 1-bit control signal *Rnd* specifies that mantissa rounding is required, the 1-bit control signal *cpp* specifies that the two's complement of the normalized resultant mantissa is required and the 1-bit control signal *ManOvf* indicates that one of the following operations: resultant mantissa rounding or complementing, or mantissa addition, results in mantissa overflow. This module also generates the sign bit of the result (SR) and the inexact result flag (Inx).

4. *Exponent Output Stage (ExpOtp)*: performs the exponent adjustment according to the signals *adj* and *ManOvf* and detects result overflow or underflow. In addition this module generates a 1-bit control signal *ManZ* specifying that the fractional part of the final result is to be zero.

5. *Mantissa Rounder-Complementer (ManInc)*: rounds and/or complements the output ($manr_n$) of the Normalizer module or generates zero as the fractional field of the result.

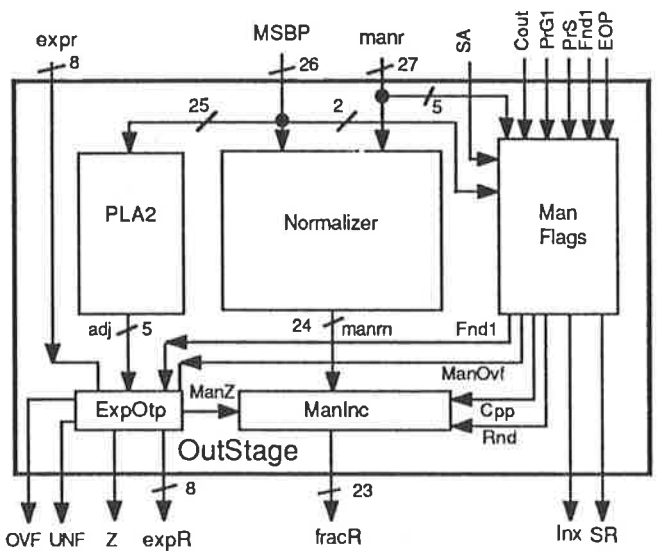


Figure 7.8: Output Stage: first level of decomposition.

The ISSL descriptions of the modules Align Unit, Mantissa Adder and Output Stage were combined to form the top level composed function specification, ie topological definition, of the FPA. This description denoted FPA8 is given in Appendix C.

An alternative representation of the decomposition of the FPA system is shown in Figure 7.9. This diagram depicts how the modules at the lowest level of the hierarchical decomposition are grouped together to form the top level system description and simplifies the visualization of alternate system partitioning strategies which may be more

appropriate in other stages of the design process. The part of the diagram above the broken line represents the decomposition of the FPA system discussed so far. All the modules which lie above the broken line correspond to the second level of decomposition and, with the exception of the Sign Unit and the PLA2, are further decomposed into simple one bit functional modules and in some cases control line driver functional modules. The modules Sign Unit and PLA2 are not decomposed further. The Exponent Input Stage and the Exponent Output Stage contain two levels of hierarchy, one of which is shown below the broken line.

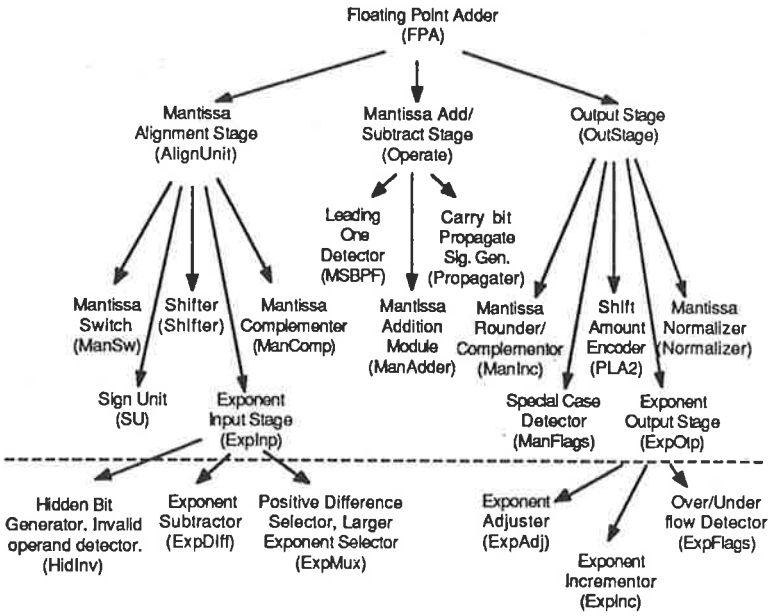


Figure 7.9: Floating Point Adder: hierarchical decomposition.

Before any detailed circuit design can be commenced the translation of the free form specification of the system, given in Section 7.2, into a mathematical language (in this case ISSL) must be verified and the system partitioning just described and the functional system description FPA8Fun given in Appendix C must be validated, as was discussed in Chapter 2.

7.6 Partitioning Validation

The verification of the result of the translation process of the free form specification of the system into the mathematical description was carried out by applying about thirty different combinations of inputs including some special cases and comparing the generated output with hand calculations. This gave a degree of confidence that the top

level functional specification module FPA8Fun was in fact a correct representation of the intended function.

The most rigorous validation of the proposed partitioning would be performed by applying the same excitations to modules FPA8Fun and FPA8 and then comparing their outputs for all possible combinations of inputs. A testing module would be used to write vectors, collect the results, perform the comparison and write the diagnostics to the user. There was a degree of confidence that most of the modules would perform according to specification. It was felt however, that the functional definitions of the MSBPF and the ManFlags were too complex for the designer to account fully for all special input cases, and a formal module based function and partitioning validation of the complete system was performed, as discussed in Section 2.6.2.1.

As expected, as a result of this exercise a number of faults in the original functional specifications of the ManFlags and the MSBPF which passed the informal validation stage were identified and promptly corrected.

7.7 Physical Architecture

The floorplan of the FPA system defined at the first level of partitioning is readily formulated from the block diagram of Figure 7.5. The three composition modules, the Align Unit, the Operate and the Output Stage, are stacked on top of one another, with the data signals passing vertically through them. The data signals enter the FPA system through the input ports of the Align Unit and the results are delivered to the external environment through the output ports of the Output Stage. The planning phase of the design process resulted in the floorplan of the FPA system, defined at the second level of hierarchical partitioning, shown in Figure 7.10. The particular decisions made and the criteria used will now be discussed in more detail. For this purpose, it is more convenient to analyse the FPA system at the second level of partitioning. The modules may then be grouped into those which belong to the mantissa data path, those which belong to the exponent data path and those which belong to the sign data path. The discussion will consider the floorplans for the three modules of the first level of decomposition in turn, although no specific reference will be made to them.

In general, the placement of the various modules is such, that the data signals can vertically ripple through them; leaving one module via the output ports located along the bottom edge of the bounding box and entering the next module via the input ports located along the top edge of the bounding box. In many cases the order of the modules

may be derived from the block diagrams of Figures 7.6, 7.7 and 7.8, but in some cases there are a number of possible module placements and these have to be decided upon.

There are two choices for the first module of the mantissa data path; either the Mantissa Switch or the Mantissa Complementer, since the functions of these two modules are independent of each other, ie it is not necessary to know which of the two mantissas needs to be complemented in order to decide which of the two mantissas needs to be unnormalized. In either case the Alignment Shifter operates on data signals which have already passed through the Mantissa Complementer.

An examination of the functions of the Exponent Input Stage and the Sign Unit shows, that the control signals CX and CY, used by the Mantissa Complementer, become valid before the control signal *shift*, used by the Alignment Shifter, and the control signal $C_{o,ex}$, used by the Mantissa Switch, become available. Thus, placing the Mantissa Complementer at the input of the Mantissa Switch allows for the Mantissa Complementer to operate in parallel with the Exponent Input Stage, ie the selected mantissa can be one's complemented by the Mantissa Complementer in parallel with the task of determining the positive exponent difference, performed by the Exponent Input Stage. Additional benefits of this module positioning are simpler logic expressions for the two control signals CX and CY.

The main disadvantage of this module arrangement is a more complicated implementation of the Alignment Shifter caused by the need to shift both positive and negative operands. This was the major consideration in positioning the Mantissa Complementer at the output of the Alignment Shifter. The small increase in delay (equal to the delay of the Mantissa Complementer), which results from this module placement, is more than compensated for by the simpler hardware realization of the Alignment Shifter, because now positive numbers need to be shifted only. The final order is the Mantissa Switch followed by the Alignment Shifter, followed by the Mantissa Complementer, with the three modules pitch matched and connected together by abutment.

The inputs to the Mantissa Switch are two 24-bit numbers, and the inputs to the Alignment Shifter and the Mantissa Complementer are two 27-bit numbers (one of them simply passes through the module without being altered in any way). By padding the inputs of the Mantissa Switch with three zeros at the least significant end their lengths are extended to 27 bits which results in equal lengths of the three modules.

The control signals necessary for the Mantissa Switch and the Alignment Shifter are generated by the Exponent Input Stage, as shown in Figure 7.6. This module is therefore placed in physical proximity of the other two, to minimize the lengths of the wires

which run between the Exponent Input Stage and the Mantissa Switch, and between the Exponent Input Stage and the Alignment Shifter. All control lines entering the Mantissa Switch, the Alignment Shifter and the Mantissa Complementer span the complete lengths of the respective modules. Thus, space needs to be reserved for control line drivers on the left hand sides of each of the three modules.

The placement of the Sign Unit is dictated, to some extent, by the order of the incoming data, ie the sign bits are at the leftmost side of the data words. This module is placed on the left hand side of the Exponent Input Stage. In this way the lengths of the wires routing the two sign bits are minimized. In addition, the physical proximity of the Sign Unit to the Mantissa Complementer assures that the lengths of the control lines, generated by the former and used by the latter, are also kept to a minimum.

The Mantissa Adder derives its data inputs from the Mantissa Complementer. By placing the Mantissa Adder below the Mantissa Complementer it is possible to economically connect the two modules by abutment, provided proper pitch matching is observed.

It is possible to position either the MSBPF or the Propagater at the output of the Mantissa Adder. However, the coupling between the Mantissa Adder and the MSBPF is stronger than the coupling between the Mantissa Adder and the Propagater (this can be seen from Figure 7.5). Placing the MSBPF at the output of the Mantissa Adder and the Propagater at the output of the MSBPF, results in the simplest possible interconnection between these three modules. Again, by observing proper module pitch matching the modules MSBPF and Mantissa Adder, and the modules MSBPF and Propagater are connected by abutment.

The *EOP* control signal line for the Mantissa Adder enters it at the right hand side, but the logic circuitry (the Sign Unit) generating this control signal is placed on the left hand side boundary of the FPA system. In order to avoid running an extra control line the length of the Mantissa Adder, the control signal *EOP* is regenerated from the control signals *CX* and *CY* which are routed across the length of the Mantissa Complementer (placed immediately above the Mantissa Adder).

The remainder of the floorplan is influenced by the design of the Normalizer. Two options for the implementation of this module are recognized. In the first implementation, the normalization shift control signal (*MSBP*), generated by the MSBPF, is first encoded into the 5-bit integer (*adj*) by the PLA2 before being applied to the Normalizer, as the shift distance control signal. This approach makes it possible to realize the Normalizer as a modification of the Alignment Shifter, but introduces an additional delay into the FPA system and requires a complicated connection between the relevant

modules. The additional delay is caused by the fact that the output of the PLA2 must stabilize before the Normalizer can commence its operation.

In the second implementation, the normalization shift control signal (*MSBP*) is used directly to effect the shift. This makes it possible for the Normalizer to operate concurrently with the PLA2 and results in a simpler module interconnection, but requires that a second shifting circuit be designed. This second circuit is implemented as a modified version of the barrel shifter described in [MeCo80] and [WeEs85].

The interconnection of the relevant modules, when the Normalizer is implemented as a modified barrel shifter, is very natural because by placing the PLA2 next to the Normalizer and on its left hand side, and the Normalizer below the module Operate, the normalize shift distance control signal is made to pass vertically from the module Operate to the Normalizer. The direction of flow of the shift distance control signal lines can be altered by ninety degrees very economically, within the barrel shifter circuit, and directly connected to the PLA2.

The physical structure of the barrel shifter is very regular and basically requires the design of only one functional cell. Thus, the physical effort involved in designing the barrel shifter is very close to the design effort needed to modify the Alignment Shifter to convert it into the Normalizer.

The savings in module interconnect complexity and the absence of the additional delay of the second approach were the main reasons for adopting the barrel shifter implementation of the Normalizer module, in preference to the first implementation involving a modification of the Alignment Shifter.

The placement of the Special Case Detector module (ManFlags) presents special problems. It must be placed in physical proximity to, and at the least significant end of, the module Operate because most of its inputs are generated at this end of the modules Propagater and MSBPF. Although the physical size of the ManFlags is small compared to all the other modules its placement next to the MSBPF or the Propagater would result in an uneconomical utilization of the silicon area. Such a placement would generate, within the bounding box of the FPA system, a lot of "dead" space because the right hand side of the bounding box of the FPA system would be pushed beyond the right hand side edges of the bounding boxes of the Mantissa Switch, the Alignment Shifter, the Mantissa Complementer and the Mantissa Adder.

In order to place the ManFlags in the optimum position with respect to the Normalizer and the MSBPF, the length of the Normalizer is made smaller than the length of the module Operate, and the two modules are connected by a channel routing block. The

ManFlags can now be placed immediately below the module Operate and to the right of the Normalizer.

A routing channel is also used to interconnect the Normalizer and the ManInc, with the ManInc placed immediately below the Normalizer.

Finally the Exponent Output Stage is placed below the PLA2 and connected to it by abutment. The other input to the Exponent Output Stage is derived from the Exponent Input Stage and the interconnection of these two modules is achieved by running a vertical exponent interconnection bus next to, and on the left hand side of, the module Operate.

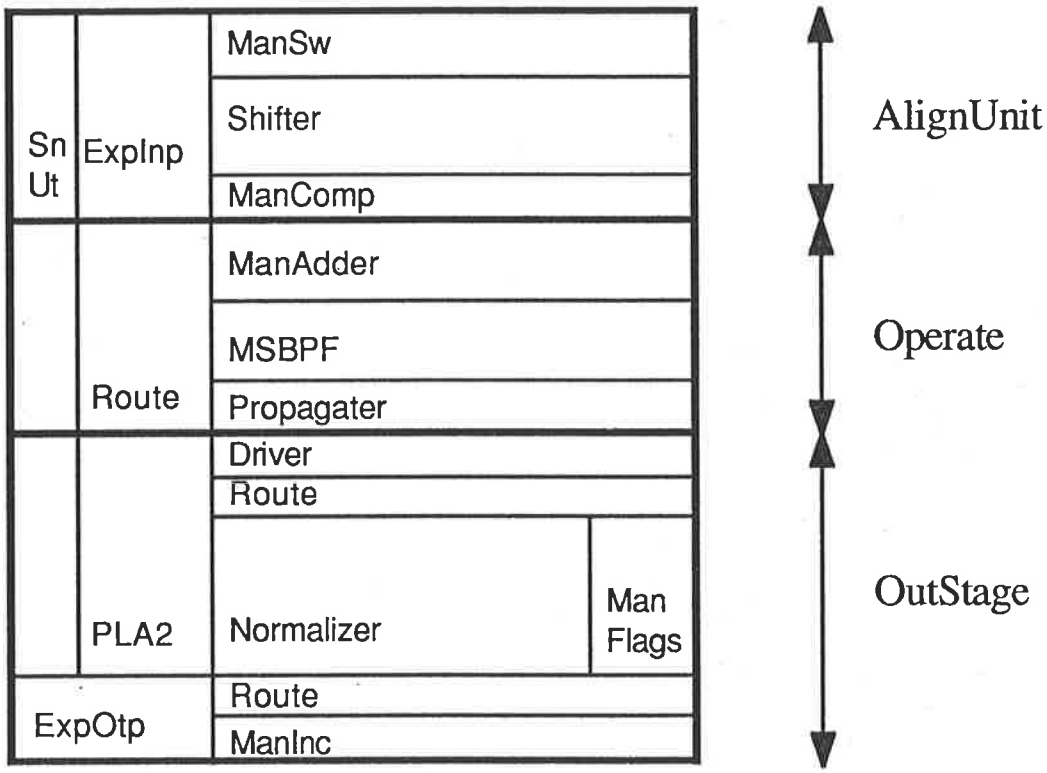


Figure 7.10: Floating Point Adder: floorplan at the second level of partitioning.

It can be seen from the diagram of Figure 7.10 that the space below the Sign Unit is rather badly utilized. This space may be used, for example, to place a control circuit once the FPA is incorporated into the Mathematical Processor system in the way shown in Figure 6.16.

The wiring strategy adopted in the implementation of the FPA system is as follows:

1. The second layer metal is used for long vertical interconnections, eg connections between the modules Mantissa Adder, MSBPF and Propagater;

2. The first layer metal is used for running the power rails and long horizontal interconnections inside functional modules, eg control lines;

3. The polysilicon layer is used for providing vertical interconnect within the functional modules as well as for providing short vertical interconnections between such modules and

4. The use of diffusion layer interconnect is avoided unless it results in a substantial area reduction or interconnect simplification within a functional module.

With the floorplan of the FPA system exposed, down to the second level of hierarchical decomposition, it is now possible to discuss in detail the composition and physical implementations of the various modules. The sequence with which the modules are further exposed approximately follows the path of the data signals within the FPA system; first the components of the Align Unit are discussed, followed by a description of the composition of the module Operate and finally by a description of the modules within the Output Stage.

7.7.1 Sign Unit (SignUnit)

In order to determine whether the two mantissas are to be added together or subtracted from each other it is necessary to examine the opcode and the sign bits of the two operands. All possible combinations of these three binary variables and the resulting effective operations are presented in Table 7.1 with the following notation used: $OP = 0$ for addition and $OP = 1$ for subtraction, $ManOvf = 1$ if the result of mantissa operation overflowed and $ManOvf = 0$ if the result of mantissa operation did not overflow, $EOP = 0$ if the effective mantissa operation is an addition and $EOP = 1$ if the effective mantissa operation is a subtraction. The carry out signal C_{out}, from the most significant bit position of the Mantissa Adder, indicates mantissa addition overflow ($C_{out} = 1$) or lack of it ($C_{out} = 0$) when the two mantissas are being added together. When mantissa subtraction is performed $C_{out} = 1$ indicates a positive and $C_{out} = 0$ indicates a negative mantissa difference. The logical expressions for the various control signals generated by the Sign Unit are derived from the data presented in Table 7.1. The logical expression for the *effective operation* control signal EOP reads:

$$EOP = SA \oplus SB \oplus OP, \tag{7.8}$$

and the control signals *one's complement mantissa A* (C_{manA}) and *one's complement mantissa B* (C_{manB}) are defined by the following logic expressions:

$$C_{manA} = SA \cdot \overline{(SB \oplus OP)},$$

SA	SB	OP	Complement	Effective Operation	C_{out}	SR	$ManOvf$
0	0	0		$manA + manB$	0	0	0
					1	0	1
0	1	0	$manB$	$manA - manB$	0	1	0
					1	0	0
1	0	0	$manA$	$manB - manA$	0	1	0
					1	0	0
1	1	0	-	$manA + manB$	0	1	0
					1	1	1
0	0	1	$manB$	$manA - manB$	0	1	0
					1	0	0
0	1	1	-	$manA + manB$	0	0	0
					1	0	1
1	0	1	-	$manB + manA$	0	1	0
					1	1	1
1	1	1	$manA$	$manB - manA$	0	1	0
					1	0	0

Table 7.1: Effective mantissa operation as a function of SA, SB and OP, and the interpretation of the carry signal C_{out}.

$$C_{manB} = \overline{SA} \cdot (SB \oplus OP). \tag{7.9}$$

The logic expressions given in (7.9) are valid if the operation of complementing is carried out prior to the alignment operation. Since, in this implementation the required mantissa is one's complemented at the output of the Alignment Shifter the logic expressions (7.9) must be modified to take into account the information regarding which mantissa is unnormalized. The control signal $C_{o,ex}$, generated by the Exponent Input Stage, $= 0$ if mantissa B must be unnormalized and $C_{o,ex} = 1$ if mantissa A must be unnormalized. Two new complement control signals are defined: *one's complement input X* (CX) and *one's complement input Y* (CY) by the following logic expressions:

$$\begin{aligned} CX &= C_{manB} \cdot C_{o,ex} + C_{manA} \cdot \overline{C_{o,ex}}, \\ CY &= C_{manB} \cdot \overline{C_{o,ex}} + C_{manA} \cdot C_{o,ex}. \end{aligned} \tag{7.10}$$

The logic specification of the Sign Unit is given in Figure 7.11(a) and its leaf module environment specification in Figure 7.11(b).

7.7.2 Exponent Input Stage (ExpInp)

The Exponent Input Stage performs a number of operations on the exponents of the operands. It is further decomposed into three modules as shown in Figure 7.12(a):

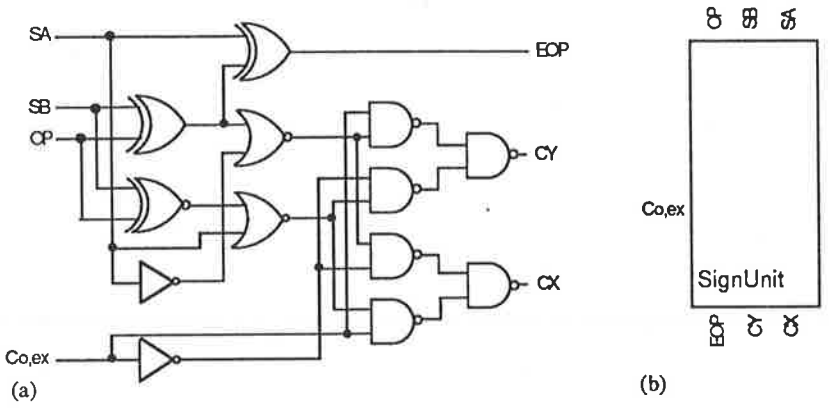


Figure 7.11: Sign Unit: (a) logic implementation, (b) leaf module environment specification.

1. *Implicit Bits-Invalid Operand Flags Generator (HidInv)*: generates the implicit mantissa bits *hidA* and *hidB*, checks for invalid operands and generates appropriate status flags *invA* and *invB*.

2. *Exponent Subtracter (ExpDiff)*: calculates the positive and the negative exponent differences and generates the control signal $C_{o,ex}$.

3. *Exponent Multiplexer (ExpMux)*: selects the larger operand exponent as the intermediate value of the resultant exponent (*expr*) and generates the alignment shift distance control (*shift*).

The three modules are pitch matched and are connected together by abutment as shown in the floorplan of Figure 7.12(b). Their logic implementation is described in the following three sections.

7.7.2.1 The Implicit Bits-Invalid Operands Flag Generator (HidInv)

One of the functions performed by the Exponent Input Stage is the detection of invalid operands, indicated by the generation of one or both status flags, *invA* and *invB*. In addition, this module examines the exponents of the operands and generates the implicit bits which are attached to the fractional parts of the operands to form their mantissas. These functions are performed by the functional module HidInv.

The possible invalid operands include infinity and the quantity NAN (see Table 6.1). The invalid operand flags are generated by testing the exponents of the operands for the reserved value 255 (the exponent value of an operand with the Standard's representation of infinity or a quantity NAN.) The required logic expressions for the two flags are given

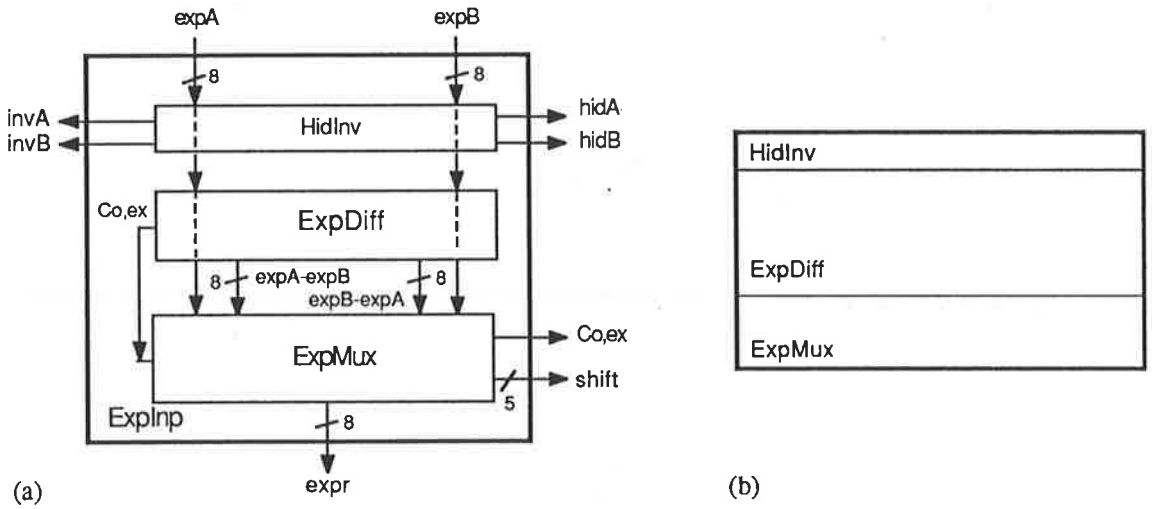


Figure 7.12: ExpInp: (a) block diagram, (b) floorplan.

by:

$$\begin{aligned} invA &= expA_8 \cdot expA_7 \cdots expA_2 \cdot expA_1, \\ invB &= expB_8 \cdot expB_7 \cdots expB_2 \cdot expB_1. \end{aligned} \tag{7.11}$$

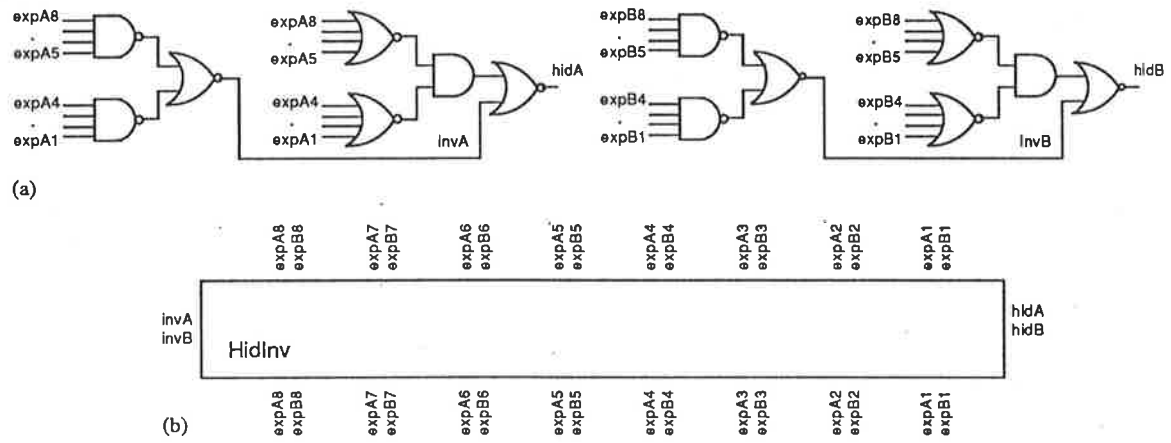


Figure 7.13: HidInv: (a) logic implementation, (b) leaf module environment specification.

The implicit bits of the operands are generated by simply testing the two exponents for the reserved exponent value of zero. This value of the exponent corresponds to a denormalized number or to the Standard's representation of zero. In both cases the implicit bit is zero.

As specified in Section 6.4.4, any operand which is denormalized is treated as zero and any operand which is NAN is treated as infinity by the FPA system. Both zero and infinity are characterized by the mantissa field of zero, as discussed in Chapter 6. The implicit bits $hidA$ and $hidB$ are also used as control signals by the Mantissa Switch to generate mantissa zero input corresponding to an operand being either denormalized (which is converted to zero) or a NAN (which is converted to infinity). The required logic expressions for the implicit bit $hidA$ of operand A, and for the implicit bit $hidB$ of operand B are given by:

$$\begin{aligned} hidA &= \overline{\overline{expA_8} \cdot \overline{expA_7} \cdots \overline{expA_2} \cdot \overline{expA_1} + invA}, \\ hidB &= \overline{\overline{expB_8} \cdot \overline{expB_7} \cdots \overline{expB_2} \cdot \overline{expB_1} + invB}. \end{aligned} \tag{7.12}$$

Figure 7.13(a) depicts the logic implementation of expressions (7.11) and (7.12), and Figure 7.13(b) shows the environment specification of the complete function module HidInv.

7.7.2.2 Exponent Subtracter (ExpDiff)

The alignment shift distance (*shift*), by which the mantissa of the smaller operand must be unnormalized, is given by the positive difference of the two exponents. This quantity is calculated by the Exponent Subtracter. One possible implementation of this module consists of a single 8-bit adder that always performs $expA - expB$ in two's complement notation, for example. The result is a two's complement number which must be two's complemented if the difference is negative, before it can be applied to the Alignment Shifter as the shift control. Thus, in addition to the adder a two's complementing circuit is necessary.

A faster method of determining the shift distance is to evaluate the possible exponent differences $expA - expB$ and $expB - expA$ in parallel followed by a selection of the positive one. This approach was chosen since it is about twice as fast as the first and the difference in silicon area required by the two approaches is minimal.

The two functions are combined into one module resulting in some of the hardware being shared. The logic implementation of the 1-bit functional module ExpDiffbit is shown in Figure 7.14 with the section shown in Figure 7.14(a) being shared by the two 1-bit logic sections shown in parts (b) and (c) of the figure. The functional module ExpDiffbit operates on the single bits $expA_i$ and $expB_i$ of the 8-bit exponent words $expA$ and $expB$ and generates the corresponding single bits of the two possible exponent differences. Eight such modules are replicated to form the 8-bit Exponent Subtracter, as shown in Figure 7.14(c). The 1-bit adders are optimized for carry propagation by

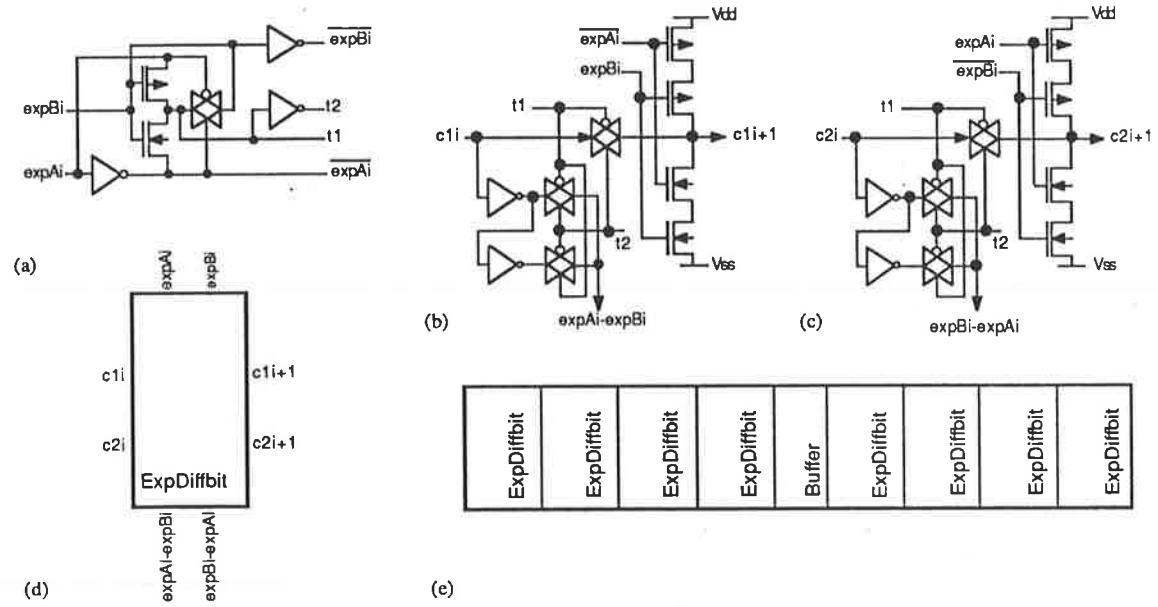


Figure 7.14: ExpDiff: (a) shared logic, (b) logic implementation of the one bit adder executing $expA_i - expB_i$, (c) logic implementation of the one bit adder executing $expB_i - expA_i$, (d) leaf module environment specification, (e) floorplan.

using only one transmission gate per stage. This is the well known and widely accepted Manchester carry chain technique.

In the complete 8-bit module the critical path includes eight series transmission gates which results in an unacceptable performance since the time delay of such a chain is proportional to the square of the number of transmission gates connected in series. Therefore, a buffer module is included after the first four adder stages as shown in the floorplan of Figure 7.14(e) to improve the propagation of the two carry signals c_1 and c_2.

7.7.2.3 Exponent Multiplexer (ExpMux)

The function of this module is to select the positive exponent difference or to generate a constant as the shift distance control (*shift*) for the Alignment Shifter. Two cases need to be considered.

- Case 1: $shift = |expA - expB| > 26$, in which case one of the operands is so small that any greater amount of shift is meaningless (only the "sticky bit" remains) and the shift amount control signal (*shift*) is set to its maximum value, which in this case is 31.

- Case 2: $0 \leq shift = |expA - expB| \leq 26$ and in this case one of the mantissas is shifted right by the number of places given by the positive difference of the exponents.

The alignment shift distance control (given by the positive exponent difference) is determined from the following logic expression:

$$\begin{aligned} shift_i &= \left[(expB_i - expA_i) \cdot C_{o,ex} + (expA_i - expB_i)\overline{C_{o,ex}}\right] \cdot \overline{OR_3} \\ &+ OR_3, \ i = 1,2\ldots,5, \end{aligned} \tag{7.13}$$

where the control signal OR_3 is the logical OR function of the three most significant bits of the positive exponent difference. The state of OR_3 determines whether the exponent difference is greater than or less than 31. In expression (7.13) the term $(expB_i - expA_i)$ denotes bit i of the exponent difference obtained by subtracting $expA$ from $expB$ and the term $(expA_i - expB_i)$ denotes bit i of the exponent difference obtained by subtracting $expB$ from $expA$.

The intermediate value of the resultant exponent (the larger operand exponent) is given by the following logic expression:

$$expr_i = expB_i \cdot C_{o,ex} + expA_i \cdot \overline{C_{o,ex}}, \ i = 1,2\ldots,8. \tag{7.14}$$

The functions specified by expressions (7.13) and (7.14) are combined into one functional module ExpMuxbit shown in Figure 7.15(a) and (b).

The logic necessary to generate the control signal OR_3 causes irregularity in the structure of the Exponent Multiplexer because it involves three out of the eight bits of the positive exponent difference. Its logic implementation is shown in Figure 7.15(c) and its environment specification in Figure 7.15(d). As a result the Exponent Multiplexer is composed of the functional module ExpMux3 and five instances of the functional module ExpMuxbit, as shown by the floorplan in Figure 7.16

7.7.3 Mantissa Switch (ManSw)

The Mantissa Switch module is composed of twenty seven instances of a 1-bit functional module ManSwbit and a signal driver module (Driver) for driving the control signals $C_{o,ex}$, $hidA$ and $hidB$ along the corresponding control signal lines spanning the entire length of the Mantissa Switch module, as shown in Figure 7.17(a).

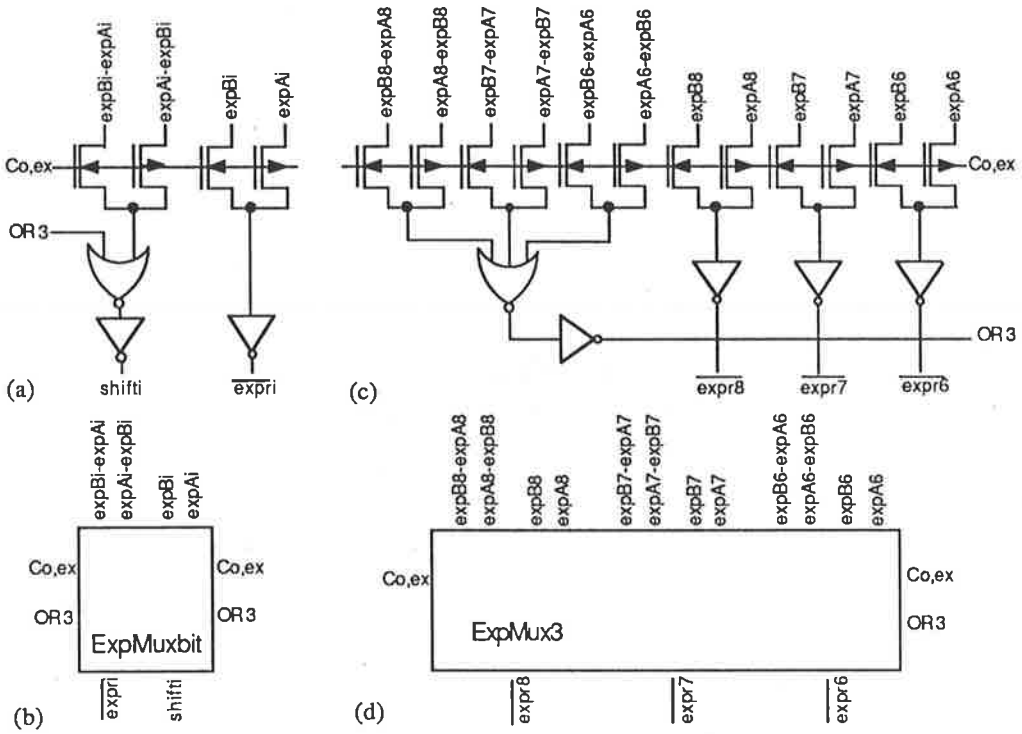


Figure 7.15: ExpMux: (a) ExpMuxbit functional module logic implementation, (b) ExpMuxbit leaf module environment specification, (c) ExpMux3 functional module logic implementation, (d) ExpMux3 leaf module environment specification.

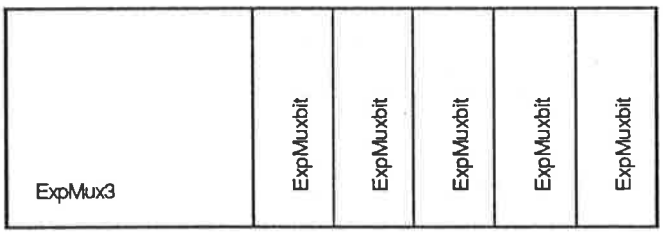


Figure 7.16: ExpMux: floorplan.

The two 1-bit data inputs include $manA_i$ and $manB_i$, and the two 1-bit data outputs $shifter_i$ and $bypass_i$ are determined according to the following logic expressions:

$$\begin{aligned} shifter_i &= (manA_i \cdot C_{o,ex} + manB_i \cdot \overline{C_{o,ex}}) \cdot hidA, \\ bypass_i &= (manB_i \cdot C_{o,ex} + manA_i \cdot \overline{C_{o,ex}}) \cdot hidB, \quad i = 1, \ldots, 27, \end{aligned} \tag{7.15}$$

where $shifter_i$ specifies a single bit of the operand mantissa to be unnormalized by passing it through the Alignment Shifter and $bypass_i$ specifies a single bit of the normalized operand mantissa which is simply routed through the Alignment Shifter and is directly applied to the Mantissa Complementer. In expression (7.15) $manA_i = fracA_i$ and $manB_i = fracB_i$ for $i = 4, \ldots 26$, and $manA_i = manB_i = 0$ for $i = 1, 2$ and 3, and

$manA_{27} = manB_{27} = 1$.

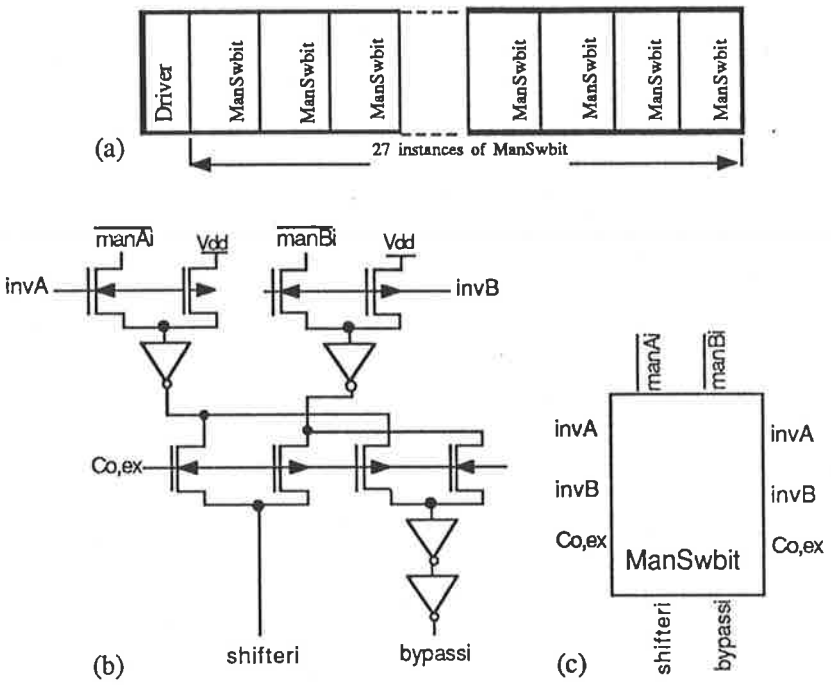


Figure 7.17: ManSw: (a) floorplan, (b) logic implementation of the single bit functional module ManSwbit, (c) leaf module environment specification of the ManSwbit module.

The logic implementation of the 1-bit leaf module ManSwbit is given in Figure 7.17(b) and its leaf module environment specification is given in Figure 7.17(c). A full CMOS implementation of this functional module with transmission gates would require eight transmission gates and six control lines (or three control lines with three additional inverters to generate the complements of the control signals). A pseud-nMOS implementation of this circuit is used to reduce the number of transistors saving silicon space. Three common control signal lines are run horizontally across the functional module and form the only interconnection between the neighbouring modules.

7.7.4 Alignment Shifter (Shifter)

The functions of this module are to unnormalize (shift right) the mantissa of the smaller operand by the specified number of places, in order to align it with the mantissa of the larger operand, and to round the shifted mantissa to 27 bits according to the RN rounding mode algorithm. The 5-bit shift distance control signal (*shift*) generated by the Exponent Input Stage is directly applied to the Alignment Shifter without any further decoding necessary because of the particular shifting circuit used.

The floorplan of the Alignment Shifter is given in Figure 7.18(a). This module is composed of a Driver module (containing control line drivers), 26 instances of a 1-bit functional module Shifterbit and a functional module Stickybit implementing the "sticky bit" function.

The operation of mantissa unnormalization is performed by a tree shifter, a circuit that allows any amount of shift to be performed in $\log_2 n$ stages, where n is the number of bits of the operand. The first stage the the shifting part of the Alignment Shifter, shifts the operand 0 or 1 place to the right. The second stage performs a shift of 0 or 2 places to the right and so on in increasing powers of 2 up to stage $\log_2 n$, where a shift of 0 or $n/2$ places is performed.

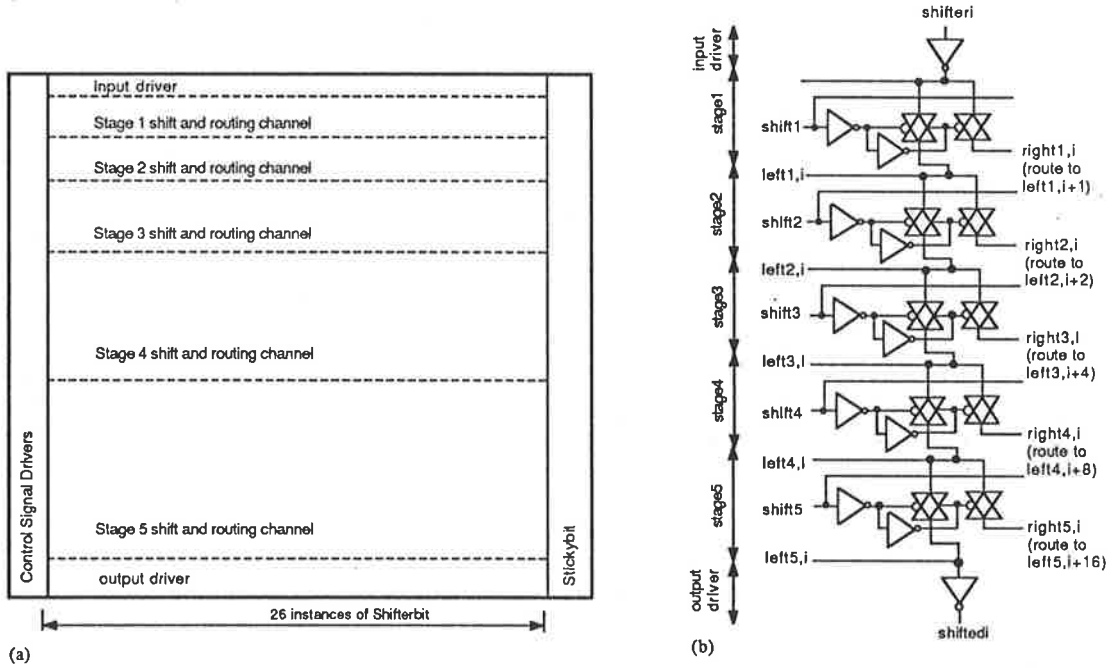


Figure 7.18: Shifter: (a) floorplan, (b) logic implementation of a bit slice section.

The leaf module is implemented with pass gate logic. Data signal buffering is applied at the input and the output of the Alignment Shifter. The critical data signal path includes the input and the output drivers, and five transmission gates. Circuit simulations showed that no improvement in circuit speed was possible by introducing additional buffering within the path to break up the chain of five transmission gates. The logic implementation of the 1-bit functional module Shifterbit is shown in Figure 7.18(b).

The implementation of the "sticky bit" (defined in Section 6.4.4) function in the FPA system requires a 29 input logic OR gate. This has been implemented as a pseudo-nMOS NOR gate followed by an isolation pass transistor and a sense amplifier with its threshold

level shifted towards the positive power supply voltage level [CSPL79], [WeEs85]. This circuit design technique is termed F^2L [CSPL79] and it provides improved delay for high fan-in circuits by trading noise margin for sensitivity. A detailed discussion of this circuit design technique can be found in the cited references.

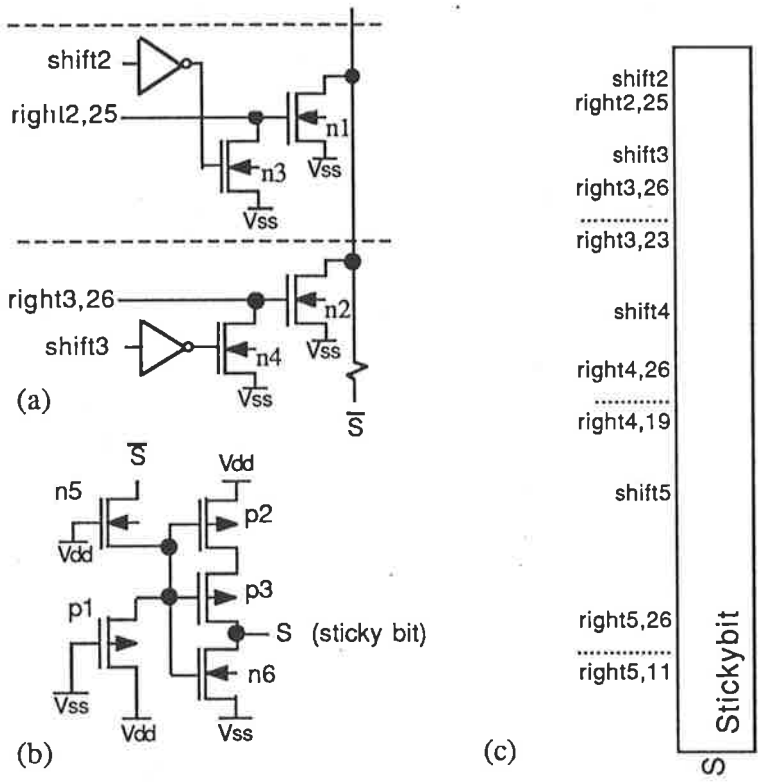


Figure 7.19: Stickybit: (a) logic implementation of a section of the complete module, (b) logic implementation of the output stage, (c) leaf module environment specification.

For reasons of limited space the logic implementation of only that part of this functional module which is connected to the second output of the second stage and the first output of the third stage of the rightmost Shifterbit module is shown in Figure 7.19(a). The logic implementation of the output stage consisting of the pull-up transistor **p1**, the isolation transistor **n5** and the output sense amplifier composed of transistors **p2**, **p3** and **n6**, is shown in Figure 7.19(b). The sizes of the various transistors were optimized using the results of circuit simulations carried out using the modified FACTS circuit simulation program.

7.7.5 Mantissa Complementer (ManComp)

At the output of the Alignment Shifter one of the mantissas is one's complemented if the effective mantissa operation is a subtraction. The two control signals CX and CY

generated by the functional module Sign Unit are used to one's complement the required mantissa.

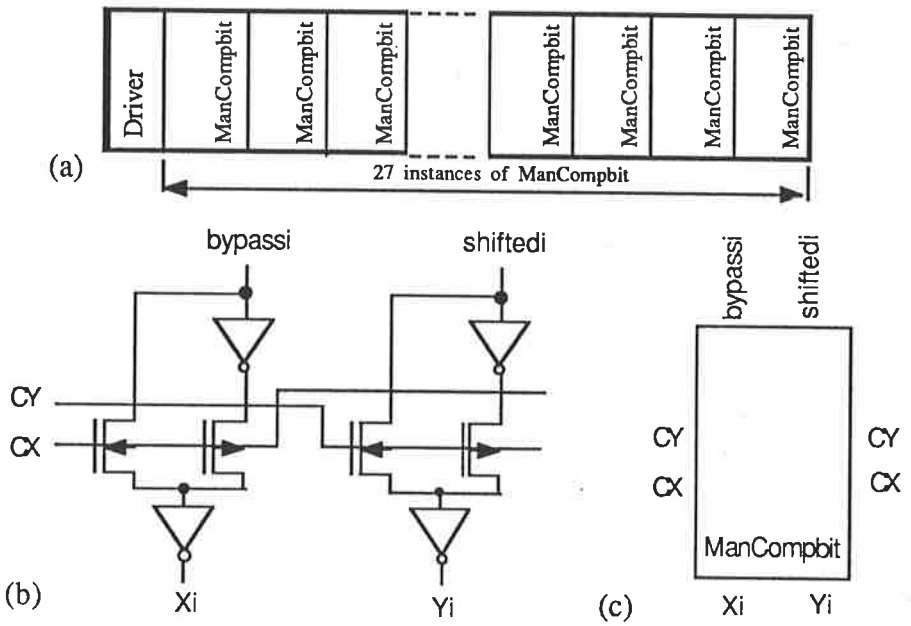


Figure 7.20: Mantissa Complementer: (a) logic implementation for the bit slice module, (b) leaf module environment specification, (c) floorplan.

The Mantissa Complementer is composed of a functional module Driver, containing signal drivers for the control signal lines which span the complete length of this module, and 27 instances of a 1-bit functional module ManCompbit, as shown in the floorplan of Figure 7.20(a). The logic implementation of the 1-bit leaf module is given in Figure 7.20(b) and its environment specification is shown in Figure 7.20(c). It is implemented with pass transistor logic with signal restoring inverters at its outputs.

7.7.6 Mantissa Adder (ManAdder)

The physical implementation of the Mantissa Adder module is critical since this is one of the modules which has the largest delay and ultimately determines the overall delay of the FPA system.

In the most simple implementation, an n-bit parallel adder may be decomposed into n 1-bit adders connected together. The worst case delay is then determined by the time the carry bit takes to ripple across the complete adder from the least significant bit position to the most significant bit position. Thus, the overall delay may be decreased by making the delay of the carry bit across the 1-bit adder cell as small as possible.

However, the possible speed improvement which can be achieved in this way is rather limited and ripple carry adders, although attractive from the power dissipation and silicon area point of view, are commonly used for operands of small sizes, eg < 12 bits is recommended in [Hwan79] and [Kuck81].

The techniques for increasing the speed of addition of two binary numbers have been studied by many authors, from the point of view of silicon area requirements, the power dissipation and the maximum delay. A number of techniques of various complexity have been developed to speed up the propagation of the carry bit across a carry propagate adder [Hwan79], [Kuck81], [WaFl82], [WeEs85]. These include: the carry look ahead adder, the carry select adder, and the conditional sum adder.

The design of the Mantissa Adder is another place in the design of the FPA system where a decision, involving a compromise, has to be made. Its physical implementation is, however, influenced to a large degree by other modules in the FPA system. In particular, there are three additional modules of similar complexity (and thus having similar silicon area requirements) and circuit operation characteristics (and thus having similar maximum delays) as the Mantissa Adder. These are the MSBPF, the Propagater and the ManInc. In all of these modules there is a signal, analogous to the carry bit in the Mantissa Adder, that can potentially propagate across the complete lengths of the respective modules. It will be argued, in the following, that the maximum delay of all such signals are of equal importance in determining the performance of the FPA system. This in effect limits the amount of additional hardware that can be used to increase the propagation speed of the carry signal across the Mantissa Adder, since this hardware must also be included in the MSBPF, the Propagater and the ManInc, if a reduction of the maximum delay of the FPA system is to be achieved.

In order to understand better the operation of the FPA system and the way the various modules contribute to the overall delay, it is helpful to examine the data flow graph for a part of the FPA shown in Figure 7.21. Of particular interest are the compositions of the modules Operate and Output Stage, since the implementation of the module Align Unit is the same in the FPA and the CFPA systems (CFPA denotes here, as defined in Section 7.3.1, the conventional hardware realization of floating point addition, as shown in Figure 7.2). In the figure, the broken lines are used to indicate data paths that result from a section of circuitry being shared by the two modules involved, and thus the delays of such paths can be neglected.

The Mantissa Adder and the MSBPF commence generating their respective outputs simultaneously, and as soon as the output of the Mantissa Complementer becomes stable. Since the Normalizer derives its inputs from both the Mantissa Adder and

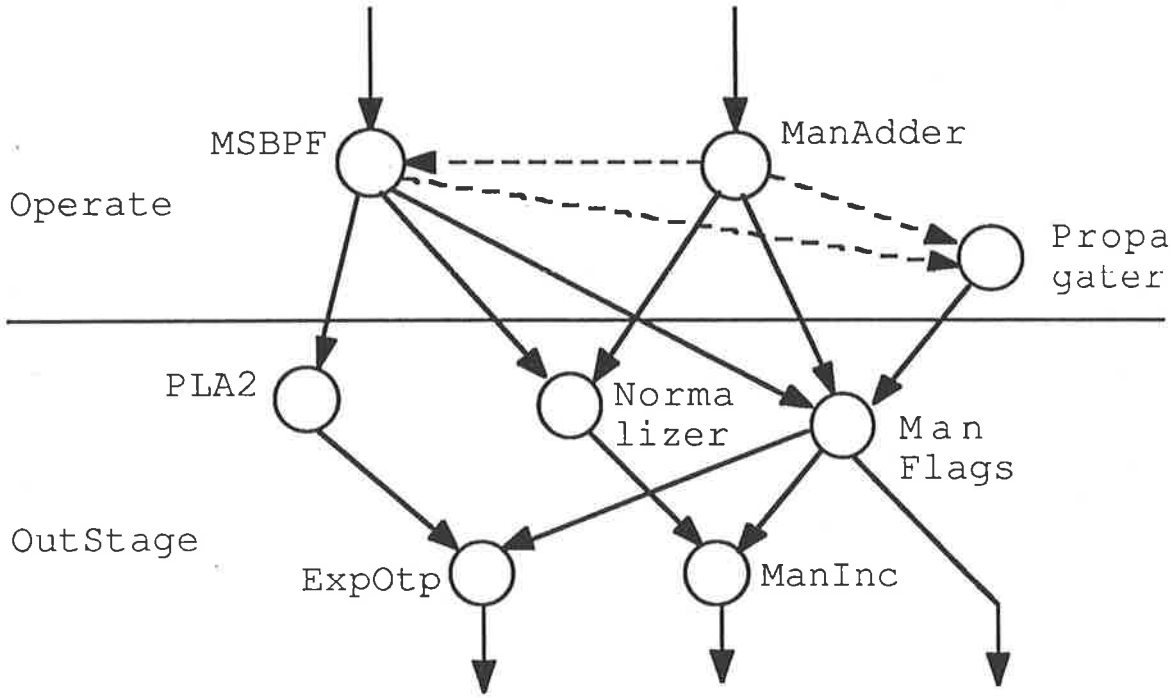


Figure 7.21: Floating Point Adder: a partial data flow graph.

the MSBPF, the time at which the outputs of the Normalizer commence to stabilize depends on the time at which the slowest output from either the Mantissa Adder or the MSBPF becomes stable. Thus, the maximum delays of these two modules are of equal importance in determining the maximum delay of the FPA system, ie reducing the maximum delay of only one of the two modules does not improve the performance of the FPA system.

The maximum delays of the Normalizer and the ManFlags also determine to the same extent the maximum delay of the FPA system. This is because the ManInc derives its inputs from these two modules and thus, the time at which its output commences to stabilize depends equally on the maximum delays of the Normalizer and the ManFlags.

The ManFlags in turn derives its inputs from the Mantissa Adder, the MSBPF and the Propagater, making the maximum delay of the Propagater of equal importance to the maximum delays of the Mantissa Adder and the MSBPF.

It follows, from the above argument, that for optimum performance the technique used to increase the propagation speed of the carry bit across the Mantissa Adder must also be used to increase the propagation speed of the *Fnd* signal (to be defined in Section 7.7.7) across the MSBPF and to increase the propagation speed of the *Pr* signal (to be defined in Section 7.7.8) across the Propagater. If the same technique is also applied in the ManInc, to increase the propagation speed of the carry bit, then the extra hardware

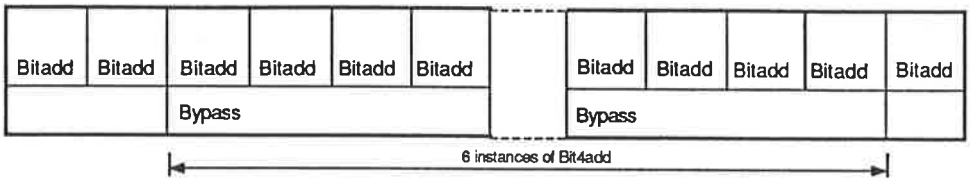


Figure 7.22: Mantissa Adder: floorplan.

designed to improve the speed of the Mantissa Adder must be multiplied by a factor of four. This, in effect, limits the amount of silicon area (and the power dissipation) that can be occupied by the extra circuitry used for this purpose.

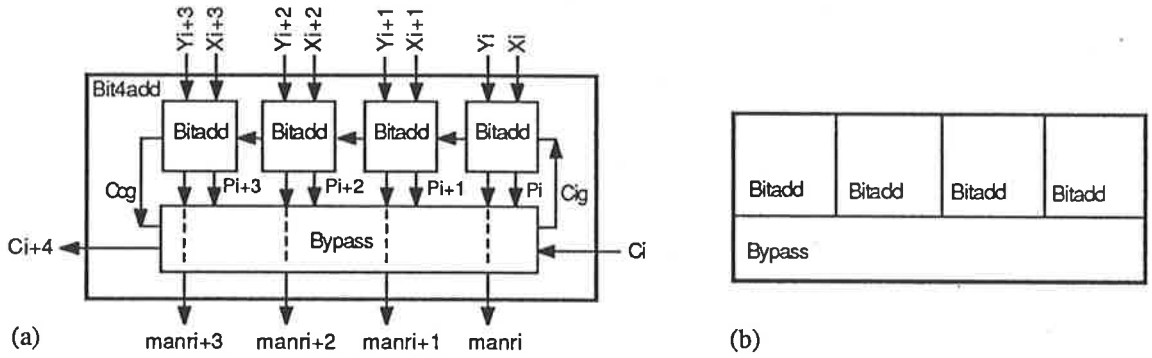


Figure 7.23: Bit4add: (a) block diagram, (b) floorplan.

One of the more popular techniques used to speed up the carry bit propagation is the pseudo carry look-ahead circuit [Hwan79], [Kuck81], [WaFl82] and [WeEs85]. Rather than implementing the full carry look-ahead module the adder is subdivided into groups of typically four 1-bit adders and the carry look-ahead technique is then applied within such groups of adders. In this way the carry bit has to ripple through only a limited number of 1-bit adder stages. The logic expression for the pseudo carry look ahead circuit for a group of four bits is given as:

$$C_{i+4} = G_{i+3} + P_{i+3} \cdot G_{i+2} + P_{i+3} \cdot P_{i+2} \cdot G_{i+1} + P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot G_i + P_{i+3} \cdot P_{i+2} \cdot P_{i+1} \cdot P_i \cdot C_i. \tag{7.16}$$

where C_i is the carry into the 4-bit adder group, C_{i+4} is the carry out of the 4-bit adder group. The carry propagate signal P_i and the carry generate signal G_i of the ith 1-bit adder stage are defined by the following Boolean expressions:

$$P_i = X_i \oplus Y_i, \quad G_i = X_i \cdot Y_i, \tag{7.17}$$

where X_i and Y_i are the ith bits of the aligned mantissa X and Y, respectively.

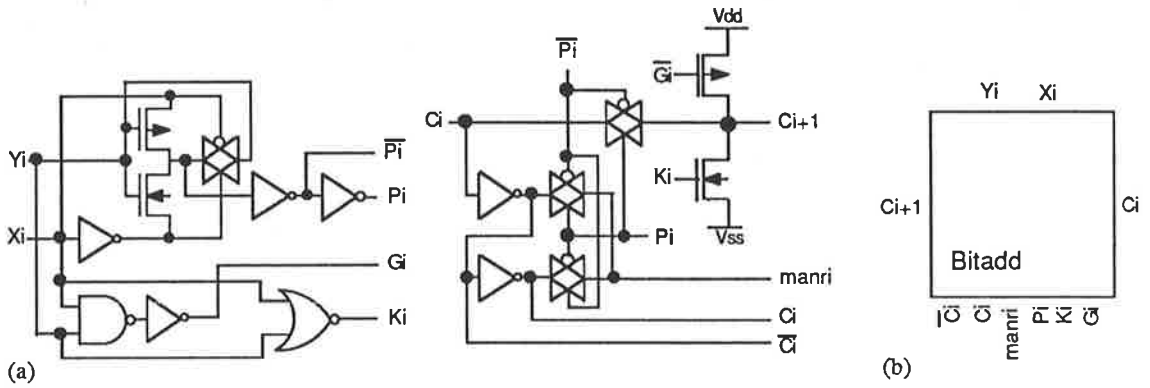


Figure 7.24: Bitadd: (a) logic implementation, (b) leaf module environment specification

Such a circuit was implemented and it was found that, although the delay improvement was substantial ($\approx$ 50%) over the delay of the simple ripple carry adder, the extra silicon area necessary for the hardware was approximately 100%, which was considered unacceptable. However, by implementing the carry bypass method (also known as the Manchester adder method) [Hwan79], [Kuck81], [WaFl82] and [WeEs85], a speed increase comparable to that obtained with the pseudo carry look ahead technique was achieved, but the silicon area penalty was only $\approx$ 25%.

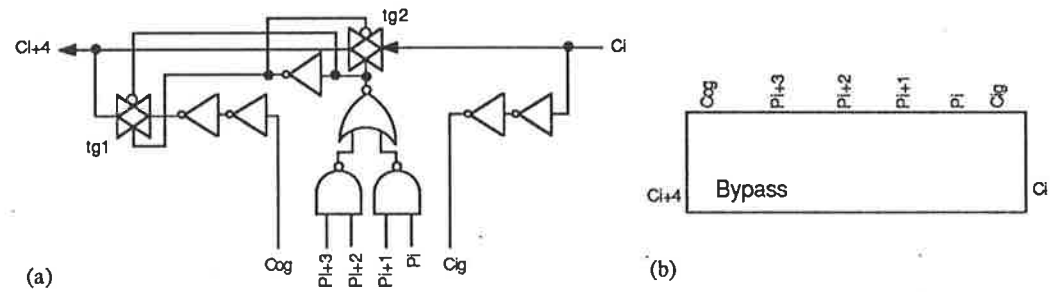


Figure 7.25: Bypass: (a) logic implementation, (b) leaf module environment specification.

The 27-bit Mantissa Adder module employs the carry bypass technique. The floorplan of this module is shown in Figure 7.22. A block diagram of the 4-bit adder group module (Bit4add) is shown in Figure 7.23(a) and the corresponding floorplan in Figure 7.23(b). The two signals C_{ig} and C_{og} are defined in Figure 7.25(a). The logic implementation of the 1-bit adder module (Bitadd) is given in Figure 7.24(a) and its leaf module environment specification is given in Figure 7.24(b). The logic implementation of the carry bypass module (Bypass), is shown in Figure 7.25(a) with the leaf module environment specification in Figure 7.25(b).

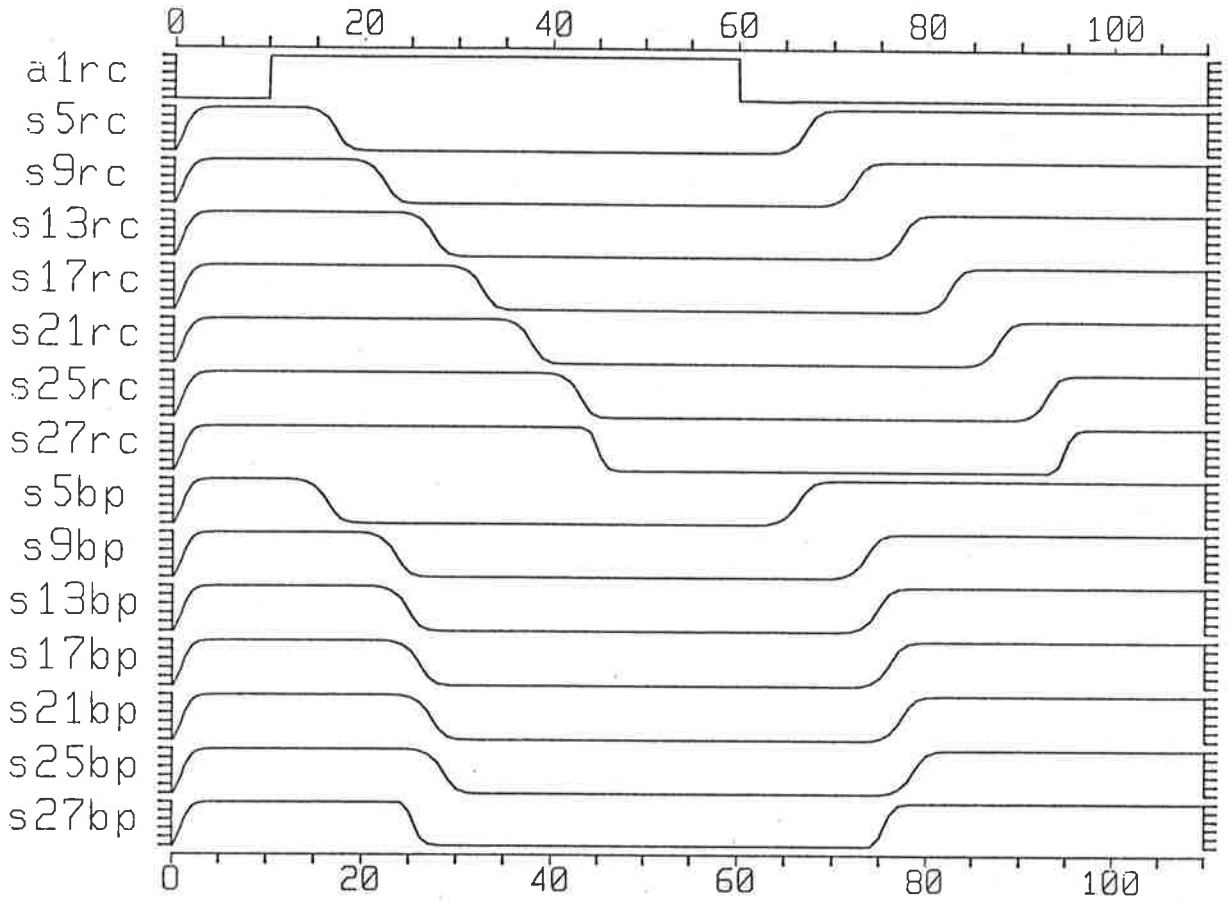


Figure 7.26: Carry propagate adder performance comparison (27 bits).

Figure7.26 illustrates the simulation results of the worst case delay of a 27-bit ripple carry adder implemented using the Bitadd modules (buffered every four stages) and the 27-bit Mantissa Adder, implemented as described above. The sum output waveforms in bit positions 5, 9, 13, 17, 21 and 25 are only included for the two adders. The extension **rc** indicates the ripple carry adder and the extension **bp** indicates the Mantissa Adder. It can be seen from the figure that the slowest output of the Mantissa Adder (waveform s25bp) becomes stable approximately $17ns$ before the slowest output of the ripple carry adder becomes stable. This represents a speed improvement of approximately 49%.

The mask representation obtained by compacting the symbolic layout of the module Bit4add according to the $Orbit^+$ process design rules is given in Figure 7.27. The dimensions of the overall bounding box are 304λ by 340λ, with the bounding box of the Bypass module being 304λ by 66λ. Thus, the Bypass circuit provides approximately a 50% speed improvement over the ripple carry adder at the cost of a 25% increase in silicon area.

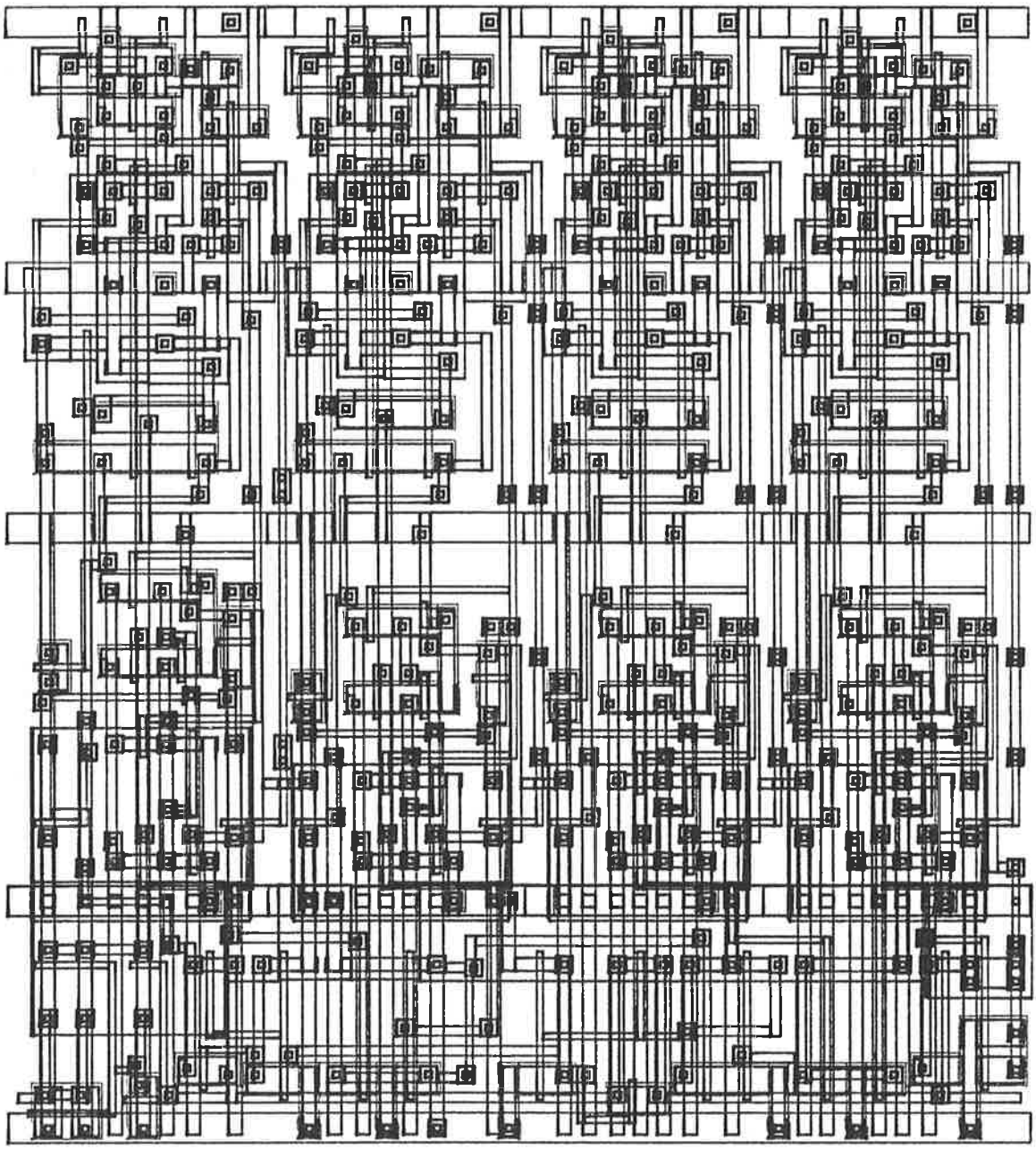


Figure 7.27: Bit4add: mask domain representation.

7.7.7 Most Significant Bit Position Finder (MSBPF)

The function of this module is to determine the position of the leading bit in the result of mantissa subtraction or addition. In the conventional approach, as discussed in Section 7.3.1, this is carried out by examining the output of the Mantissa Adder. More precisely, it is the output of the mantissa two's complementing module that is examined, see Figure 7.2.

One of the original features of the Floating Point Adder system is that a circuit has been developed (the MSBPF module) that determines the position of the leading bit in the mantissa sum or difference by examining the inputs (X and Y) rather than the output ($manr$) of the Mantissa Adder. This is the method of combining steps 4 and 6 of the conventional floating point addition algorithm (given in Section 7.3.1) into one step of the modified algorithm. This allows for the position of the leading bit to be determined in parallel with the mantissa addition or subtraction. The module generates a 28-bit signal *MSBP* which has only one of its bits asserted high and which indicates by how many places the output of the Mantissa Adder needs to be shifted in order to normalize it. (Only 26 out of the 28 bits are used since as discussed in Section 7.3.2.2 the position of the leading bit never occurs in the position of the second guard bit $i = 2$ or the "sticky bit" $i = 1$.) The function of the MSBPF and its implementation will now be explained in greater detail.

According to the discussion of Section 7.3.2.1 the result of mantissa addition may need at most a 1-bit shift to the right for its normalization. The detection of the leading bit reduces in this case to the detection of mantissa overflow, which is done by testing the state of the carry out from the most significant bit position of Mantissa Adder.

The task of detecting the position of the leading bit in a mantissa difference is much more difficult because there can be up to 24 leading zeros in the mantissa difference. The number of leading zeroes (and thus the position of the leading bit) is determined by successively examining groups of three adjacent pairs of bits of the two mantissas being subtracted, starting from the most significant bit position. In some cases, however, the result is accurate only to within one bit position. An examination of the carry signal resolves the possible uncertainty.

All possible combinations of operand bits for the three bit positions $i+2, i+1, i$ are presented in Table 7.2, with the position of the leading bit (*MSBP*) indicated in the final column of the table. The cases for which the leading bit does not correspond to the bit position i or $i+1$ are indicated by ND (non determinate). C_{i+1} stands for the carry bit into bit position $i+1$ and DC represent a *don't care* case.

The MSBPF is partitioned into 1-bit functional modules MSBPFbit. Some of the signals needed by the MSBPFbit functional module are also used inside the Bitadd functional module located in the same bit position. As a result some of the hardware is shared between the two. The common signals include: X_i, G_i, P_i, K_i and C_i. The way these are generated is shown in Figure 7.24(a).

The signal $\overline{P_{i+1}}$ is derived from the Bitadd module in bit position $i+1$. It indicates

(X_{i+2}, Y_{i+2})	(X_{i+1}, Y_{i+1})	(X_i, Y_i)	C_{i+1}	$MSBP$
$(0,1)$ or $(1,0)$	$(0,1)$ or $(1,0)$	(DC,DC)	DC	ND
	$(0,0)$	$(1,1)$	DC	ND
		$(0,1)$ or $(1,0)$ or $(0,0)$	0	$i+1$
			1	i
	$(1,1)$	$(1,1)$	DC	ND
		$(0,1)$ or $(1,0)$ or $(1,1)$	0	i
			1	$i+1$
$(0,0)$ or $(1,1)$	$(0,0)$	$(0,0)$	DC	ND
		$(0,1)$ or $(1,0)$ or $(1,1)$	0	i
			1	$i+1$
	$(1,1)$	$(0,0)$	DC	ND
		$(0,1)$ or $(1,0)$ or $(0,0)$	0	$i+1$
			1	i

Table 7.2: Position of the leading bit within a 3-bit section, for all combinations of component bits.

whether the position of the leading bit is likely to be found in bit position i or $i+1$.

In addition there are a number of control signals directly derived from the MSBPFbit modules in bit position $i+1$ or in bit position $i-1$. The control signal Gr_{i+1} generated in bit position $i+1$ and used by the MSBPFbit module in bit position i is given by:

$$Gr_{i+1} = \overline{X_{i+1}} \oplus \overline{P_{i+2}}. \tag{7.18}$$

This signal indicates whether the combination of operand bit pairs (X_i, Y_i) in bit position i must be one of $\{(0,0);(1,0);(0,1)\}$ $(Gr_{i+1} = 0)$ or one of $\{(1,1);(1,0);(0,0)\}$ $(Gr_{i+1} = 1)$ in order to generate a leading bit in position i or $i+1$.

The control signal Fnd_{i+1} indicates whether a leading bit has been found at a more significant bit position than the one being examined. This signal is analogous to the carry bit in a parallel adder circuit. However, it always propagates right across the circuit and it does so from the most significant bit position to the least significant bit position.

The logic expression for Fnd_i, which signal is generated in bit position i and which indicates if the leading bit has been found in bit position i or in a more significant bit position, is given by:

$$Fnd_i = Fnd_{i+1} \cdot PrF_i + \overline{PrF_i}. \tag{7.19}$$

The internally generated signal PrF_i indicates whether a conclusive decision can be made about the position of the leading bit on the basis of the signals examined in the current bit position. It is analogous to the carry propagate signal (P_i) in the Mantissa

Adder in that it indicates whether the control signal Fnd is propagated across the bit position i or not. It is given by:

$$PrF_i = P_{i+1} + \overline{P_{i+1}}(Gr_{i+1} \cdot X_i \cdot Y_i + \overline{Gr_{i+1}} \cdot \overline{X_i Y_i}) \tag{7.20}$$

In order to determine exactly the position of the leading bit it is also necessary to examine the value of the carry bit (C_i), as mentioned earlier in this section. A decision signal (D_{i+1}) is generated, when the carry bit is available, to finally determines the exact position of the leading bit. The logic expression of the decision signal is given by:

$$D_{i+1} = \overline{P_{i+1}} \cdot [Gr_{i+1} \cdot \overline{C_{i+1}} + (\overline{Gr_{i+1}} \cdot C_{i+1})]. \tag{7.21}$$

Having defined all the necessary signals it is now possible to give a a complete expression for $MSBP_i$, a signal indicating that the leading bit lies in bit position i, where $i = 1, \ldots 28$. The required logic expressions are given by:

$$MSBP_i = \overline{Fnd_{i+1}} \cdot \overline{PrF_i} \cdot \overline{D_{i+1}} + \overline{Fnd_{i+1}} \cdot PrF_i \cdot D_i, \quad i = 1, \ldots 27, \tag{7.22}$$

and

$$MSBP_{28} = C_{out} \cdot \overline{EOP}, \tag{7.23}$$

where $MSBP_{28} = 1$ indicates that the output of the Mantissa Adder must be shifted one place to the right to normalize it.

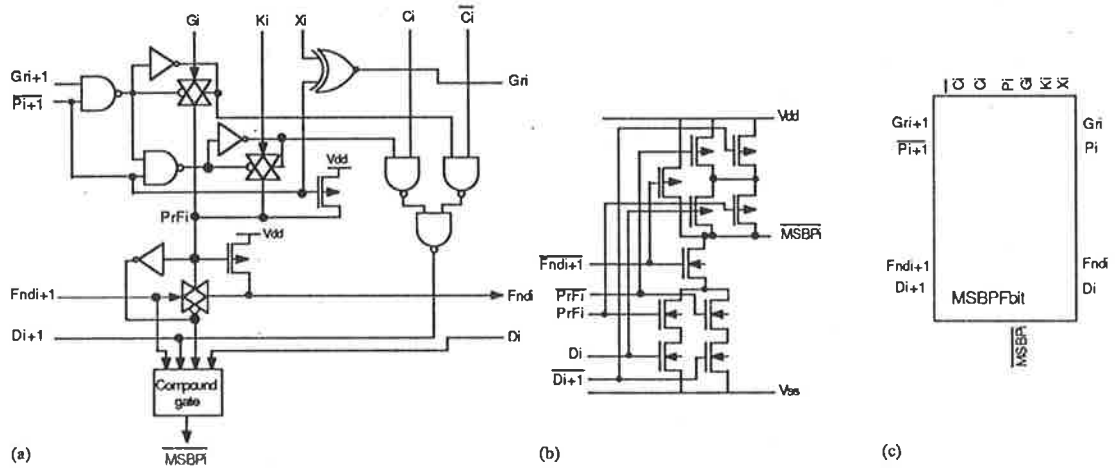


Figure 7.28: MSBPFbit: (a) logic implementation, (b) implementation of the compound gate, (c) leaf module environment specification.

Note that if the circuit fails to find a leading bit then $Fnd_1 = 0$ and $MSBP_i = 0$ for $i = 1, 2, \ldots 28$. This fact is used as described later to determine a possible mantissa

overflow condition or a mantissa zero result. The logic diagram of the part of the 1-bit slice of the MSBPF, MSBPFbit, which is required in addition to the Bitadd module to implement the function described by the logic expression (7.22) is depicted in Figure 7.28.

A similar circuit was independently proposed in [HKBB85]. In their implementation an *error* signal was generated. That error signal was then used (rather than the decision signal D_i) to resolve the uncertainty about the position of the leading bit.

In order to reduce the delay of propagation of the Fnd_i control signal, the Bypass module is used with the inputs P_i, P_{i+1}, P_{i+2} and P_{i+3} (Figure 7.25) replaced by PrF_i, PrF_{i+1}, PrF_{i+2} and P_{i+3}, respectively, and with the signals C_i and C_{i+4} replaced by Fnd_i and Fnd_{i+4}, respectively. In addition, because the control signal Fnd propagates in the opposite direction to the carry signal, the orientations of the Bypass modules incorporated into the MSBPF module are altered accordingly.

The floorplan of the MSBPF is similar to the floorplan of the Mantissa Adder (shown in Figure 7.22) with the Bitadd modules replaced by MSBPFbit modules. The two input NAND gate necessary to implement logic expression (7.23) is incorporated into the routing channel (Route) connecting the modules Driver and Normalizer (Figure 7.10).

7.7.8 Carry Propagate Signal Generator (Propagater)

Consider an example of two 8 bit mantissas being subtracted from each other as shown in Figure 7.29. The leading bit of a normalized mantissa, in this case, lies in bit position 8. The aligned mantissas X and Y (Y is being subtracted from X and thus has been two's complemented) have values 166×2^{-7} and 38.75×2^{-7}, respectively. Their positive difference is 127.75×2^{-7} which has the leading bit in bit position 7, thus requiring a one place normalization shift to the left. The normalized mantissa difference has the value 255.5×2^{-7} and according to the RN rounding mode algorithm, discussed in Section 7.7.11.1, it needs to be rounded. The rounding operation produces mantissa overflow, since its result is 256×2^{-7} whose leading bit is in bit position 9.

The function of the Propagater is to generate control signals that provide sufficient information to detect *in advance* a potential mantissa overflow that can either occur during the operations of mantissa rounding or two's complementing. The detection of this condition prior to performing these operations allows the intermediate value of the resultant exponent (*expr*) to be adjusted in parallel with these mantissa operations, rather than following them. This is an additional novel architectural feature in the

mantissa bit position		8	7	6	5	4	3	2	1	G1	G2	S
X		1.	0	1	0	0	1	1	0	0	0	0
Y		1.	1	0	1	1	0	0	1	1	1	0
manr	1	0.	1	1	1	1	1	1	1	1	1	0
manrn		1.	1	1	1	1	1	1	1	1	0	0
rounding	+									1		
	1	0.	0	0	0	0	0	0	0	0	0	0
fracR		.0	0	0	0	0	0	0	0	0	0	0

Figure 7.29: Example: mantissa overflow during rounding.

floating point adder architecure presented in this thesis, and it combines steps 8 and 10 of the conventional floating point addition algorithm presented in Section 7.3.1.

In order to anticipate a possible mantissa overflow as a result of mantissa rounding or two's complementing, it is necessary to examine the output of the Mantissa Adder to determine whether a carry signal that can originate in the bit position of the first guard bit (rounding) or the "sticky bit" (complementing) can propagate to the position of the most significant bit of a normalized mantissa.

The required function for the control signal Pr_{G_1}, indicating that mantissa rounding will overflow, is a logical AND of all the carry propagate signals P_i between the bit position for which $MSBP_i = 1$ (the position of the leading bit) and the bit position of the first guard bit ($i = 3$).

The required function for the control signal Pr_S, indicating that mantissa complementing will overflow, is a logical AND of all the carry propagate signals P_i between the bit position for which $MSBP_i = 1$ and the bit position of the "sticky bit" ($i = 1$).

The functional behaviour of the Propagater is realized by generating a test signal Pr_i in the bit position for which $MSBP_i = 1$ and determining whether this test signal propagates to bit positions given by $i = 3$ (signal $Pr_{G_1} = Pr_3$) and $i = 1$ (signal $Pr_S = Pr_1$). This test signal can be thought of as being a pseudo-carry signal propagating across the Propagater in parallel with the carry signal propagating across the Mantissa Adder, but in the opposite direction.

The Propagater derives its inputs from the Mantissa Adder (inputs P_i) and from the MSBPF (inputs $\overline{PrF_i}$ and Fnd_{i+1}). The required logic expression is given by:

$$Pr_i = \overline{PrF_i} \cdot \overline{Fnd_{i+1}} \cdot P_i + Pr_{i+1} \cdot P_i. \tag{7.24}$$

The first part of expression (7.24) generates the test signal Pr_i in the bit position for

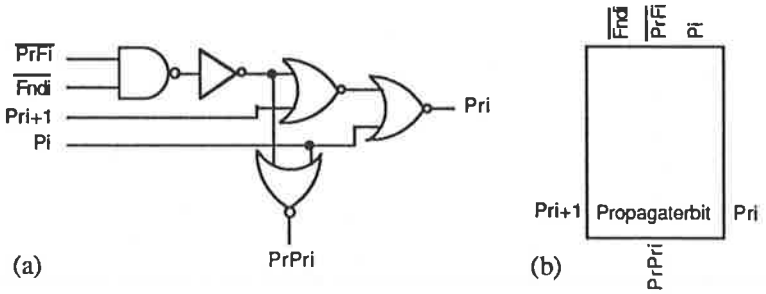


Figure 7.30: Propagaterbit: (a) logic implementation, (b) leaf module environment specification.

which $MSBP_i = 1$. The second part of the expression determines whether the bits of the operands in the ith bit position will result in a possible carry signal to propagate across it. The signal PrF_i is given by expression (7.20), the signal Fnd_i is given by expression (7.19) and the carry propagate signal (P_i) is given by expression (7.17). The logic implementation of a 1-bit functional module Propagaterbit is given in Figure 7.30.

The Bypass module is employed to increase the speed of propagation of the Pr signal and to equalize the propagation of this signal and the propagation delays of the carry bit and the Fnd_i signals. The inputs P_i, P_{i+1}, P_{i+2} and P_{i+3} to the Bypass circuit (Figure 7.25) do not exist as part of expression (7.24) and need to be explicitly generated. This is because expression (7.24) cannot be written as $Pr_i = (generate) + \overline{generate} \cdot Pr_{i+1}$, as is the case for the carry bit or the Fnd_i signal, ie the condition for generating the Pr_i signal within module i and the condition for propagating the Pr_i signal across module i cannot be expressed as complements of each other. The correct expression for the bypass signal within the Bypass circuit is:

$$PrPr_i = \overline{\overline{PrF_i} \cdot \overline{Fnd_{i+1}}} \cdot P_i. \tag{7.25}$$

The Bypass module, used in the Mantissa Adder and the MSBPF modules and shown in Figure 7.25, is again employed. Its use in this case increases the propagation speed of the Pr_i test signal across the Propagater. This time, the inputs P_i, P_{i+1}, P_{i+2} and P_{i+3} of the Bypass module are replaced with $PrPr_i, PrPr_{i+1}, PrPr_{i+2}$ and $PrPr_i + 3$, respectively, and the inputs C_i and C_{i+4} of the Bypass module are replaced by Pr_i and P_{i+4}, respectively.

The floorplan of the Propagater is similar to that of the Mantissa Adder, shown in Figure 7.22, with the Bitadd modules replaced by the Propagaterbit modules. In addition, because the control signal *Pr* propagates in the opposite direction to the carry signal, the orientations of the Bypass modules incorporated into the Propagater are altered accordingly.

7.7.9 Shift Distance Encoder (PLA2)

This module converts a "one-of-twenty five" control signal *MSBP* to its 5-bit binary representation. The input is derived from the MSBPF. The most economical, from the point of view of design effort, implementation of this module is to use the OR plane of a complete PLA (programmable logic array) circuit. This module has been realized with the pseudo-nMOS circuit design technique.

7.7.10 Mantissa Normalizer (Normalizer)

The function performed by this module is to shift the 26-bit input word *manr* (the second guard and the "sticky" bits do not participate in shifting) either one place to the right or anywhere between zero and twenty four places to the left. A floorplan of the modified barrel shifter implementing this function is shown in Figure 7.31(a), and the logic implementation of the 1-bit functional module is shown in Figure 7.31(b).

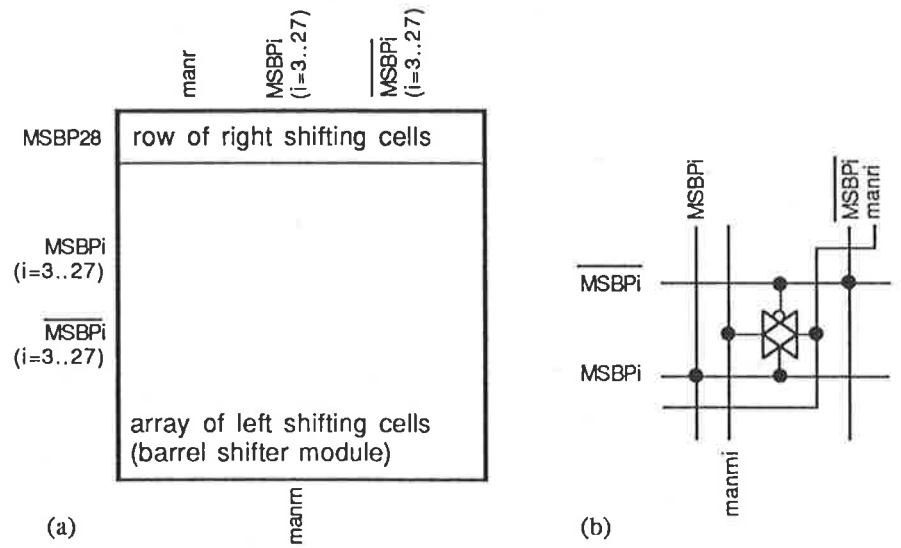


Figure 7.31: Normalizer: (a) floorplan, (b) leaf module logic implementation.

The first row of the leaf modules are interconnected to shift the input word one bit po-

sition to the right, depending on the shift control signal $MSBP_{28}$ (defined by expression (7.23)), and the remaining rows of leaf modules comprise a left shifting barrel shifter [MeCo80], [WeEs85]. The shift control bus ($MSBP$) and the data bus ($manr$) enter the Normalizer at the top edge of the module's bounding box. The output bus ($manr_n$) is made to run in the direction of the input data bus, rather than in the direction perpendicular to it, as in the conventional design. This modification allows an area efficient connection between the Normalizer and the ManInc, with the ManInc module, operating on the output $manr_n$, being placed directly below the Normalizer. The shift control bus is connected to a bus running in the perpendicular direction across the width of the Normalizer, Figure 7.31(b). This extra bus is included to change the direction of the control signals and make them available on the left boundary of the Normalizer, where it can be efficiently connected to the Shift Distance Encoder (PLA2) module, as shown in Figure 7.10.

The lengths of the data and the shift control buses results in a substantial capacitive loading on these signal lines. Signal driver module (Driver) is included, for the abovementioned signals, and placed immediately below the Propagater (Figure 7.10), to reduce the maximum delay of the Normalizer to an acceptable level.

7.7.11 Special Case Detector (ManFlags)

This module generates the following control and status signals:

1. *Rnd*: *mantissa round* control signal indicating that mantissa rounding is necessary. This is a single bit quantity that is added to the normalized resultant mantissa ($manr_n$) in the position of the least significant bit ($manr_{n,4}$).

2. *cpp*: *result complement* control signal indicating a negative mantissa difference which must be converted to the signed magnitude notation. This is a single bit control signal for the ManInc to perform this operation.

3. *ManOvf*: *mantissa overflow* control signal indicating that a mantissa addition has resulted in mantissa overflow or that mantissa rounding or complementing will result in mantissa overflow. This signal causes the intermediate value of the resultant exponent ($expr$) to be incremented by one.

In addition, the ManFlags module generates the inexact result flag (Inx) and the sign bit of the result (SR). The logic implementations of the Inx flag, the cpp control signal and the sign bit SR are considered first.

Exact result is obtained whenever both guard bits and the "sticky bit" are each equal to zero. Thus, the *Inx* flag is generated by examining the first ($G_1 = manr_{n,3}$) and the second ($G_2 = manr_{n,2}$) guard bits, and the "sticky bit" ($S = manr_{n,1}$):

$$Inx = G_1 + G_2 + S. \tag{7.26}$$

The IEEE Standard for Floating Point Arithmetic specifies the following algorithm for setting the sign of the result under the condition of computed zero: "when the sum of two operands with opposite signs (or the difference of two operands with like signs) is exactly zero, the sign of the result shall be positive". The complete expression for the sign bit of the result is generated using the data shown in Table 7.1:

$$SR = SA \cdot \overline{EOP} \cdot Fnd_1 + cpp. \tag{7.27}$$

The first part of expression (7.27) gives the value of SR if the effective mantissa operation is an addition, and the second part of the expression gives the value of SR if the effective mantissa operation is a subtraction. When the result of mantissa addition is zero then the signal $Fnd_1 = 0$, irrespective of the value of SA, thus setting $SR = 0$, as required. When the result of subtraction is zero then $cpp = 0$ in all cases, and again the final sign is positive, ie $SR = 0$.

The output of the Mantissa Adder needs to be two's complemented when the effective mantissa operation is a subtraction ($EOP = 1$) and the mantissa difference is negative ($C_{out} = 0$):

$$cpp = EOP \cdot \overline{C_{out}}. \tag{7.28}$$

The logic expressions of the remaining control signals are much more complex and are discussed in detail in the following subsections.

7.7.11.1 Rounding Conditions

In the conventional approach the resultant mantissa is examined, following its normalization (ie the output $manr_n$ of the Normalizer), to determine whether rounding is necessary. The performance of the FPA system is further enhanced in this work by modifying the conventional floating point hardware architecture such that the resultant mantissa *prior* to its normalization (ie the output *manr* of the Mantissa Adder), rather than $manr_n$, is examined to determine whether mantissa rounding is required.

This new approach allows for the generation of the control signal *Rnd* to proceed in parallel with the operation of mantissa normalization performed by the Normalizer. Provided the delays of the Normalizer and the ManFlags are made equal (and as small

Positive *manr*				Negative *manr*				Shift amount	Round cond.	Mantissa overflow
$manr_4$	G_1	G_2	S	$manr_4$	G_1	G_2	S	(left)	*Rnd*	*ManOvf*
0	0	1	1	1	1	0	1	0 or 1	1	0
0	1	0	1	1	0	1	1	1 or 0	1	0
0	1	1	0	1	0	1	0	0 or 1	1	0
0	1	1	1	1	0	0	1	0 or 1	1	0
1	0	1	1	0	1	0	1	1	1	0
1	1	0	0	0	1	0	0	0	1	0
1	1	0	1	0	0	1	1	0	1	0
1	1	1	0	0	0	1	0	0 or 1	1	1
1	1	1	1	0	0	0	1	0 or 1	1	1

Table 7.3: Combinations of the relevant mantissa bits and normalization shift distance (to the left) that can cause rounding condition ($Rnd = 1$) and/or mantissa overflow ($ManOvf = 1$) as a result of rounding.

as possible), mantissa rounding (performed by the ManInc) can commence as soon as the output of the Normalizer becomes stable, thus effectively eliminating the delay of the ManFlags module. In addition, the implementation of the FPA system is such that the rounding condition is determined prior to taking the two's complement of a negative output of the Mantissa Adder. The cost of the performance increase, achieved with these architectural modifications, is that the implementation of the ManFlags becomes more complex than the implementation of the corresponding functional module in the CFPA system.

In order to determine whether mantissa rounding is required by examining the output (*manr*) of the Mantissa Adder, it is necessary to know the amount of the normalization shift indicated by the signal *MSBP*. Because of this fact no additional performance increase can be achieved by using the bits of the operands X and Y, in preference to the corresponding bits of *manr*. Furthermore, the use of the relevant bits of the resultant mantissa (*manr*) rather than the corresponding bits of X and Y has the advantage of greatly simplifying the necessary logic.

The following Boolean expression can be derived which indicates that post-addition rounding is necessary, by considering columns 1 to 4 and 9 to 10 of Table 6.2:

$$\begin{aligned} Rnd_{add} &= \overline{EOP} \cdot [MSBP_{27} \cdot G_1 \cdot (manr_4 + G_2 + S) \\ &+ C_{out} \cdot manr_4 \cdot (manr_5 + G_1 + G_2 + S)]. \end{aligned} \tag{7.29}$$

According to the notation defined in Section 7.3.2 $G_1 = manr_3$, $G_2 = manr_2$, $S = manr_1$ and $manr_4$ is the least significant bit of the mantissa field. The first part

of expression (7.29) gives the rounding condition when no normalization is required ($MSBP_{27} = 1$), ie the leading bit of the result is in bit position with significance of 1. The second part of this expression specifies the rounding condition in case of an overflowed mantissa sum ($C_{out} = 1$), in which case the resultant mantissa is shifted right by one bit to normalize it.

Two properties of floating point subtraction, as discussed in Section 7.3.2.2, are used in deriving the logic expression for the rounding condition following mantissa subtraction. Firstly, the normalization shifts following mantissa subtraction are to the left only, since there can be no mantissa overflow on subtraction ($C_{out} = 1$ indicates a positive and $C_{out} = 0$ indicates a negative mantissa difference) and secondly, when the normalization shift is two or more bit positions, then mantissa rounding is never necessary, since following normalization the first guard bit = 0.

A logic expression for the signal *Rnd* in the case when the output *manr* of the Mantissa Adder is positive, following mantissa subtraction, can be derived by considering columns 1-4 and 10 of Table 7.3:

$$\begin{aligned} Rnd_{subt,pos} &= EOP \cdot C_{out} \cdot [MSBP_{27} \cdot G_1 \cdot (manr_4 + G_2 + S) \\ &+ MSBP_{26} \cdot G_2 \cdot (G_1 + S)], \end{aligned} \tag{7.30}$$

According to expression (7.30) mantissa rounding is required when the effective mantissa operation is a subtraction ($EOP = 1$), the resultant mantissa is positive ($C_{out} = 1$) and one of the following conditions is true:

1. The output of the Mantissa Adder is normalized ($MSBP_{27} = 1$), $G_1 = 1$ and at least one of $manr_4$, G_2 or S also = 1 or,
2. The output of the Mantissa Adder requires a shift of one place to the left to normalize it ($MSBP_{26} = 1$), $G_2 = 1$ and either G_1 or S or both = 1.

The rounding condition in the case of the output of the Mantissa Adder being negative ($C_{out} = 0$) can be derived by considering columns 5-10 of Table 7.3:

$$\begin{aligned} Rnd_{subt,neg} &= EOP \cdot \overline{C_{out}} \cdot \Big\{ MSBP_{27} \cdot [\overline{G_1} \cdot (G_2 + S) + G_1 \cdot \overline{manr_4} \cdot \overline{G_2} \cdot \overline{S}] \\ &+ MSBP_{26} \cdot (\overline{G_2} \cdot S + \overline{G_1} \cdot G_2 \cdot \overline{S}) \Big\}. \end{aligned} \tag{7.31}$$

Combining expressions (7.29), (7.30) and (7.31) with the logical OR operations gives the overall rounding condition:

$$Rnd = Rnd_{add} + Rnd_{subt,pos} + Rnd_{subt,neg}. \tag{7.32}$$

7.7.11.2 Mantissa Overflow

In accordance with the flowchart given in Figure 7.4 the exponent needs to be modified following mantissa rounding if the latter results in mantissa overflow. In this design the exponent is modified *before* or *during* the operation of mantissa rounding. This is an additional novel architectural feature implemented in this design to enhance the performance of the FPA system.

An eventual mantissa overflow, which results during rounding or two's complementing, is anticipated by this look-ahead module allowing the operations of exponent modification and rounding (and/or two's complementing) to proceed in parallel. The required logic expressions are developed in the following subsections.

Mantissa Overflow: Rounding Following Addition

The only input to the ManInc that can cause the operation of mantissa rounding to overflow is of the form:

$$manr_{n,i} = 1, \quad i = 3, \ldots, 27. \tag{7.33}$$

This condition is detected by examining the carry propagate test signal Pr_{G_1}, described in Section 7.7.8, which indicates whether a carry of one generated at the least significant bit of the normalized resultant mantissa ($manr_n$) can propagate to the most significant bit position resulting in mantissa overflow. When the effective mantissa operation is an addition ($EOP = 0$) the mantissa overflow control signal ($ManOvf$) is given by the following Boolean expression:

$$ManOvf_{add} = \overline{EOP} \cdot (C_{out} + Pr_{G_1}), \tag{7.34}$$

where, as pointed out earlier, $C_{out} = 1$ indicates mantissa overflow as a result of addition.

Mantissa Overflow: Rounding Following Subtraction

It has been demonstrated in Section 7.3.2.2 that following mantissa subtraction the operation of mantissa rounding does not overflow when no normalization is required. Furthermore, it has been shown that mantissa rounding can only overflow if the operation of mantissa subtraction is followed by normalization shift of one bit.

Table 7.3 shows which combinations of the four bits ($manr_1$ to $manr_4$) of the Mantissa Adder output may generate a rounding condition ($Rnd = 1$) and in what cases the operations of mantissa rounding or two's complementing may result in mantissa overflow ($ManOvf = 1$).

Consider first the case of a positive mantissa difference. The overflow condition is determined as in the case of mantissa addition by examining the carry propagate test signal Pr_{G_1}:

$$ManOvf_{pos} = EOP \cdot C_{out} \cdot MSBP_{26} \cdot G_2 \cdot Pr_{G_1}. \tag{7.35}$$

Expression (7.35) indicates that when the effective mantissa operation is a subtraction ($EOP = 1$) and the output of the Mantissa Adder is positive ($C_{out} = 1$) the operation of mantissa rounding overflows when:

1. The normalization is a single place shift to the left ($MSBP_{26} = 1$) and
2. A potential carry signal can propagate to the most significant bit position of the normalized resultant mantissa ($Pr_{G_1} = 1$).

Note that in this case it is also necessary to check if $G_2 = 1$ because the condition $Pr_{G_1} = 1$ alone does not imply rounding, as was the case with mantissa addition.

When the result of mantissa subtraction is negative the overflow condition on rounding can be derived by considering the data in columns 4-10 of Table 7.3:

$$ManOvf_{neg} = EOP \cdot \overline{C_{out}} \cdot MSBP_{26} \cdot Pr_{G_1} \cdot (\overline{G_2} \cdot S + G_2 \cdot \overline{S})]. \tag{7.36}$$

Mantissa Overflow: Two's Complementing

The implementation of the MSBPF is such that when the Mantissa Adder output is negative and of the form $(2^i - 1)2^{-27}$ (where $3 < i < 27$) then the amount of normalization shift indicated by the signal $MSBP$ is one too many. This causes the operation of mantissa two's complementing to overflow. Such cases are detected with the use of the Pr_S test signal (defined in Section 7.7.8) which indicates that a carry, if present, will propagate from the position of the "sticky bit" (S) to the position of the most significant bit of the normalized resultant mantissa:

$$ManOvf_{comp} = EOP \cdot \overline{C_{out}} \cdot Pr_S. \tag{7.37}$$

The overall control signal indicating mantissa overflow is given by the following logic expression:

$$ManOvf = ManOvf_{add} + ManOvf_{pos} + ManOvf_{neg} + ManOvf_{comp}. \tag{7.38}$$

7.7.11.3 Physical Implementation of the ManFlags Module

The logical implementation of the ManFlags was challenging because of its high complexity and lack of any regularity. This module is required to generate the signals *Inx*, *SR*, *cpp*, *Rnd* and *ManOvf* according to the logic expressions (7.26), (7.27), (7.28), (7.32), and (7.38).

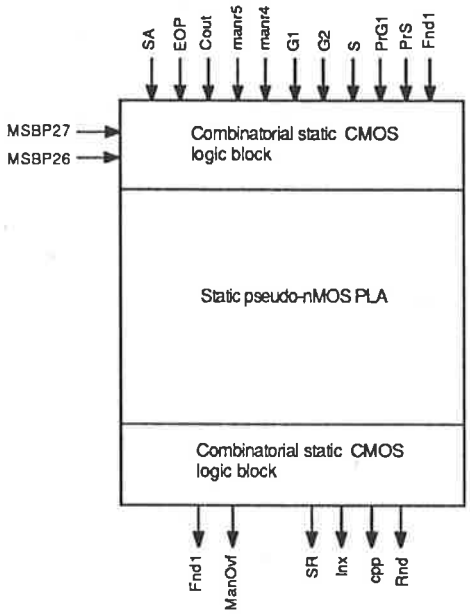


Figure 7.32: ManFlags: block diagram.

As discussed in Section 7.7.6, the delays of the circuit elements generating the signals *Rnd* and *cpp* are of equal importance in determining the overall performance of the FPA system as the time delay of the Normalizer, since the output of the Normalizer and the control signals *Rnd* and *cpp* constitute inputs to the ManInc which cannot stabilize its output before all the abovementioned inputs become stable.

It is also required to update the intermediate value of the resultant exponent in parallel with the operations performed by the ManInc. For this reason, the time delay taken to generate the control signal *ManOvf* is also equally significant, since this signal determines when the Exponent Output Stage can commence to stabilize its output. Thus, if the time delay of the module generating the signal *ManOvf* is too large there is no performance advantage in anticipating the mantissa overflow which results during mantissa rounding or two's complementing.

The logic expression for the *cpp* signal is simple and its fast implementation does not present any special problems. The critical paths of the circuit elements generating the control signals *Rnd* and *ManOvf* have been optimized by subdividing the relevant logic

expressions into two parts: those involving signals arriving early, ie EOP, S, G_1, G_2, $manr_4$ and $manr_5$, and those involving signals arriving later, ie C_{out}, $MSBP_{27}$, $MSBP_{26}$, Pr_{G_1} and Pr_S. In this way the ManFlags commences part of its operation in parallel with the Mantissa Adder, when the early arriving signals are available, and completes its operation in parallel with the Normalizer when the late arriving group of signals becomes available, ie when the output of the Mantissa Adder, the MSBPF and the Propagater become defined.

The logic implementation of the ManFlags module is composed of a combination of static CMOS combinatorial logic and a static PLA implemented using pseudo-nMOS logic. A block diagram showing the general structure of the ManFlags is given in Figure 7.32.

7.7.12 Mantissa Rounder-Complementer (ManInc)

Mantissa rounding is performed, if required, by taking the output $manr_n$ of the Normalizer and adding a one in the the least significant bit position. This function is achieved by a 24-bit incrementer module ManInc.

If the result of mantissa subtraction is negative it is represented in two's complement form and it must be converted to signed magnitude notation. The conversion is achieved by taking the two's complement of the output of the Normalizer and setting the sign of the result (SR) to one. The two's complement is formed, in the usual way, by entering the one's complement of the number to be complemented and adding one in the least significant bit position.

In the conventional floating point addition algorithm the output from the Mantissa Adder is complemented first, if it is negative, before being normalized and rounded. This necessitates a separate mantissa incrementing module connected to the output of the Mantissa Adder, in addition to a rounding module positioned at the output of the Normalizer. Thus, two mantissa incrementing circuits are required and the two operations of mantissa complementing and rounding are performed sequentially.

In order to save on hardware and increase the parallelism of the FPA system the two's complement is produced at the same time as the operation of mantissa rounding by a single module which is a modification of an incrementer circuit. In this way the tasks described in steps 5 and 9 of the conventional floating point algorithm given in Section 7.3.1 are combined into a single step. This is an additional novel performance enhancing feature of the hardware floating point addition system described in this chapter.

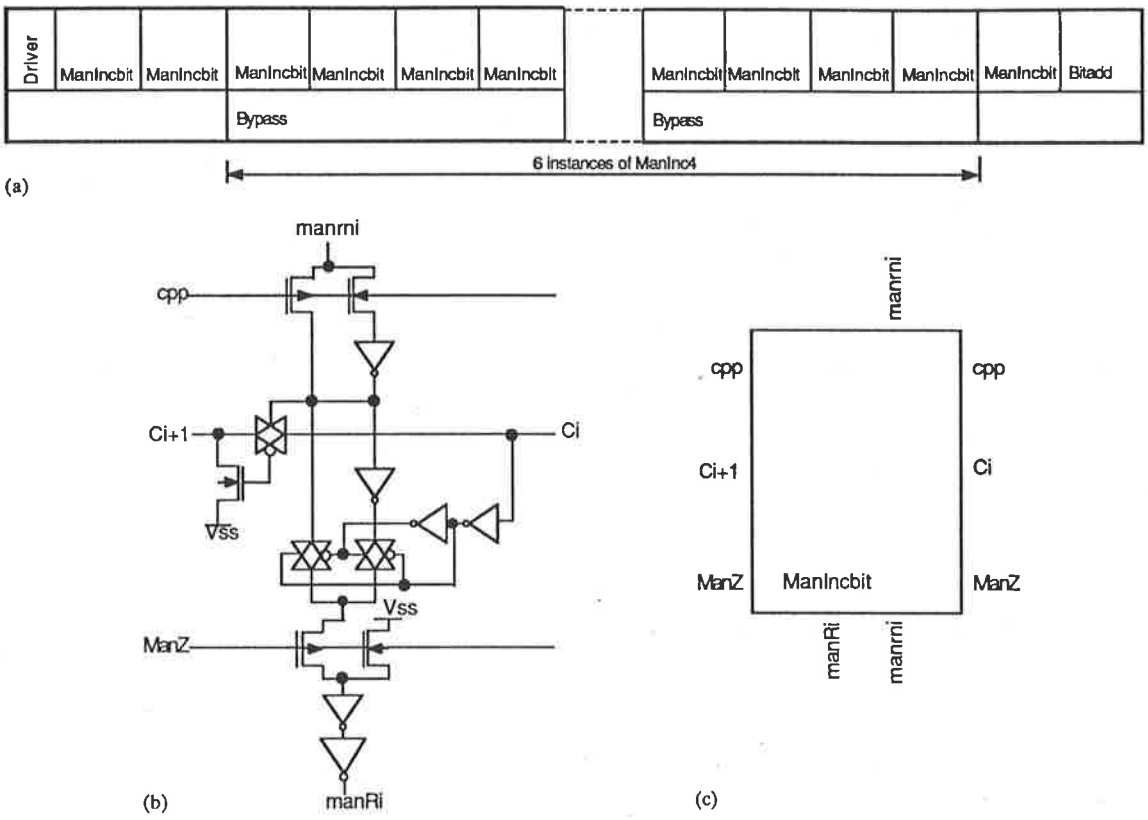


Figure 7.33: ManInc: (a) floorplan, (b) ManIncbit logic implementation, (c) ManIncbit leaf module environment specification.

For this to be possible the ManInc needs to be able to increment an input word by one or by two. This is achieved by replacing the 1-bit incrementer cell ManIncbit in the least significant bit position of the ManInc by a 1-bit adder cell Bitadd, as shown in Figure 7.33(a). The two's complementing is effected by summing the *cpp* control signal and the rounding is effected by summing the *Rnd* control signal in the additional input of this full adder module (both control signals are generated by the ManFlags, as discussed in Section 7.7.11).

At the leaf module level the 1-bit incrementer is simpler than a 1-bit adder cell. However, at the composition module level the performance of the ManInc is subject to the same limitation as the performance of the Mantissa Adder, ie the propagation of the carry signal. The maximum delay of the ManInc is decreased by employing the carry bypass technique, as discussed previously. The same carry Bypass circuit used to improve the performance of the Mantissa Adder is used in the ManInc. The results of circuit simulations and compaction indicate that a 50% speed increase is achieved, over a simple ripple carry incrementer, for an additional 35% in silicon area using the carry bypass technique. The floorplan of the ManInc is shown in Figure 7.33(a). The logic

implementation of the 1-bit module ManIncbit is shown in Figure 7.33(b) and its leaf module environment specification in Figure 7.33(c).

The necessary one's complementing logic is also included within the ManIncbit module, with the *cpp* control signal being used to effect the one's complementing operation. A signal driver module (Driver) is included on the left hand side of the ManInc to amplify the control signal *cpp* along the control line which span the complete length of this module.

The output of the ManInc is passed through a 2:1 multiplexer circuit which is used to set the mantissa field of the result to zero, in case of the final result being zero or infinity. The required control signal (*ManZ*) is generated within the Exponent Output Stage, described in the following section. The signal driver module (Driver) also amplifies the control signal *ManZ* along a control line spanning the complete length of the ManInc module.

7.7.13 Exponent Output Stage (ExpOtp)

The intermediate value of the resultant exponent (*expr*) may need to be decremented by an integer between zero and twenty four or incremented by one, to compensate for mantissa normalization. There is no simple method of combining these two operations and implementing them with a single module. As a result the Exponent Output Stage is specified as a composition of three submodules as depicted in Figure 7.34(a):

1. *Exponent Adjuster (ExpAdj)*: subtracts the encoded mantissa normalization distance (*adj*), generated by the PLA2, from the intermediate value of the resultant exponent (*expr*);

2. *Exponent Incrementer (ExpInc)*: operates under the control of the signal *ManOvf* and increments the exponent of the result by one in case of a mantissa overflow, and

3. *Flags Generator (ExpFlags)*: generates the *OVF*, the *UNF* and the *Z* flags. In addition, this module generates the control signal *ManZ* which forces the output of the ManInc to zero and generates the constants 255 or 0 for *expR*, if required.

Figure 7.34 (b) shows the floorplan of the Exponent Output Stage in terms of its composition modules.

The three modules are described in the following sections in terms of their compositions and functions.

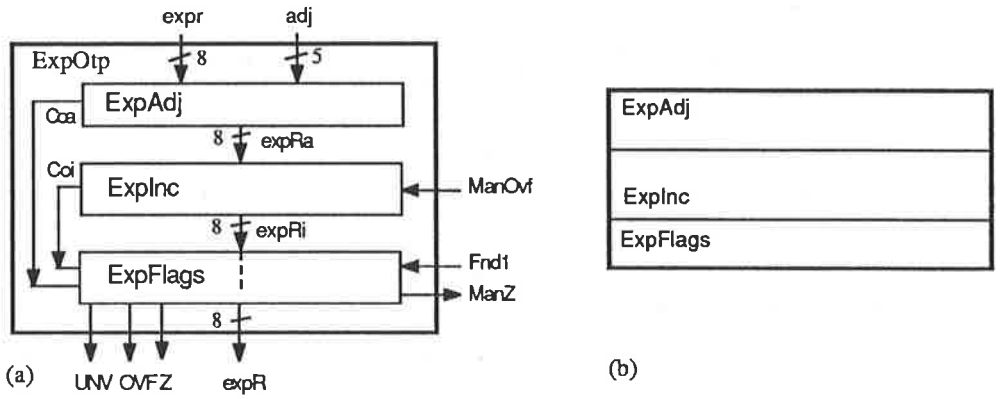


Figure 7.34: Exponent Output Stage: (a) block diagram, (b) floorplan.

7.7.13.1 Exponent Adjuster (ExpAdj)

This module adjusts the intermediate value of the resultant exponent by subtracting from it a 5-bit number *adj* corresponding to the amount of normalization shift applied to *manr* following a mantissa subtraction operation.

The Exponent Adjuster is functionally equivalent to the logic section of the Exponent Subtracter performing $expB - expA$. The floorplan of the Exponent Adjuster is similar to that shown in Figure 7.14(c), but with the ExpDiffbit modules replaced by 1-bit bit modules ExpAdjbit. The relevant parts of the 1-bit leaf module ExpDiffbit, shown in Figure 7.14(a) and (b), are reused in the implementation of the corresponding 1-bit functional module ExpAdjbit.

7.7.13.2 Exponent Incrementer (ExpInc)

This is an 8-bit incrementer circuit that increments by one the output ($expRa$) of the Exponent Adjuster, if there is mantissa overflow. The mantissa overflow signal ($ManOvf$) generated by the ManFlags constitutes the required increment control signal for the Exponent Incrementer.

The 1-bit incrementer leaf module ManIncbit (Figure 7.33(b)), modified not to contain the one's complementing components at its input and the multiplexing components at its output, is reused in the implementation of the 1-bit functional module ExpIncbit. The floorplan of the Exponent Incrementer is similar to the floorplan of the Exponent Subtracter, shown in Figure 7.14(c), with the ExpDiffbit modules replaced by the ExpIncbit modules.

The carry chains in the Exponent Adjuster and the Exponent Incrementer are buffered

after four 1-bit stages, as was done in the case of the Exponent Subtracter. The combined delay of the two 8-bit modules is found to be smaller than the delay of the 24-bit ManInc.

7.7.13.3 Flags Generator (ExpFlags)

In single precision floating point format, a result overflow occurs when the exponent of the result exceeds the maximum permitted value of 254. An overflow flag is generated and the final result delivered has the Standard's representation of infinity with an appropriate, as discussed in Section 6.4.4.

The logic equations for the required overflow flag (OVF) reads:

$$OVF = C_{oa} \cdot C_{oi} + expRa_8 \cdot expRa_7 \cdots expRa_2 \cdot expRa_1 \cdot C_{oa}, \tag{7.39}$$

where C_{oa} is the carry out from the most significant bit position of the Exponent Adjuster, indicating whether the output of Exponent Adjuster is positive or negative, and where C_{oi} is the carry out from the most significant bit position of the Exponent Incrementer, indicating whether the output of the Exponent Incrementer overflows.

It can be easily shown that when both operands are legal operands, then the intermediate value of the resultant exponent cannot exceed 255. However, if one or both of the operands are NAN or infinity then the resultant exponent can reach a value of 256. The first part of expression (7.39) indicates this condition. The second part of expression (7.39) indicates that $expRa = 255$. Note that one or both invalid operand flags ($invA$ and $invB$) are also asserted high by this module.

A result underflow occurs when the result of exponent adjustment becomes < 1. An underflow flag is set and the Standard's representation of zero with the appropriate sign is delivered as the final result, as discussed in Section 6.4.4. The required logic expression for the underflow flag which can be derived by considering operands causing result underflow is given by:

$$UNF = \overline{C_{o,a}} \cdot \overline{C_{o,i}}. \tag{7.40}$$

An additional function of the ExpFlags is to generate constants 255 (when the delivered result is infinity) and 0 (when the delivered result is zero) as the final value of the resultant exponent ($expR$). The required logic expression for $expR$ including the various pathological cases described above is given by:

$$expR_i = OVF + expRi_i \cdot \overline{ManZ} \cdot \overline{UNF} \cdot Fnd_1, \quad i = 1, \ldots, 8. \tag{7.41}$$

According to expression (7.41) $expR = 255$ if the result overflows (OVF=1), $expR = 0$ if the result underflows (UNF=1) or if the mantissa operation results in mantissa value of zero ($\overline{Fnd_1} = Z = 1$), and finally when there is no overflow or underflow and the resultant mantissa ($manr$) is different from zero then $expR = expRi$. When the delivered result is either the Standard's representation of infinity or is NAN, then the mantissa field of the result = 0. The control signal $ManZ$, generated by the ExpFlags, sets the mantissa of the final result to zero. Following the above discussion the required logic expression can be derived and it is given by:

$$ManZ = OVF + UNF + \overline{Fnd_1}, \tag{7.42}$$

where the signal $\overline{Fnd_1}$ indicates the special case of computed zero for the mantissa field and it is generated by the MSBPF, as described in Section 7.7.7.

7.8 Design Evaluation

The area requirements and the speed performance of the Floating Point Adder (FPA) are compared to those of the conventional hardware realization (CFPA) of floating point addition.

7.8.1 Area Requirements

The silicon area occupied by the FPA system is approximately 10% larger than that of the CFPA realization. This is established by examining the major differences between the decompositions of the FPA and the CFPA systems.

The implementation of the CFPA (Figure 7.2) includes an extra mantissa incrementing module, not present in the FPA, at the output of the Mantissa Adder module, to perform two's complementing of the resultant mantissa. The FPA, on the other hand, includes the Propagater module which is not present in the CFPA system. The functional complexities of the Propagater module and the additional mantissa incrementing module of the CFPA system are approximately the same and, thus, it can be assumed that the silicon areas occupied by these modules are also equal.

The implementation of the MSBPF module (denoted CMSBPF) in the CFPA system is slightly simpler and therefore occupies less area than the MSBPF module in the FPA system. A possible design of the CMSBPF module is given in [ChFi83]. It is based on the "divide and conquer" approach where the output of the Mantissa Adder is divided into 8-bit subwords and the position of the leading bit is found in parallel in each of

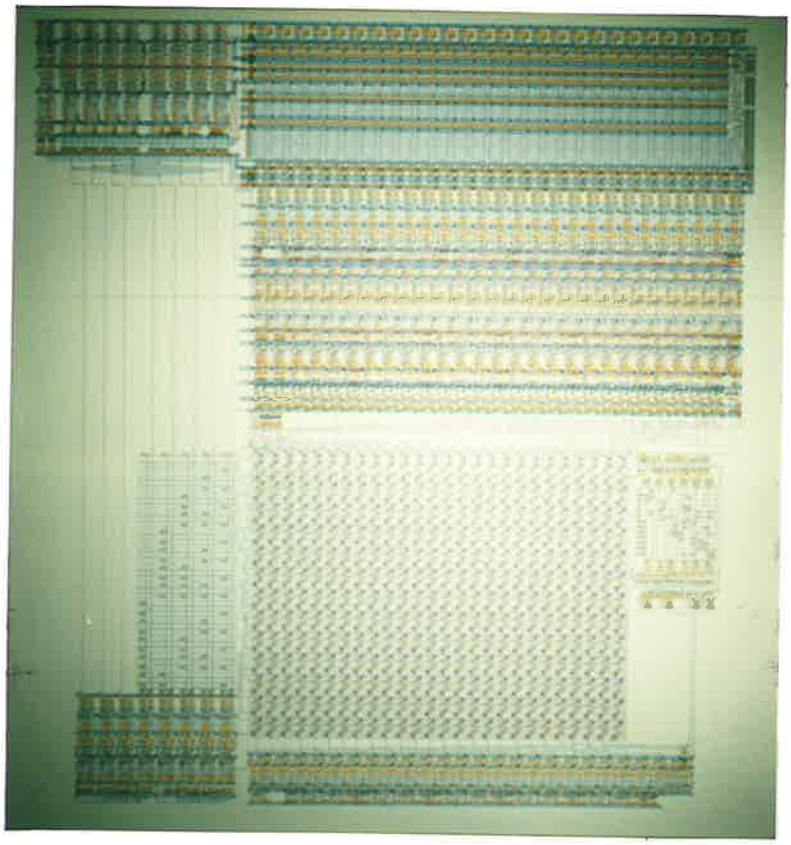


Figure 7.35: Floating Point Adder: symbolic domain representation.

the subwords. A 6-bit binary number is then generated indicating the position of the leading bit in the 27-bit mantissa adder output or indicating that the result is a zero.

This module, as described in [ChFi83], was implemented in the symbolic domain to obtain speed and area estimates. It was found that the number of transistors required to implement the CMSBPF module was about 70% of the number of transistors in the MSBPF module. The required module interconnection in the CMSBPF was more complex, however, with a large percentage of the wires spanning a distance corresponding to the length of an 8-bit section of the MSBPF and with a 6-bit bus delivering the final result spanning the complete length of the module. These long wires present in the CMSBPF module result in the difference in the silicon area occupied by the CMSBPF module being only 10% less than the area of the MSBPF module. Thus, if all other modules are implemented using the same circuit design techniques, the total silicon areas of the CFPA and of the FPA will be approximately the same.

The silicon area of the FPA system was estimated by separately compacting a number of substantial parts of the circuit (since it was still not possible to compact the complete circuit with the available software tools), namely a four bit slice of the man-

tissa data path, the Exponent Input Stage and the Exponent Output Stage modules. It is found that for the hypothetical 1-micrometer double metal p-well $Orbit^+$ process the dimensions of the overall bounding box of the FPA system are $2600\lambda \times 2800\lambda$ or $(1.30 \times 10^{-3}m) \times (1.40 \times 10^{-3}m)$.

The total number of transistors in the FPA system is 9614 with the total number of manually placed transistors being 770, giving a regularity factor of $9614/770 = 12.5$. A plot of the complete FPA system description in the symbolic domain is shown in Figure 7.35.

7.8.2 Speed Performance

It was possible to perform circuit simulations at the transistor level using the modified FACTS program and circuit connectivity information extracted from the symbolic circuit domain on the complete 32-bit FPA system. The process parameters were set for the $Orbit^+$ process, as discussed in Section 4.6. A typical simulation run took an average of 95 cpu minutes on a VAX[1] 785 computer. A total of 40 different simulation runs of the complete circuit were performed (in addition to the simulations performed on separate functional modules) in order to establish a degree of confidence in the functional correctness, to establish that the planned level of module concurrency was achieved and to obtain a performance estimate.

The following discussion refers to the data flow graph for the FPA system shown in Figure 7.36. (A corresponding data flow graph for the CFPA system is given in Figure 7.37.) The following conclusions have been made about the delays within the Align Unit, using FACTS simulations. The delays of the data paths between the modules ExpInp (Exponent Input Stage) and Mantissa Switch (ManSw) and between the modules ExpInp and Sign Unit (denoted SU in the figure) are equal and larger than the data path delay between the modules ExpInp and Alignment Shifter. The data path delays between the modules Sign Unit and Mantissa Complementer (ManComp) are negligible compared to the delays of the remaining data paths within the Align Unit module. A maximum delay of the Align Unit module results when there is a carry propagating across the complete length of the ExpInp module, which module must stabilize its outputs before the modules Mantissa Switch and Alignment Shifter can evaluate their outputs. The data signal then propagates through the modules Mantissa Switch, Alignment Shifter and Mantissa Complementer before it enters the Operate module. As stated at the beginning of this chapter, the part of the floating point algorithm performed by the

[1] VAX is a trademark of the Digital Equipment Corporation.

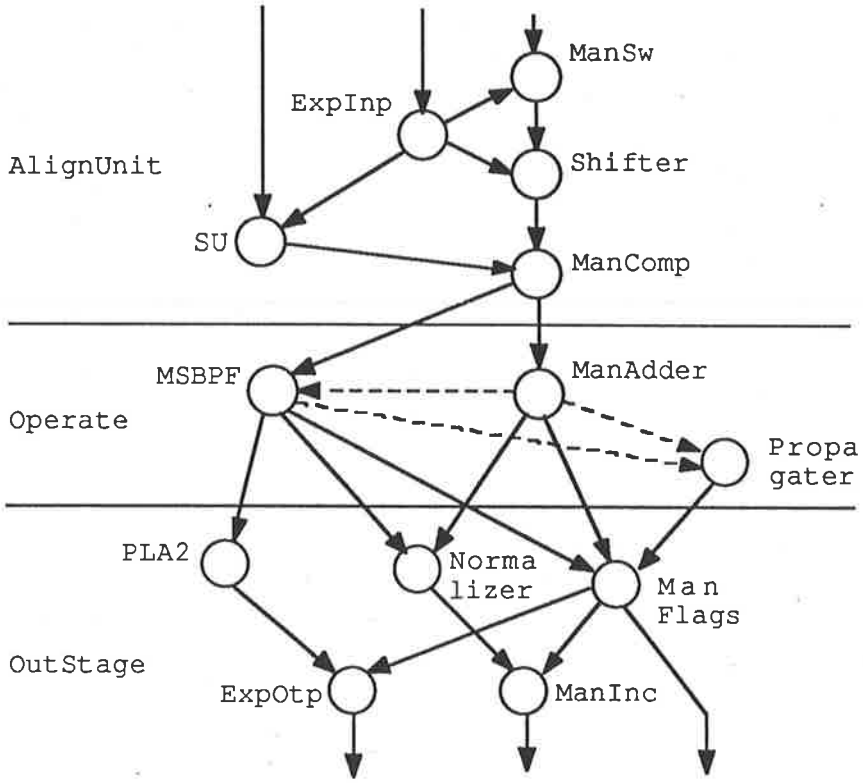


Figure 7.36: FPA: data flow graph.

Align Unit is carried out in the same manner in both the FPA and the CFPA systems. Thus, the implementation of the Align Unit module is the same in both the FPA and the CFPA systems.

An examination of the remainder of Figure 7.36 reveals that the many possible data paths make it difficult to determine the worst case delay of the FPA system. The data paths between the modules MSBPF, Mantissa Adder and Propagater, which are indicated by broken lines, are a direct result of parts of circuitry being shared between the three modules. The delays along these paths are data independent and small compared to, for example, the maximum possible delay of the MSBPF module. Thus, these three modules can be assumed to commence evaluating their outputs simultaneously.

As discussed previously, for best operation of the FPA system the time delays of the modules Mantissa Adder, MSBPF and Propagater must be equal and as small as possible. With the outputs of these three modules stable, the modules Normalizer, PLA2 and ManFlags simultaneously begin to generate their stable outputs, followed by the generation of the stable outputs by modules ExpOtp (Exponent Output Stage) and ManInc, which modules also operate in parallel. A number of simulation results exercising possible critical paths of the FPA system are presented below. Since the various modules contribute differently to the overall FPA system delay, during mantissa addition and

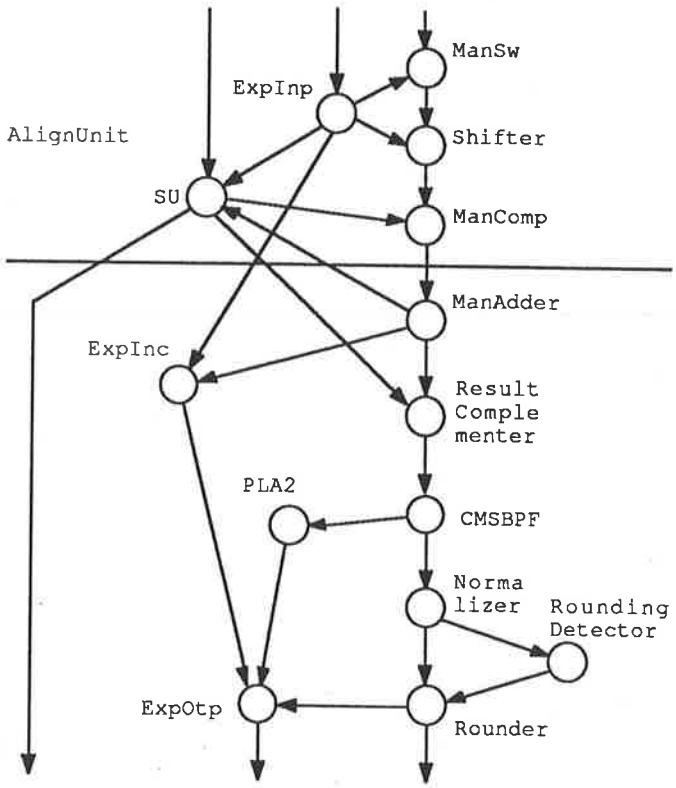


Figure 7.37: Conventional Floating Point Adder: data flow graph.

subtraction, the discussion will be divided accordingly.

Addition

Two cases are of special interest:

1. Example 1: mantissa overflow ($C_{out} = 1$) during the operation of mantissa addition (set 1 of the input vectors given in Table 7.4 and the results shown in Figure 7.38); and

2. Example 2: mantissa overflow during the operation of mantissa rounding (set 2 of the input vectors given in Table 7.4 and the results shown in Figure 7.40).

Because of space limitation, it is not possible to present the voltage waveforms at all input and output nodes and some internal nodes of the circuit, and therefore only the voltage waveforms at a number of "interesting" nodes at various points of the critical signal path are given. The input vectors of interest are applied at time $t = 40ns$, with the time prior to that point reserved for the circuit to stabilize, or for a set of initial conditions to be applied (eg inputs to set the carry chains in the various modules to

Figure	time	quantity	value
7.38	$< 40ns$	expB	01111110
(set 1)		manB	1.00000000000000000000001
		expA	01111111
		manA	1.01111111111111111111111
	$> 40ns$	expB	01111111
		manB	1.01111111111111111111111
		expA	01111110
		manA	1.00000000000000000000010
7.40	$< 40ns$	expB	01111110
(set 2)		manB	1.00000000000000000000000
		expA	01111111
		manA	1.00000000000000000000000
	$> 40ns$	expB	01111111
		manB	1.00000000000000000000000
		expA	01111110
		manA	1.11111111111111111111111

Table 7.4: Addition input vectors.

logic level one). The following discussion will only consider the simulation results after time $t = 40ns$.

Example 1

Set 1 of the input vectors, given in Table 7.4, causes carry signals to propagate across the complete lengths of the ExpDiff, the Mantissa Adder and the ExpOtp modules. With reference to Figure 7.38, the control signal $C_{o,ex}$ (waveform Coex) stabilizes at $t = 55ns$, at which time the Mantissa Switch passes the mantissa of the smaller operand ($manA$) to the Alignment Shifter. The shift distance control signal becomes valid at $t = 61ns$ (waveform shift2), at which time the Alignment Shifter commences the required shifting operation. The mantissa of the larger operand ($manB$) becomes available at the inputs of the Mantissa Adder (waveforms Y4 and Y26), after passing through the Mantissa Complementer, at $t = 57ns$, while the unnormalized mantissa stabilizes at $t = 71ns$ (waveform X4). This corresponds to the completion of the sequence of operations performed by the Align Unit and to the commencement of the Mantissa Adder operation. The maximum delay (t_{align}) of the Align Unit is therefore $t_{align} = 71ns - 40ns = 31ns$. Note that the intermediate value of the resultant exponent (waveform expr1) stabilizes at $t = 58ns$, which is well within the maximum delay of the Align Unit module.

The mantissa sum becomes available at $t = 89ns$ (waveform manr25). Bit $manr_{25}$ is the last bit of the mantissa sum to stabilize. This is as expected, since in order to define

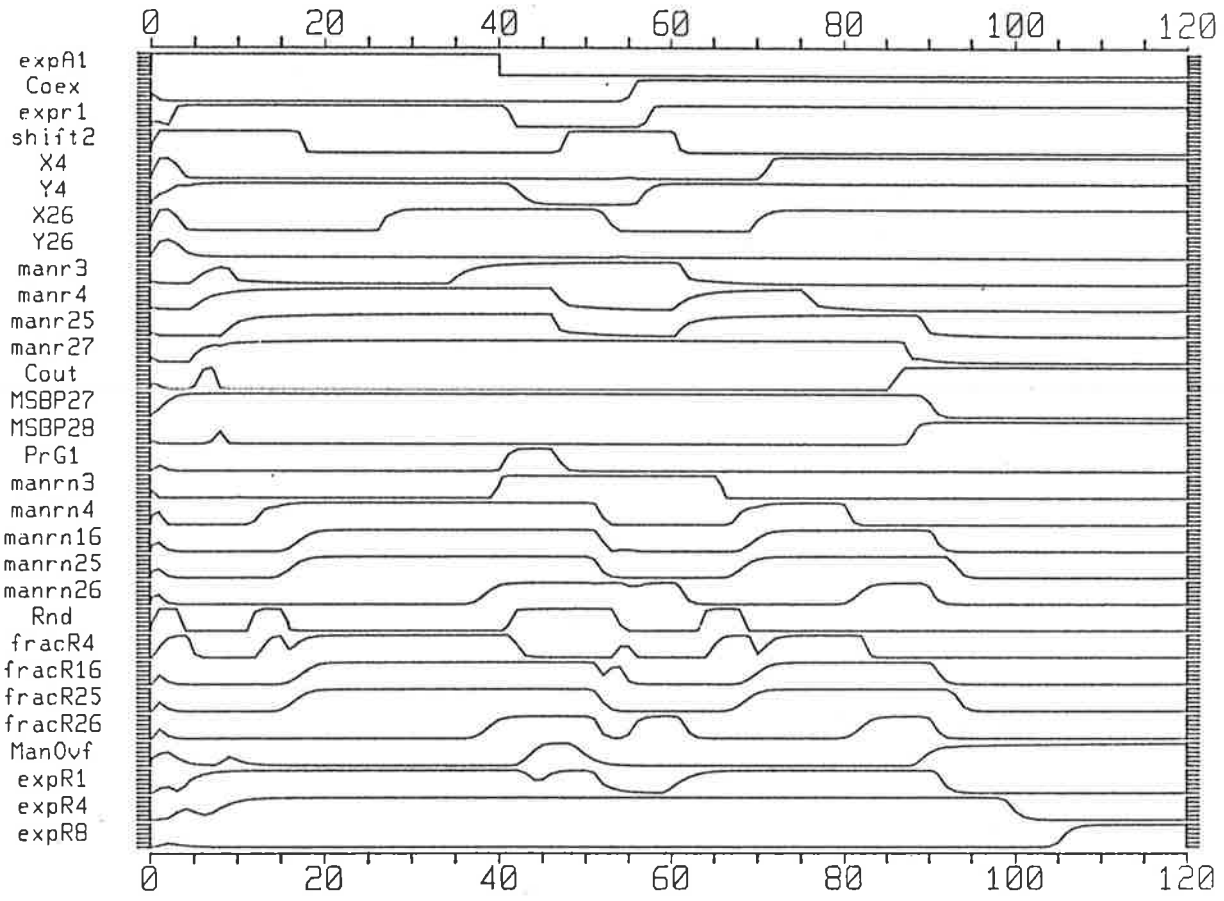


Figure 7.38: Floating Point Adder: FACTS simulation results for set 1 of the input vectors of Table 7.4. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

the output $manr_{25}$, the carry signal has to ripple through the four 1-bit adder (Bitadd) modules of the last Bit4add module, and because the carry signal propagates much faster across the Bypass module than it does across these four 1-bit adder modules, the carry signal reaches bit position $i = 27$ before it reaches bit position $i = 25$, resulting in $manr_{27}$ becoming stable before $manr_{25}$ becomes stable. The maximum delay (t_{add}) of the Mantissa Adder is $= 89ns - 71ns = 18ns$.

The signal $MSBP_{27}$, which is the slowest output of the MSBPF, for this combination of inputs, settles to its final value at approximately the same time as the slowest bit of the mantissa sum becomes stable. Thus, in this case the delay (t_{msbpf}) of the MSBPF is approximately equal to t_{add}, and as intended the MSBPF and the Mantissa Adder operate in parallel.

The mantissa overflow signal ($ManOvf$) becomes defined at $t = 90ns$ and $ManOvf = 1$ as a result of the signal C_{out} being $= 1$ (see expression (7.34)), ie mantissa addition resulted in an overflow. The actual delay of the ManFlags is $t_{ovf} = 90ns - 86ns = 4ns$ (C_{out} which defines $MAnOvf$ stabilizes at $t = 86ns$), but the effective delay of this

module is $t_{ovf} = 90ns - 89ns = 1ns$, where $t = 89ns$ is the time at which the last bit of the Mantissa Adder output stabilizes. This is a direct consequence of using the carry bypass technique which results in the carry out (waveform Cout) from the most significant bit position of the Mantissa Adder becoming stable before the last bit of the sum output (waveform manr25) becomes stable, and consequently the ManFlags commences its operation before the slowest output of the Mantissa Adder becomes stable. (Note that the signal *EOP*, which is not shown but also used to define *ManOvf*, stabilizes at $t = 43ns$ and thus the delay of this signal in insignificant).

The exponent incrementing module (ExpInc) starts to evaluate as soon as *ManOvf* stabilizes, ie at $t = 90ns$, with the slowest bit of output ($expR_8$) becoming available at $t = 106ns$. The maximum delay (t_{exp}) of this module is $t_{exp} = 106ns - 90ns = 16ns$. The evaluation of the status flags occurs in parallel and the final flags also become available at $t = 106ns$ (the flags are not included in the figure for the reasons of limited space).

The final result can be latched into the output register a delay $t_{FPA,set1} = 66ns$ after the application of the inputs. The delay FPA system for this set of input vectors can be written as:

$$\begin{aligned} t_{FPA,set1} &= t_{align} + t_{add} + t_{ovf} + t_{exp}, \\ &= 31ns + 18ns + 1ns + 16ns, \\ &= 66ns. \end{aligned} \tag{7.43}$$

For this particular set of input vectors the CFPA system experiences approximately the same delay as the FPA system, provided the same circuit design methods are used in the implementation of the corresponding modules. This is because:

1. Firstly, the implementation of the Align Unit in both the FPA and the CFPA systems are identical and therefore the delay t_{align} is also identical.

2. Secondly, there is no performance advantage, in this case, gained in the FPA system over the CFPA system by the use of the novel MSBPF module, since the position of the leading bit can only be determined once the carry signal has propagated across the complete length of the Mantissa Adder.

This is graphically shown in Figure 7.39, where the signal path with the longest time delay for this set of vectors is indicated using a data flow diagram. The longest signal path includes the modules Align Unit, Mantissa Adder, ManFlags and ExpOtp. The

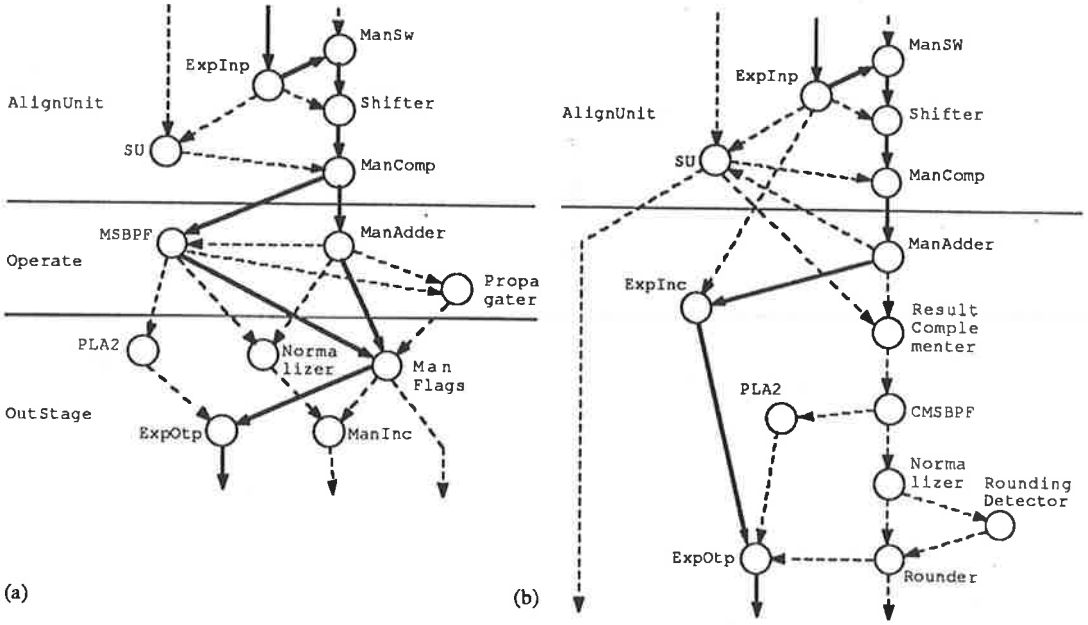


Figure 7.39: Data path with the longest time delay for set 1 of the input vectors given in Table 7.2: (a) FPA system, (b) CFPA system.

corresponding signal path for the CFPA system for this case of input vectors is shown in Figure 7.39(b).

Example 2

The difference in the time delays of the FPA and CFPA systems is influenced, in this case (set 2 of the input vectors given in Table 7.4), by the time each system takes to detect mantissa sum rounding condition and mantissa overflow. The advantage of the additional parallelism in the FPA system now becomes apparent.

With reference to Figure 7.40, the position of the leading bit (indicated by the waveform MSBP27) is detected early at $t = 62ns$, for this set of input vectors, since the carry signal does not propagate across the complete length of the Mantissa Adder. The Rnd control signal is evaluated according to expression (7.29) as soon as the signal $manr_4$ becomes available. The input (X_4) to the Mantissa Adder generating this signal passes through the Alignment Shifter and stabilizes at $t = 70ns = 40ns + t_{align}$. The Mantissa Adder output bit of interest (waveform manr4) stabilizes at $t = 74ns$. The Rnd control signal becomes available at $t = 79ns$, at which time the mantissa rounding operation commences. The final rounded mantissa becomes available at $t = 97ns$ (waveform fracR25).

This mantissa rounding operation produces mantissa overflow, but this overflow is an-

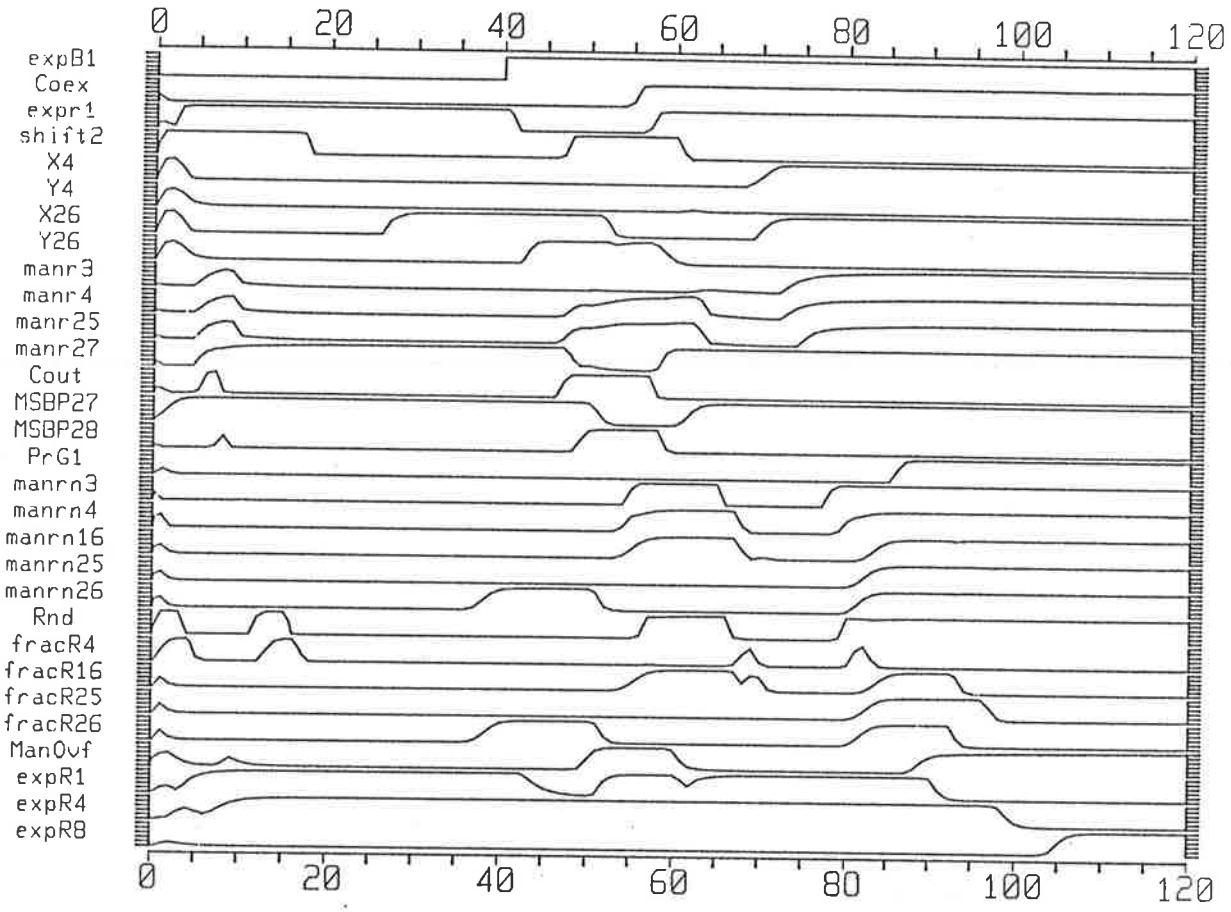


Figure 7.40: Floating Point Adder: FACTS simulation results for set 2 of the input vectors of Table 7.4. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

ticipated in advance by evaluating the Pr_{G_1} signal in parallel with the operation of mantissa addition. This signal stabilizes at $t = 86ns$. Note that in Example 1, where the carry signal was made to propagate across the complete length of the Mantissa Adder, the signal C_{out} stabilized at $t = 86ns$ (Figure 7.38 waveform Cout). Thus, the maximum delay for the signals Pr_{G_1} and C_{out} are the same and the fact that both stabilize at the same time, when made to propagate across the complete lengths of the respective modules, indicates that indeed the modules Mantissa Adder and Propagater evaluate in parallel, as intended.

The *ManOvf* signal stabilizes at $t = 90ns$, according to expression (7.34), following the evaluation of Pr_{G_1}. The delay of the ManFlags is $t_{ovf} = 4ns$, and in this case the maximum and the effective delays of this module are equal. The ExpInc commences to update the intermediate value of the resultant exponent at $t = 90ns$ with final value of the exponent becoming available at $t = 106ns$. This time difference $(106ns - 90ns)$ corresponds to the maximum time delay of the ExpInc module.

A diagram indicating the data signal path having the maximum time delay, for this set

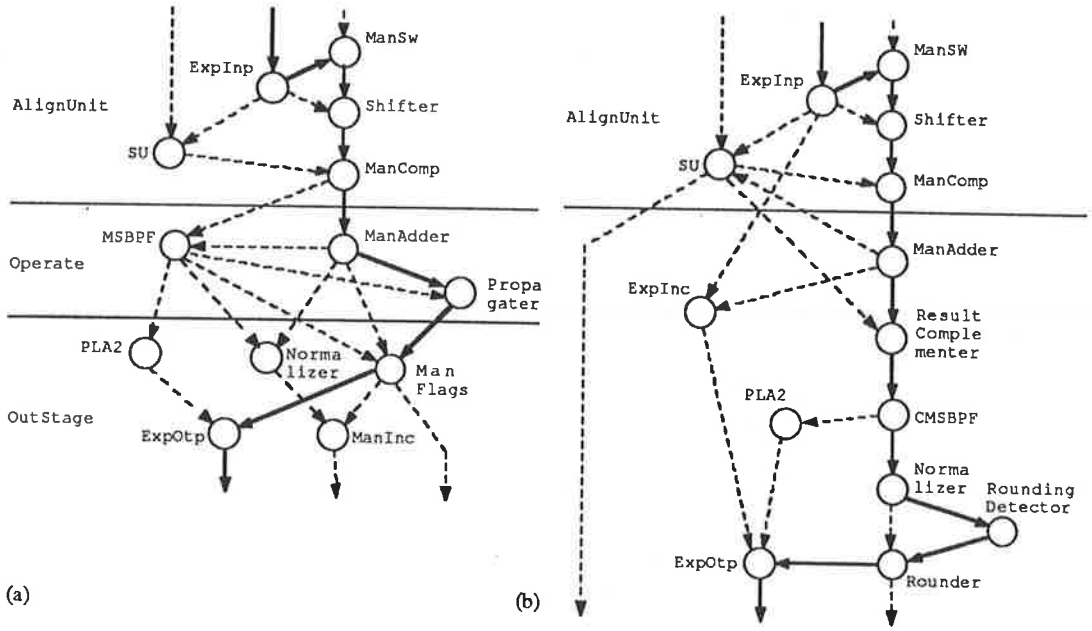


Figure 7.41: Data path with the maximum time delay for set 2 of the input vectors given in Table 7.4: (a) FPA system, (b) CFPA system.

of input vectors, is shown in Figure 7.41(a) for the FPA system. With reference to this figure and the above discussion, the delay of the FPA system to perform this addition is equal to the sum of the following:

1. The time delay (t_{align}) of the Mantissa Alignment module;
2. The maximum delay (t_{prop}) of the Propagater module;
3. The maximum delay (t_{ovf}) of the ManFlags module; and
4. The maximum delay (t_{exp}) of the ExpInc module, ie

$$\begin{aligned} t_{FPA,set2} &= t_{align} + t_{prop} + t_{ovf} + t_{exp}, \\ &= 30ns + 16ns + 4ns + 16ns, \\ &= 66ns. \end{aligned} \tag{7.44}$$

The corresponding delay of the CFPA system can be estimated using the data flow diagram of Figure 7.41(b) and is given by the sum of the following time delays:

1. The time delay (t_{align}) of the Alignment Unit;

2. The time delay (t_{add1}) of a 1-bit mantissa adder module (there is no propagation of the carry signal);

3. The delay (t_{compl}) of a 1-bit mantissa complementing module (there is also no propagation of the carry signal across the module that performs the two's complementing of the resultant mantissa and which operates on the output of the Mantissa Adder module);

4. The time delay (t_{norm}) of the normalizing shifter;

5. The time delay (t_{rnd}) required to determine the mantissa rounding condition on the basis of the output of the normalizing shifter module;

6. The maximum time delay (t_{round}) of the mantissa rounding module;

7. The maximum time delay (t_{exp}) of the exponent incrementing module, ie

$$\begin{aligned} t_{CFPA,set2} &= t_{align} + t_{add1} + t_{compl} + t_{norm} + t_{rnd} + t_{round} + t_{exp}, \\ &= 30ns + 3ns + 3ns + 8ns + 4ns + 18ns + 16ns, \\ &= 82ns. \end{aligned} \tag{7.45}$$

The various delays have been obtained from circuit simulations performed on the individual modules.

Comparing expressions (7.44) and (7.45) gives a delay advantage of the FPA system over the CFPA system of approximately $16ns$ or 20%. This delay reduction is achieved as a result of:

1. The anticipation of the mantissa overflow which eventually occurs during rounding;

2. The *Rnd* signal being evaluated using the unnormalized mantissa, rather than the normalized mantissa (the rounding condition is evaluated in parallel with mantissa normalization operation); and

3. The absence of the two's complementing module at the output of the Mantissa Adder.

Subtraction

The time delays of the FPA and CFPA systems are compared with reference to the simulation results of Figure 7.42, the corresponding input vectors being shown in Ta-

Figure	time	quantity	value
7.42	$< 40ns$	expB	10000000
		manB	1.00000000000000000000001
		expA	10000001
		manA	1.00000000000000000000000
	$> 40ns$	expB	10000000
		manB	1.00000000000000011111111
		expA	01110000
		manA	1.11111111000000000000000

Table 7.5: Subtraction input vectors.

ble 7.5. The effective mantissa operation is a subtraction ($SB = 1$, $SA = 0$, $OP = 0$), with $manB$ being subtracted from $manA$.

For the input vectors given in Table 7.5 $manA$ is shifted 16 positions to the right to align it with $manB$. A carry signal ($= 1$) propagates across the complete lengths of the Mantissa Adder and the ManInc. The position of the leading bit depends in this case on the value of the carry signal. No mantissa rounding or complementing are required in this case. The interesting feature of this set of input vectors is that initially an incorrect value of the Rnd control signal is evaluated, causing mantissa rounding and subsequent overflow to take place.

With reference to Figure 7.42, the control signal $C_{o,ex}$ stabilizes at $t = 48ns$, at which time the Mantissa Switch passes the required mantissa to the Alignment Shifter. The shift distance control signal becomes valid at $t = 53ns$ (waveform shift1), at which time the Alignment Shifter commences the required shifting operation on $manA$. The shifted mantissa passes through the Mantissa Complementer module with the output of this module stabilizing at $t = 62ns$ (waveform X3). This corresponds to the completion of the sequence of operations performed by the Align Unit. The delay of this module is $t_{align} = 62ns - 40ns = 22ns$, for this set of inputs.

The mantissa difference becomes available at $t = 80ns$ (waveform manr25). Thus, the maximum delay (t_{add}) of the Mantissa Adder module is $= 18ns$.

The signal $MSBP_{26}$, indicating the position of the leading bit of the unnormalized mantissa ($manr$), settles to its final value at $t = 82ns$. In this case, the value of the $MSBP$ signal depends, as stated earlier, on the value of the carry signal within the Mantissa Adder (see expressions (7.19)-(7.22) in Section 7.7.7), and in particular, on the value of the carry signal in position $i = 27$. Thus, the effective time delay (t_{msbpf}) needed to detect the position of the leading bit is only $2ns$.

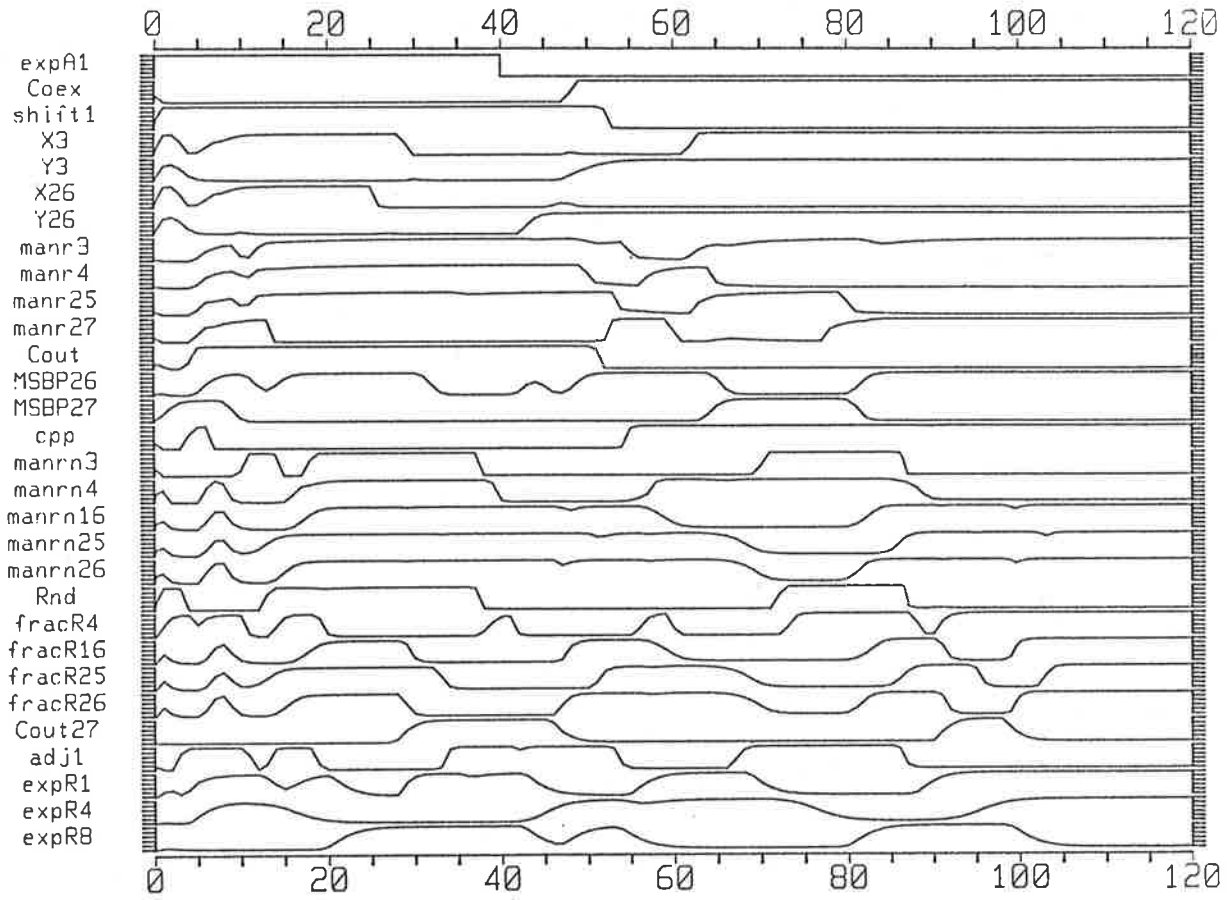


Figure 7.42: Floating Point Adder: FACTS simulation results for the input vectors of Table 7.5. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

With the normalization shift distance defined, the correct value of the *Rnd* signal is determined (according to expression (7.30)) at $t = 87ns$, and the shift distance is converted to the 5-bit number *adj* at $t = 87ns$ (waveform adj1), to be subtracted from the intermediate value of the resultant exponent. Both, the ExpOtp and the ManInc commence evaluation simultaneously at $t = 87ns$, with carry signals propagating across complete lengths of the two modules. The last bit of the ManInc output (waveform fracR25) stabilizes at $t = 104ns$. The evaluation of the status flags occurs in parallel and the final flags also become available at $t = 103ns$ (the flags are not included in the figure for the reasons of limited space).

A diagram indicating the data signal path having the maximum time delay, for this set of input vectors, is shown in Figure 7.43(a) for the FPA system. With reference to this figure and the above discussion, the delay of the FPA system to perform this subtraction is equal to the sum of the following:

1. The delay (t_{align}) of the Alignment Unit;

2. The maximum delay (t_{msbpf}) of the MSBPF module;

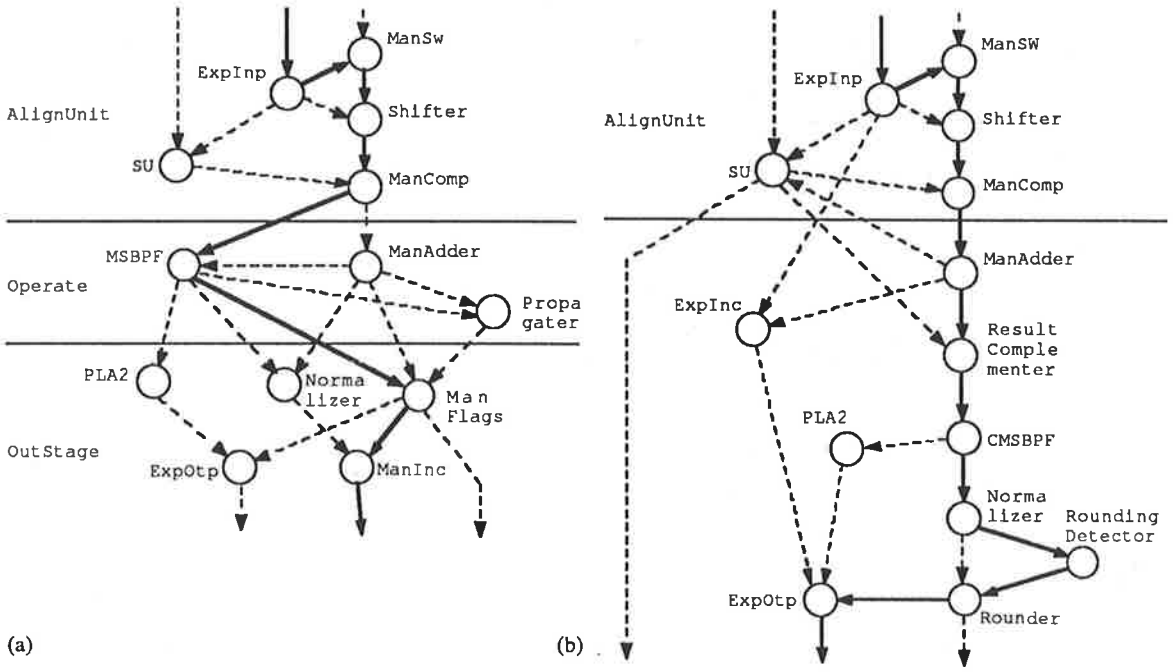


Figure 7.43: Data path with the maximum time delay for the input vectors given in Table 7.5: (a) FPA system, (b) CFPA system.

3. The delay (t_{ovf}) of the ManFlags module; and

4. The maximum delay (t_{round}) of the ManInc module, ie

$$\begin{aligned} t_{FPA,subt} &= t_{align} + t_{msbpf} + t_{ovf} + t_{round}, \\ &= 22ns + 20ns + 4ns + 18ns, \\ &= 64ns. \end{aligned} \tag{7.46}$$

The corresponding delay of the CFPA system can be estimated using the data flow diagram of Figure 7.43(b). Following the time at which the carry chain within the Mantissa Adder stabilizes ($t = 80ns$, ie waveform manr25 of Figure 7.42) the position of the leading bit will be available at $t = 80ns + t_{comp} + t_{cmsbpf}$. (The maximum delay ($t_{cmsbpf}$) of the CMSBPF, implemented according to [ChFi83], was found to be $\approx 14ns$.) With the normalization shift determined by the CMSBPF module, the normalized mantissa will become available at $t = 80ns + 3ns + 14ns + t_{norm} = 105ns$. The rounding condition will be evaluated at $t = 105ns + t_{rnd} = 109ns$. Because the input to the mantissa rounder (Rounder) has ones in all bit positions, a carry of zero will propagate across the complete length of this module cancelling the carry high propagated before the correct value of *Rnd* was established. The mantissa will become available at $t = 109ns + t_{round} = 127ns$. The control signal *ManOvf*, and the value of the *expr* will result in a carry low prop-

agating across the complete length of the exponent incrementing module (part of the ExpOtp), with the final result becoming stable at $t = 127ns + t_{exp} = 143ns$, giving $t_{CFPA,subt} = 103ns$. Thus, the total delay can be expressed as a sum of the following time delays:

1. The time delay (t_{align}) of the Alignment Unit;
2. The maximum delay (t_{add}) of the Mantissa Adder;
3. The time delay (t_{compl}) of a 1-bit mantissa two's complementing module;
4. The maximum time delay (t_{cmsbpf}) of of the CMSBPF;
5. The time delay (t_{norm}) of the Normalizer;
6. The time delay (t_{rnd}) required to determine the mantissa rounding condition on the basis of the output of the Normalizer;
7. The maximum time delay (t_{round}) of the mantissa rounding (Rounder) module;
8. The maximum time delay (t_{exp}) of the exponent incrementing module, ie

$$\begin{aligned} t_{CFPA,subt} &= t_{align} + t_{add} + t_{comp} + t_{cmsbpf} + t_{norm} + t_{rnd} + t_{round} + t_{exp}, \\ &= 22ns + 18ns + 3ns + 14ns + 8ns + 4ns + 18ns + 16ns, \\ &= 103ns. \end{aligned} \tag{7.47}$$

This last set of test vectors corresponds to the longest delay of the CFPA system. The longest delay of the FPA system occurred with the third set of addition test vectors and was $t_{FPA,set1} = 66ns$. Thus, the maximum possible speed increase of the FPA system is approximately $103ns - 66ns = 37ns$ or 36%.

7.9 Summary

The design of a novel VLSI architecture (the FPA system) of the floating point adder circuit has been described. The circuit was implemented in the symbolic domain using a combination of such circuit design techniques as: static combinatorial complementary CMOS, pseudo-nMOS and pass transistor logic.

The silicon area requirements of the FPA system is approximately 10% larger than that of the commonly used hardware realizations of this operation. The resulting speed increase has been achieved by increasing the degree of parallelism in this design by:

1. Determining the position of the leading bit in parallel with the mantissa subtraction operation;

2. Determining the mantissa rounding condition using the mantissa adder output prior to its normalization;

3. Updating the value of the resultant exponent in parallel with mantissa rounding operation, by anticipating the possible mantissa overflow during mantissa adder evaluation period; and

4. Performing the operations of mantissa two's complementing and rounding in parallel.

Circuit simulations using the modified FACTS program, with the process parameters corresponding to the hypothetical 1-micrometer CMOS process ($Orbit^{+}$), indicate a speed increase, under the worst case delay conditions, of approximately 36%, compared to the speed of the conventional VLSI architecture of this arithmetic system implemented using the same circuit design techniques.

Chapter 8

The Floating Point Multiplier

8.1 Introduction

This chapter presents a hardware realization of floating point multiplication incorporating novel ideas at the architectural level. The Floating Point Multiplier (FPM) circuit constitutes one of the major components in the Mathematical Processor system. The implementation of the parallel integer multiplier is the dominant factor determining the time delay, silicon area requirement and the power consumption of any floating point multiplier system; the design of this module is thus of crucial importance.

Historically, the need for faster response and higher throughput of digital signal processing systems, whose performance is mainly determined by the speed of multiplication, has been the major motivation of efforts to design faster integer multiplier circuits. A parallel integer multiplier has become an indispensable component in VLSI Digital Signal Processing (DSP) chips designed today and in recent years [RaGo75], [DCML82], [WMCS82], [HKMT83], [TIIK85], [AlFe87], [SiHa87].

A fundamental contribution to the design of parallel integer multipliers has been made by Wallace [Wall63] who proposed a circuit known as "the Wallace tree multiplier", one of the fastest parallel integer multiplier architectures known today. Many other proposed designs trade off some of the speed for a substantial increase in the regularity of interconnect over the Wallace tree multiplier, thus making them more suitable for VLSI implementation [HaWi70], [RaGo75], [Wase78], [Hwan79], [Kuck81], [ReKM81], [HaRP82].

The design of a multiplier which can perform the computation in time proportional to the logarithm of the number of bits of the operands and has desirable layout charac-

teristics for VLSI implementation has become one of the challenging problems in VLSI design.

Before the detailed design of the Floating Point Multiplier system which was developed in the course of this research work is presented, the most common designs of parallel integer multipliers are reviewed and their VLSI realizations are described. The study of existing designs has led to the development of two novel implementations of parallel integer multiplier having equal performances (over a limited operand lengths) and higher regularity of module interconnect compared to the Wallace tree multiplier.

The usual form, described in [Flor63], [Hwan79], [Kuck81] and [WaFl82], of the floating point multiplication algorithm is analysed and modified to enable more modules in the Floating Point Multiplier system to operate in parallel than is the case in the conventional hardware realizations of this function [WaMC82], [WoLO83], [LeCT84], [WiWi84], [WlWW84], [SiOc86], thus further increasing the speed with which floating point multiplication is executed.

Due to limitations in software tools, computer resources and time, a reduced system has been implemented to demonstrate the ideas presented herein. Performance estimations of the complete 32-bit Floating Point Multiplier system based on circuit simulation results of the reduced system and of the critical path of the complete 32-bit system are given in the concluding parts of this chapter.

8.2 A Comparative Study of Parallel Integer Multiplier Designs

The fastest method of integer multiplication is the combinatorial approach [Wase78], [HaRP82]. In this approach partial products are formed simultaneously, after which the result is produced by summing these partial products in some optimum sequence. This is the basis for the implementation of parallel integer multipliers. A hardware parallel multiplier consists, in general, of a two-dimensional array of 1-bit full adder (denoted here Full Adder) modules and is thus described as an array multiplier. It is a memoryless logic net (unlike a parallel-serial multiplier) that only requires a prescribed settling time to elapse, after the application of input signals, before the output signals can be used. There are many different configurations possible, each being classified according to the method of connecting the Full Adders and to the treatment of negative numbers. The main features of array multipliers are a requirement for large silicon area, high power dissipation and a potential for high regularity. Parallel multipliers do not have an

optimum throughput per area of silicon when compared with bit serial or pipelined multipliers [BrKu81]; the summation process of the partial products is "sequential" in nature since only a limited number of Full Adder modules have generated a stable output at any instant of time. Of the various multiplier options, however, they offer the shortest latency time [RaGo75], [BrKu81] and therefore were investigated in this work. Unsigned multipliers are of main interest in this study, as a result of the decision to follow the specifications of the IEEE Standard for Floating Point Arithmetic [IEEE85], but some discussion of signed operands multipliers is also given for completeness.

Parallel multipliers realized in VLSI using Full Adder modules are optimal from an area viewpoint and are also "asymptotically optimal" from the timing viewpoint, ie the partial products can be summed in time $O(\log n)$ [ReKM81], where n is the number of bits in the multiplier and the multiplicand words. Consequently most of the parallel integer multipliers presently implemented in VLSI are realized with Full Adder modules.

In the remainder of this section a number of parallel integer multiplier designs are reviewed as a prelude to the discussion of the two new architectures proposed in this work.

In order to compare the speed of various parallel multiplier configurations module delays are expressed in terms of the quantity τ, as described in [Hwan79], which is the delay of one level of logic, eg a two input NAND gate or a two input NOR gate. Because every logic function can be implemented exclusively with NAND or NOR gates, time delay in a multilevel logic circuit can be measured by the NAND gate levels, or numbers of τ. The delay (t_{AND}) of a two input AND gate is then approximated by $t_{AND} = 2\tau$, a Full Adder module sum delay (t_s), ie the time required for this module to generate the sum bit, is approximated by $t_s = 2\tau$ and the Full Adder module carry delay (t_c), ie the time required for this module to generate the carry bit, can also be approximated by $t_c = 2\tau$.

In digital system design the complexity and the power dissipation of a given logic circuit are normally characterized by the total number of gates (the total gate count) necessary to implement a given circuit. Such an approach can also be used for VLSI circuits, however, the total silicon area occupied by a given module and the aspect ratio of the module's bounding box are of greater significance. The total gate count, in a VLSI module, can be related to the corresponding silicon area, but does not contain any information about the aspect ratio of the module bounding box. The dimensions of the module bounding box, however, represent much more accurately the *effective* silicon area occupied by that module, and the aspect ratio of the bounding box gives a designer some information about how well the module fits into the floorplan of the complete system being designed. For this reason, when comparing different multiplier designs in this

chapter, estimates of the lengths (L_{xxx}) and the widths (W_{xxx}) of the bounding boxes are given, as a measure of the silicon area ($A_{xxx} = L_{xxx} \times W_{xxx}$) requirements, where xxx denotes the name of the module of interest, the length is taken as the horizontal and the width as the vertical dimension in all figures.

8.2.1 Array of Ripple Carry Adders (ARCA)

The conceptually most simple and most regular $n \times n$ bit parallel multiplier is composed of an $n \times n$ array of 1-bit full adders arranged as n rows of n-bit carry propagate adders, which are most commonly implemented as ripple carry adders [RaGo75]. A block diagram and a possible floorplan of this multiplier design, denoted here ARCA multiplier, for a 4-bit multiplicand word (X) and a 4-bit multiplier word (Y) are shown in Figure 8.1(a) and (b), respectively. In such a configuration the partial products are generated by the AND gates at the input of each Full Adder module. Each ripple carry adder takes the output from the ripple carry adder in the row above, accumulates one partial product into the partial sum and passes the result to the ripple carry adder in the row below. The final product is collected at the outputs of the last row ripple carry adder.

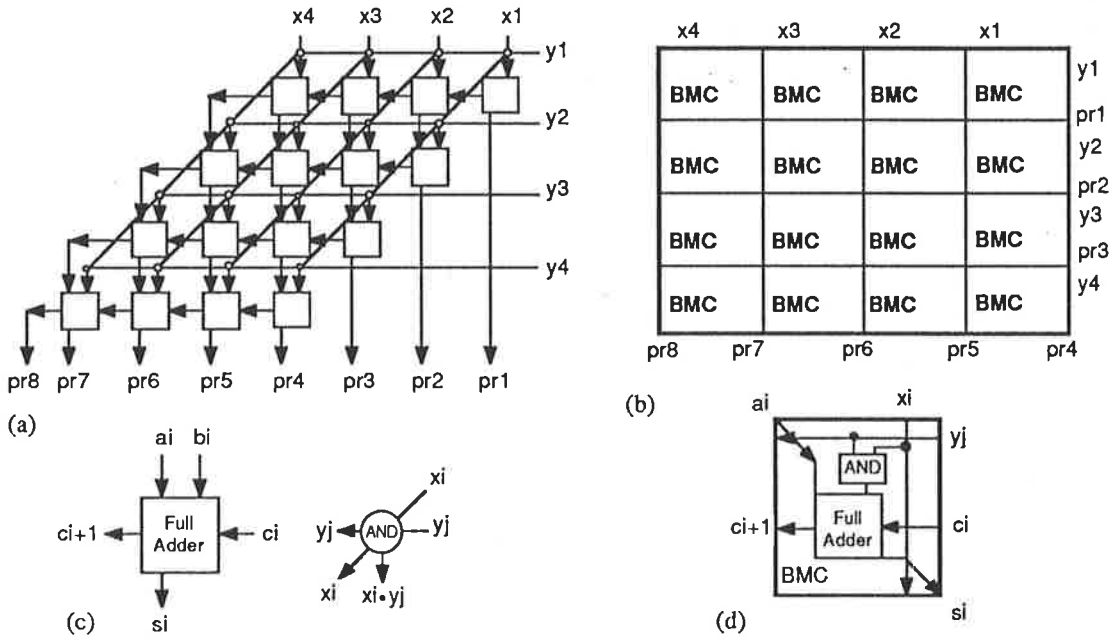


Figure 8.1: ARCA multiplier: (a) block diagram, (b) possible floorplan, (c) definition of symbols in the block diagram, (d) implementation of the BMC module.

The implementation of the ARCA multiplier requires only one type of module, denoted here *Basic Multiplier Cell* (BMC), which is composed, in the case of this multiplier,

of a 1-bit full adder functional module and a two input AND gate functional module, as shown in Figure 8.1(d). (In general, a BMC module is defined as an elementary module used to implement a particular parallel multiplier architecture.) The signal driver modules, required to amplify signals x_i and y_j along the corresponding signal lines, are omitted for all multipliers, from the diagrams, for simplicity. It should be understood that in general such drivers are required for optimum multiplier performance to reduce the propagation time delay of these signals to an acceptable level. The time delays introduced by such driver modules will also be omitted from all timing analyses.

The total settling time of the ARCA multiplier architecture is estimated as follows. The slower input to the Full Adder module generating the sum bit $pr4$ requires the delay of one AND gate, three sum delays, and two carry delays, and in order for the signal to reach $pr7$ an additional delay of three carry delays and one sum delay are necessary. (Note that since the ripple carry adder in the first row has only one input, there is no carry bit propagation and its delay is equivalent to one sum delay.) In a general case of two n-bit operands the total multiplier delay (t_{ARCA}) is given by:

$$\begin{aligned} t_{ARCA} &= t_{AND} + (n-1)t_s + (n-2)t_c + (n-1)t_c + t_s, \\ &= nt_s + (2n-3)t_c + t_{AND}, \\ &= (6n-4)\tau. \end{aligned} \tag{8.1}$$

With reference to the multiplier floorplan of Figure 8.1(b) the silicon area (A_{ARCA}) occupied by this multiplier, expressed in terms of the dimensions of the overall multiplier bounding box as discussed in the previous section, is given by:

$$\begin{aligned} A_{ARCA} &= nL_{BMC} \times nW_{BMC}, \\ &= nL_{FA} \times n(W_{FA} + W_{AND}), \\ &= n^2(A_{FA} + A_{AND}), \end{aligned} \tag{8.2}$$

where FA stands for Full Adder.

The first row ripple carry adder may be eliminated by accumulating the first two partial products in the second row ripple carry adder. Modest savings both in time delay and area are achieved in this way at the expense of increased design complexity, since now two kinds of module (BMC and BMCa) must be designed, as shown for the case of two 4-bit operands in Figure 8.2. The module BMC is, as before, composed of a Full Adder module and a two input AND gate, and the module BMCa is composed of a two input AND gate. In general, the modified implementation requires $n-1$ stages of n-bit ripple carry adders and the multiplier delay becomes:

$$\begin{aligned} t_{ARCA} &= (n-1)t_s + (2n-3)t_c + t_{AND}, \\ &= (6n-6)\tau. \end{aligned} \tag{8.3}$$

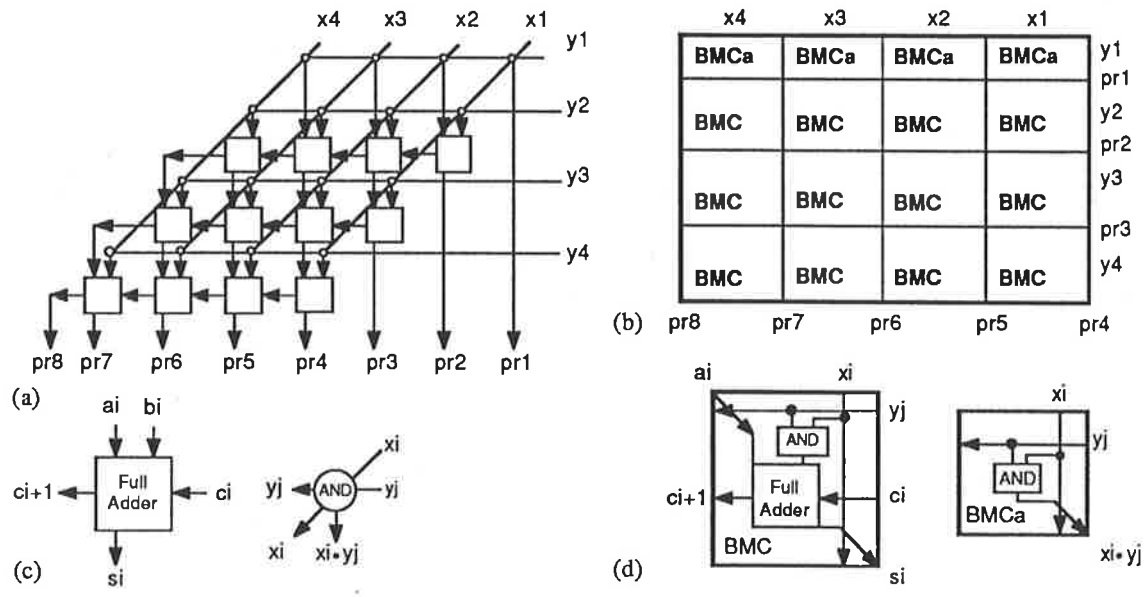


Figure 8.2: ARCA multiplier: (a) modified block diagram, (b) modified floorplan, (c) definition of the block diagram symbols, (d) implementation of the BMC and BMCa modules.

The silicon area of the modified multiplier becomes:

$$\begin{aligned} A_{ARCA} &= nL_{BMC} \times [(n-1)W_{BMC} + W_{BMCa}], \\ &= nL_{FA} \times [(n-1)(W_{FA} + W_{AND}) + W_{AND}], \\ &= n(n-1)A_{FA} + n^2 A_{AND}. \end{aligned} \tag{8.4}$$

8.2.2 A Binary Tree of Ripple Carry Adders (BRCA)

A different arrangement of n-bit ripple carry adders, in which the adders are connected as a binary tree, has been discussed in [RaGo75] and [Kuck81]. The way in which the partial products are combined in an 8-bit version of such a multiplier (BRCA multiplier) is depicted schematically in Figure 8.3. The bits of the partial products are represented by "asterisks" and the ripple carry adders are shown as rectangles.

In this example there are eight 8-bit partial products. Two such partial products, shifted with respect to one another by one bit position, are summed together (reduced) by an 8-bit level-1 adder to form a 9-bit sum. The 9-bit sum is then combined with the 1-bit input, which do not participate in the summation (eg the 1-bit input in bit position 1 of the level-1 adder-1, Figure 8.3) to form a 10-bit result. The individual bits of the four 10-bit addends ($sum_{1,1}$, $sum_{1,2}$, $sum_{1,3}$ and $sum_{1,4}$) formed in this manner are represented by "deltas" in the figure. These bits constitute the inputs to the level-2

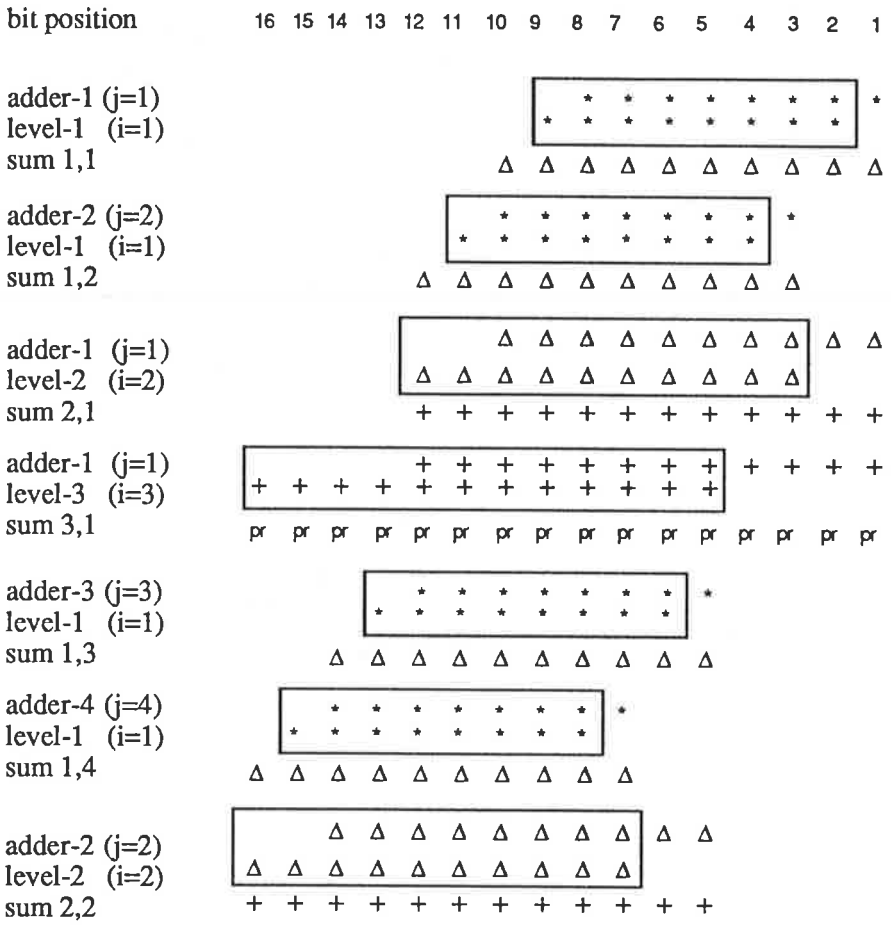


Figure 8.3: Accumulation of partial products with a tree of ripple carry adders.

adders.

There are two level-2 adders, each being 10-bits long. Each adder on level-2 sums two 10-bit addends, shifted with respect to one another by two bit positions, to form a 10-bit sum. Each of the two adder outputs is combined with the two 1-bit inputs, which do not participate in the summation at level-2, to form a 12-bit result. The individual bits of the two 12-bit addends produced in this way ($sum_{2,1}$ and $sum_{2,2}$) are represented in the figure by the "plus signs".

A 12-bit level-3 adder combines two 12-bit addends, generated by the level-2 adders, to form a 12-bit sum. The two 12-bit words are shifted with respect to one another by four bit positions. This sum is combined with the remaining four 1-bit inputs, which do not participate in the summation at level-3, into the final 16-bit product ($sum_{3,1}$) whose individual bits are represented by the letters "pr" in the figure.

The settling time of the BRCA multiplier can be estimated in the following manner. With reference to Figure 8.3, level-2 adders start adding after an initial delay composed of one sum and one carry delays, ie $t_s + t_c$. Level-3 adders commence addition after an

additional delay of $t_s + 2t_c$. In general, the delay between commencement of addition on level-i, and commencement of addition on level-$(i+1)$ is $t_s + 2^{i-1}t_c$. The maximum length of the carry bit path at the final level of summation is $3n/2 - 1$ bits. The total multiplier delay (t_{BRCA}) is then given by:

$$\begin{aligned} t_{BRCA} &= \sum_{i=1}^{i=\log_2 n-1} \left(t_s + 2^{i-1}t_c\right) + \left(\frac{3n}{2} - 1\right) t_c + t_s + t_{AND}, \\ &= (\log_2 n)t_s + (2n-2)t_c + t_{AND}, \\ &= (4n + \log_2 n^2 - 2)\tau. \end{aligned} \tag{8.5}$$

The BRCA multiplier architecture provides the biggest speed improvement over the ARCA multiplier if the carry signal delay of the Full Adder module used is substantially smaller than the corresponding sum signal delay.

The main difficulty in obtaining a space efficient implementation of this multiplier stems from the fact that the adders are of different lengths at successive levels. A possible floorplan is given in Figure 8.4. In this floorplan the length of overall bounding box is determined by the length of the final level adder. The length of the final level adder, in a general case, is determined as follows.

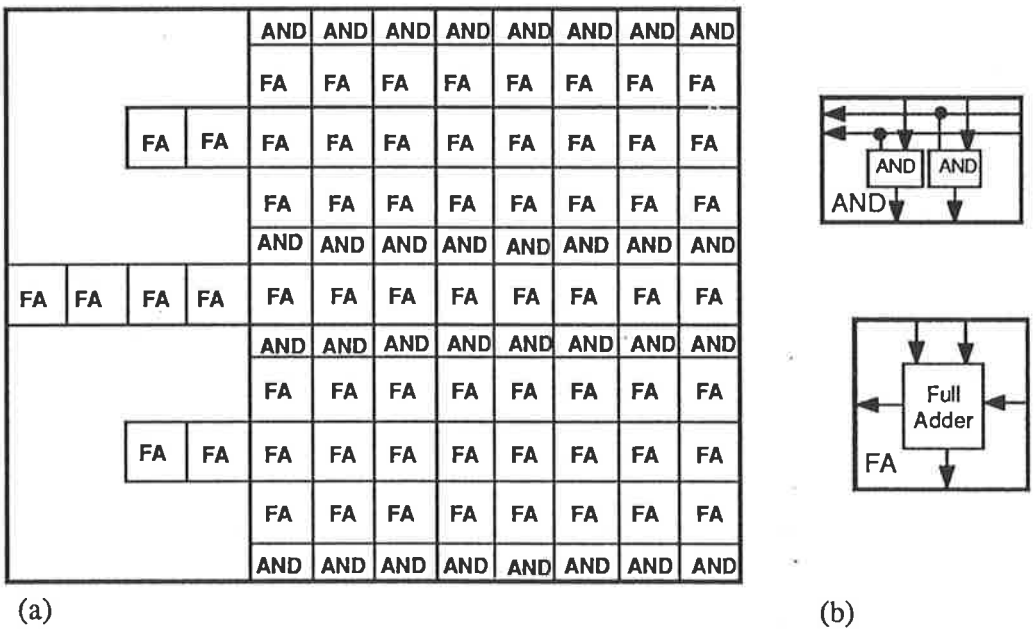


Figure 8.4: BRCA multiplier: (a) possible floorplan, (b) implementation of the FA and AND modules.

At the first level of partial product summation each adder adds two n-bit numbers to produce at most an $(n+2)$-bit sum. Since at the least significant bit position there is only one input, all level-1 adders need only be n-bits long, with the most significant bit of the sum corresponding to the carry bit out from the most significant bit position.

Each level-2 adder combines two $(n+2)$-bit addends into an $(n+4)$-bit result. The two numbers being added are shifted with respect to each other by two bit positions, ie the maximum value of the larger input is four times the maximum value of the smaller input. Therefore, the length of any level-2 adder need only be $(n+2)$ bits.

Proceeding in this manner, it can be shown that any level-3 adder need only be $(n+4)$ bits long and in general any level-i, where $i = 2 \ldots \log_2 n$, adder need only be $(n+2^{i-1})$ bits long (note that this result is incorrectly given in [Kuck81] as $n+2^{i-1}+i-2$). Using this result the length of the final level adder ($i = \log_2 n$) is $n + 2^{i-1} = (3n/2)$ bits.

Assuming the floorplan given in Figure 8.4 the effective silicon area (A_{BRCA}) occupied by the BRCA multiplier is given by (note, the number of ripple carry adders at any level $i = 1, \ldots, \log_2 n$ is given by $n/2^i$):

$$\begin{aligned} A_{BRCA} &= \frac{3}{2} n L_{FA} \times \left(\sum_{i=1}^{i=\log_2 n} \frac{n}{2^i} W_{FA} + n W_{AND} \right), \\ &= \frac{3}{2} n [(n-1) A_{FA} + n A_{AND}]. \end{aligned} \tag{8.6}$$

The result given by expression (8.6) may be reduced by replacing the FA modules in the most significant bit positions of all ripple carry adders at levels $i > 1$ by a two input OR gate. This is possible because if there is a carry into the most significant bit position within adder-j ($j = 1, 2 \ldots, n/2^i$), for an even value of j, on level-i (Figure 8.3) then there will not be a carry into the most significant bit position within adder-$j/2$ on level-$(i+1)$. Thus, the two carry bits are mutually exclusive and they can be combined by a two input OR gate. This result will first be demonstrated by an example and then extended to a general case.

Consider a 4×8 bit multiplier, ie a 4-bit multiplier and an 8-bit multiplicand. There are, as can be deduced from Figure 8.3, two level-1 adders ($i = 1, j = 1, 2$) each being 8 bits long and one level-2 adder ($i = 2, j = 1$) being 10 bits long. The maximum result, denoted $sum_{i,j_{max}}$, that can be generated by the level-1 adder-2 ($i = 1, j = 2$), assuming unsigned operands is given by:

$$\begin{aligned} sum_{1,2_{max}} &= 4(2^8 - 1) + 8(2^8 - 1), \\ &= 12(2^8 - 1). \end{aligned} \tag{8.7}$$

According to expression (8.7) the output of the level-1 adder-2 can overflow, ie it can have a 1 in bit position 12. However, if the output of the level-2 adder-1 ($i = 2, j = 1$) is to overflow, ie have a one generated in bit position 13, the input of this adder, which input corresponds to the output $sum_{1,2}$, must be such that:

$$sum_{1,2} \geq 2^{11} + 2^{10},$$

$$= 12 \times 2^8,$$

$$> sum_{1,2_{max}}, \tag{8.8}$$

which is larger than the maximum possible value given in expression (8.7). According to this result the output of the level-2 adder-1 never overflows, ie an output with a 1 in bit position 13 does not occur.

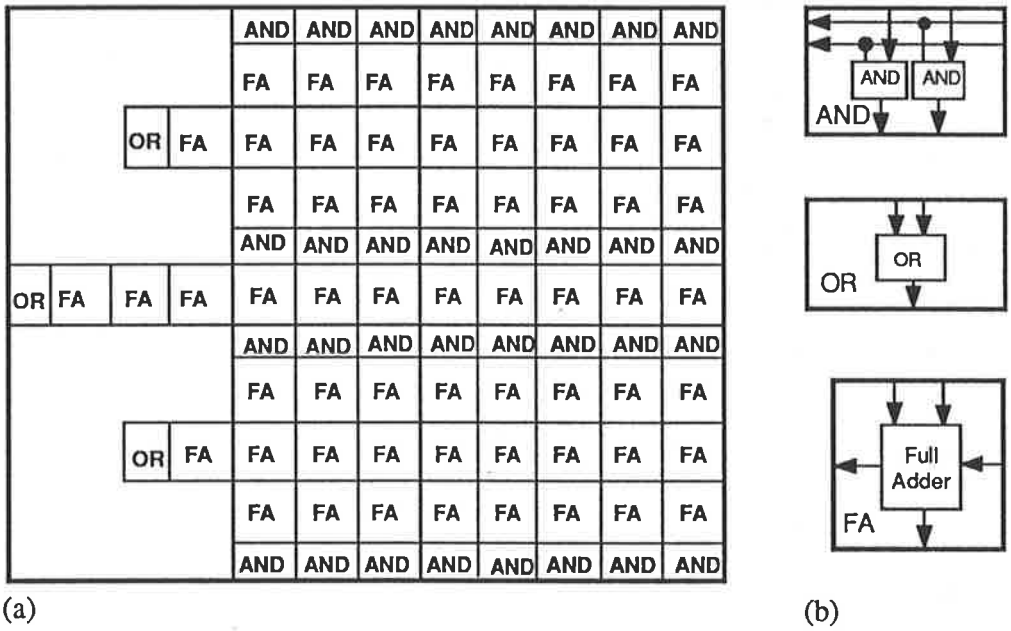


Figure 8.5: BRCA multiplier: (a) modified floorplan, (b) implementation of the functional modules FA, AND and OR.

In general, for an overflowed output $sum_{i,j}$ from adder-j, (j even) on level-i to produce an overflow within adder-$j/2$ on level-$i+1$, the output $sum_{i,j}$ must have the value given by:

$$\begin{aligned} sum_{i,j} &\geq 2^{(j-1)2^i+n}\left(2^{2^i}-1\right), \\ &= 2^n 2^{(j-1)2^i}(2^{2^i}-1). \end{aligned} \tag{8.9}$$

The maximum sum that can be generated by such an adder, however, is given by:

$$\begin{aligned} sum_{i,j_{max}} &= (2^n-1)\Sigma_{k=0}^{2^i-1} 2^{k+(j-1)2^i}, \\ &= (2^n-1)2^{(j-1)2^i}(2^{2^i}-1). \end{aligned} \tag{8.10}$$

which is less than the value given in expression (8.9).

Using the above result the length of the BRCA multiplier bounding box is reduced to $L_{BRCA} = [(3n/2)-1]L_{FA} + L_{OR}$. A possible floorplan for the modified BRCA multiplier is given in Figure 8.5(a).

This small decrease in the silicon area is not worth the additional design effort of having to introduce an extra functional module, and in general this multiplier would be implemented according to the floorplan of Figure 8.4. The above results is interesting, however, because it gives the maximum adder lengths at any level of partial product summation.

8.2.3 Array of Carry Save Adders (ACSA)

A substantial speed increase over the multiplier designs implemented with ripple carry adders (ie ARCA and BRCA multiplies) is obtained by postponing the assimilation of carries to the last row of adders [Brau63]. In the resulting multiplier architecture (called "carry save" configuration and denoted here ACSA) the interconnection of the Full Adder modules is such that the sum and the carry signals propagate simultaneously toward the higher level adders, as shown in Figure 8.6 for the case of 4-bit operands. The 1-bit full adder used is this way has been called the carry save adder (CSA).

In general, the ACSA multiplier is implemented with an $n \times n$ array of a basic multiplier cells (BMC) each composed of a 1-bit full adder and a two input AND gate, and a row of n 1-bit full adders arranged as an n-bit ripple carry adder.

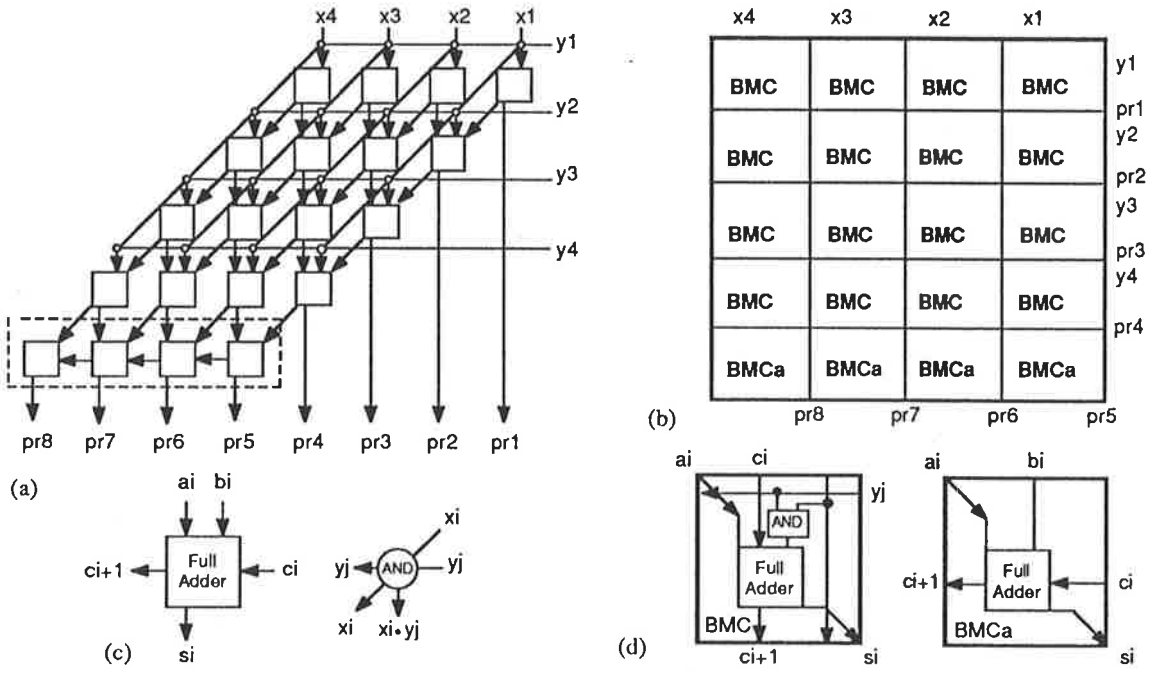


Figure 8.6: ACSA multiplier: (a) block diagram, (b) floorplan, (c) symbol definition, (d) implementation of the modules BMC and BMCa.

Tracing the worst case delay path along the fourth column (from the right) and the

bottom row of the array, Figure 8.6(a), one obtains the total multiplier delay (t_{ACSA}):

$$\begin{aligned} t_{ACSA} &= t_{AND} + (n-1)t_s + t_c + (n-1)t_c + t_s, \\ &= nt_s + nt_c + t_{AND}, \\ &= (4n+2)\tau. \end{aligned} \tag{8.11}$$

The silicon area (A_{ACSA}) occupied by the ACSA multiplier is given by:

$$\begin{aligned} A_{ACSA} &= nL_{FA} \times [(n+1)W_{FA} + nW_{AND}], \\ &= n(n+1)A_{FA} + n^2 A_{AND}. \end{aligned} \tag{8.12}$$

This configuration can be improved upon by combining the first two partial products in the second row of the Full Adder modules, thus eliminating the Full Adder modules from the first row of BMC modules in the array. In addition the leftmost column of the Full Adder modules can also be eliminated, as shown in the block diagram of the modified ACSA multiplier of Figure 8.7(a). This is the form in which this multiplier was first proposed by Braun [Brau63].

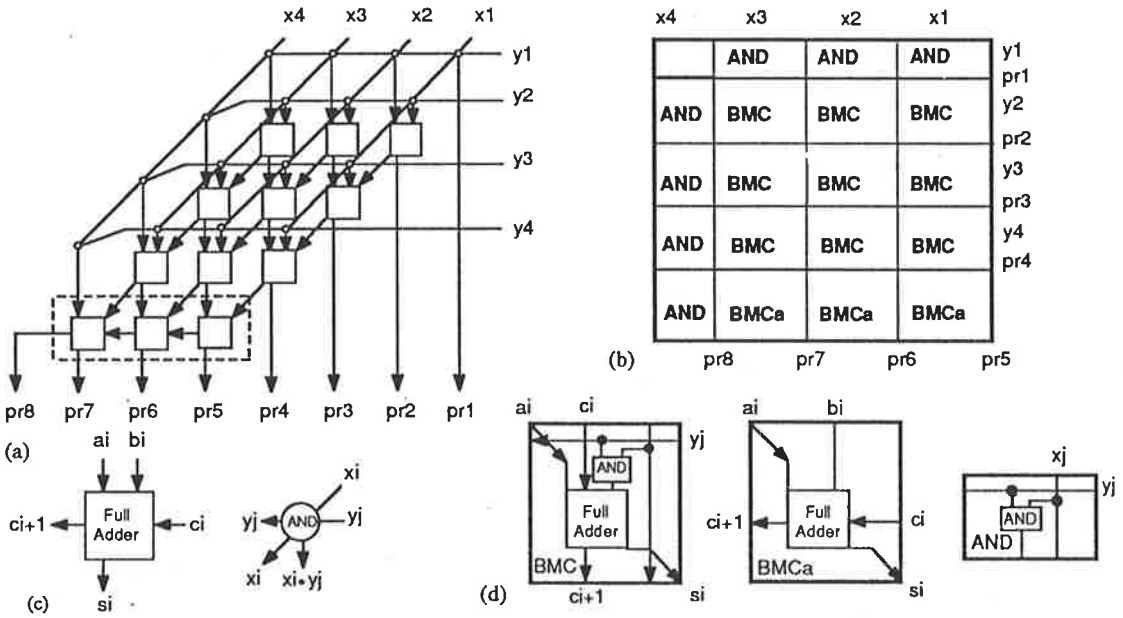


Figure 8.7: ACSA multiplier (original form [Brau63]): (a) block diagram, (b) floorplan, (c) block diagram symbol definition, (d) implementation of the modules BMC, BMCa and AND.

The time delay is reduced to:

$$\begin{aligned} t_{ACSA} &= (n-1)t_s + (n-1)t_c + t_{AND}, \\ &= (4n-2)\tau. \end{aligned} \tag{8.13}$$

and the dimensions of the bounding box of the modified multiplier architecture are reduced to:

$$\begin{aligned} A_{ACSA} &= [(n-1)L_{FA} + L_{AND}] \times n(W_{FA} + W_{AND}), \\ &= n(n-1)A_{FA} + n(n+1)A_{AND}. \end{aligned} \tag{8.14}$$

These improvements in silicon area and speed are achieved at the expense of increased design complexity, with three modules (AND, BMC and BMCa) being needed to implement this form of the ACSA multiplier, and the module interconnection on the left hand side of the array being different from the interconnection pattern in the remainder of the array, as shown in Figure 8.7(a). This is, however, a negligible increase in design effort and the ACSA multiplier is commonly implemented in this way.

An important aspect of this particular integer multiplier implementation is that it has become a standard design used as a benchmark circuit for new technology demonstrations, eg [HAMM87] and [LeGS87].

The main advantage of postponing the assimilation of the carry bits to the last row of adders is that it is now economical to replace the Full Adder modules connected as the ripple carry adder by an adder with a fast carry propagate arrangement. The delay of ACSA multiplier then becomes:

$$\begin{aligned} t_{ACSA} &= (n-2)t_s + t_c + t_{AND} + t_{CPA,n-1}, \\ &= 2n\tau + t_{CPA,n-1}, \end{aligned} \tag{8.15}$$

where $t_{CPA,n-1}$ is the delay of a fast carry propagate adder of length $(n-1)$ bits. This delay can be made approximately constant over a range of values of n of interest by an application of the carry look-ahead, or carry bypass techniques [Hwan79]. Expression (8.15) is thus left as a sum of two components: one being a function of n and the other remaining approximately constant. This permits time delay comparisons of different multiplier designs on the basis of the number of stages of the Full Adder modules (where the delay of one stage of Full Adder is $= 2\tau$).

In the carry save approach it is important that the delays t_s and t_c of the Full Adder module be equal and as small as possible, otherwise the slower signal will limit the overall multiplier performance. It is assumed in this work that only such Full Adder modules are employed.

Pezaris [Peza71] suggested a modification to the interconnection of the Full Adders modules used in the ACSA multiplier which resulted in an increased speed of propagation of the sum signals. In that multiplier design the longest path for any sum signal includes only $n/3$ Full Adder modules compared to $n-1$ in the ACSA multiplier. This reduction

in the length of the sum signal path is achieved by a "sum skip" arrangement, where the sum signals "jump" over two rows of Full Adder modules before being added to the next partial product. The carry signals still propagate as in the ACSA multiplier.

Figure 8.8(a) shows a block diagram of a similar arrangement (for a 6×6 bit multiplier) in which the sum signals "jump" over one row (rather than two rows as is the case in the design proposed by Pezaris) of Full Adder modules before being added to the next partial product, reducing the critical path for any sum signal to $n/2 = 3$ stages of the Full Adder module. This figure presents the idea proposed by Pezaris more clearly and can be easily extended to the "sum skip" arrangement over any number of rows. The individual bits, p_{ij}, of the partial products included inside a given box (Figure 8.8(b)) representing a Full Adder module or immediately above it, are summed by that adder. The adder modules marked with an "X" do not have a bit of partial product as one of their inputs.

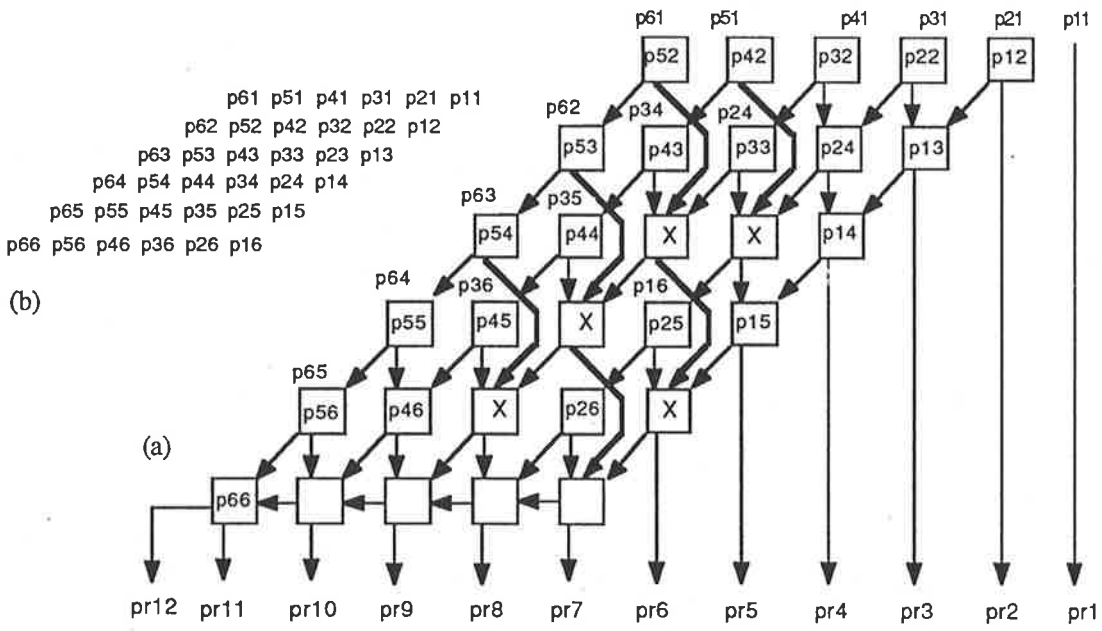


Figure 8.8: ACSA multiplier incorporating the "sum skip" idea [Peza71]: (a) block diagram, (b) matrix of partial products for the 6×6 case.

The advantage of this particular design is that if a Full Adder module is available with $t_s > t_c$ then using this method the longest path for the sum signal may be made of such a length that the total sum and carry signals delays inside the array can be made approximately equal. The module interconnection of the scheme of [Peza71] is, however, less regular than in the ACSA multiplier of Figure 8.6 or 8.7 because of an irregular distribution of the individual bits of partial products throughout the array.

8.2.4 Wallace Tree Multiplier (WT)

The idea of connecting ripple carry adders as a tree (BRCA multiplier) may also be extended to the carry save approach. Wallace has proposed a tree arrangement of Full Adder modules [Wall63], [Hawi70], [Wase78], [ReKM81].

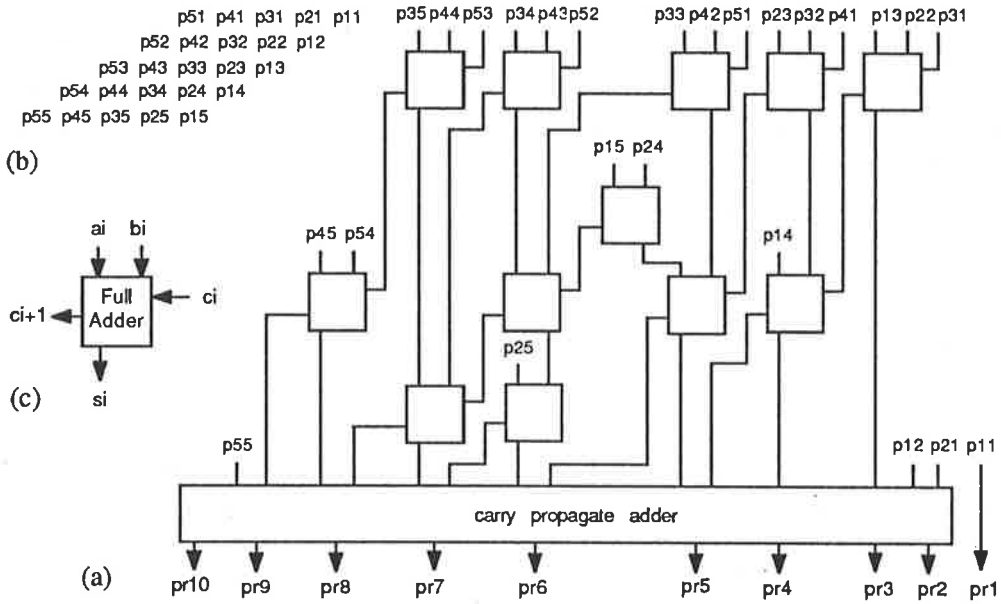


Figure 8.9: The Wallace tree multiplier: (a) block diagram for a 5×5 bit case (after [HaWi70]), (b) the corresponding matrix of partial products, (c) module definition.

The Wallace tree multiplier uses strings of Full Adders to reduce the initial n partial products to $2/3n$ addends at the first level of adders, to $(2/3)^2 n$ addends at the second level of adders etc. This process is repeated until only two addends remain whose sum is equal to the required product of the operands. The two addends are reduced to the final product by a fast carry propagate adder such as a carry look ahead or a carry bypass adder. A block diagram showing the interconnection of Full Adder modules for a 5×5 bit Wallace tree multiplier is given in Figure 8.9.

In general, for a Wallace tree multiplier, the number of Full Adder delays to reduce n partial products to two addends is $(\log_{1.5} n) - 1$. The tree of Full Adder modules is followed by a fast carry propagate adder of length $2n - \log_{1.5} n - 2$ bits. The multiplier delay (t_{WT}) is given by:

$$\begin{aligned} t_{WT} &= t_{AND} + (\log_{1.5} n - 1) t_{FA} + t_{CPA,2n-\log_{1.5} n-2}, \\ &= 2\tau \log_{1.5} n + t_{CPA,2n-\log_{1.5} n-2}, \end{aligned} \tag{8.16}$$

where $t_{FA} = \max$ of (t_s, t_c), and $t_{CPA,2n-\log_{1.5} n-2}$ is the delay of a fast carry propagate

adder of length $2n - \log_{1.5} n$ bits. The main significance of this multiplier architecture is that the delay depends logarithmically on n, rather than linearly as was the case with the previous designs, however, the final carry propagate adder is longer than in the ACSA multiplier.

Although very fast it is a very difficult architecture to implement in silicon due to its irregular interconnection of modules. A semi-regular layout for a 24-bit version of the Wallace tree multiplier also incorporating the modified Booth algorithm [Rubi75] has been presented in [ReKM81] for a technology with at least two layers of metal interconnect. The extra wires required for the Wallace trees are routed above the adders, on the second level of metal, thus not consuming any silicon area.

Dadda [HaWi70] has proposed a similar tree of Full Adder modules with the same time and area complexity as the Wallace tree.

8.2.5 The Modified Booth Algorithm Multiplier (MBA)

Another method of increasing the speed of a parallel multiplier is to reduce the number of partial products to be added together rather than to devise a fast method of summing the n partial products formed in the usual way.

The modified Booth algorithm [Rubi75] is an encoding technique that reduces the number of partial products to be summed to $n/2$, for n even and to $(n+1)/2$ for n odd. Thus, approximately half as many stages of Full Adders, compared to the ACSA multiplier, are required to form the final product and a considerable speed advantage is achieved.

This algorithm has been derived from the original Booth algorithm [Boot51] that allows the multiplication operation to bypass a string of ones or zeros in the multiplier word rather than form a partial product for each bit of the multiplier word. The original Booth algorithm and the modified Booth algorithm provide a very elegant method of multiplying two two's complement numbers in which all bits of the partial product matrix are treated in the same way.

In the modified Booth algorithm the partial products are formed in the following way [Wase78], [Hwan79], [Kuck81], [WaFl82]. For unsigned operation the n-bit multiplicand (X) is padded with two zeros to the left of its most significant bit or for two's complement operation the multiplicand is sign extended by one place. The n-bit multiplier word (Y) is padded with one zero to the right of its least significant bit. For unsigned numbers the n-bit multiplier word is also padded with one zero to the left of its most significant bit if

Successive multiplier bits			Form of the *ith* partial product
y_{i+1}	y_i	y_{i-1}	
0	0	0	$+0$
0	0	1	$+X$
0	1	0	$+X$
0	1	1	$+2X$
1	0	0	$-2X$
1	0	1	$-X$
1	1	0	$-X$
1	1	1	-0

Table 8.1: Forms of partial products in the modified Booth's algorithm

n is odd, or with two zeros if n is even. The multiplier word, modified in this way, is then subdivided into groups of three bits, y_{i+1},y_i and y_{i-1}, $i = 1, \ldots, n$, with adjacent groups sharing a common bit. Each group of three bits is examined and one partial product per such group is formed from the multiplicand according to the encoding scheme given in Table 8.1. If n is odd $(n+1)/2$ such partial products are formed, and if n is even then the number of partial products to be summed is $(n/2)+1$ (if the multiplication is performed on two's complement operands only, then the number of partial products is $n/2$ for n even or $(n/2)+1$ for n odd). Any two successive partial products are shifted by two places with respect to each other, rather than by one place as was the case in the ACSA multiplier.

The reduction in the number of carry save adders is achieved at the expense of needing to design two additional functional modules: a Booth Encoder module (BEN) for encoding a group of three multiplier word bits, and a Partial Product Generator module (PPG). A block diagram showing the structure of the MBA multiplier performing multiplication of two 7-bit unsigned integers is given in Figure 8.10, with a possible floorplan given in Figure 8.11. The major part of the floorplan is composed of an array of $(n+1)/2 = 4$ rows by $n+2$ columns of the basic multiplier cell. Each row in the array contains a BEN module, whose function it is to generate a number of control signals for the PPG modules in that row. The PPG modules produce the partial products, each of which constitutes one of the inputs to the FA (Full Adder) modules in that row.

The first two partial products can be summed by the first row of adders, thus the first row of modules of the array need contain PPG modules only. The remaining rows contain both the FA and the PPG modules. The FA modules of one row operate on the outputs of the PPG modules of that row, the sum outputs of the FA modules from the previous row shifted by two bit positions to the left, and the carry outputs of the FA modules from the previous row shifted one bit position to the left.

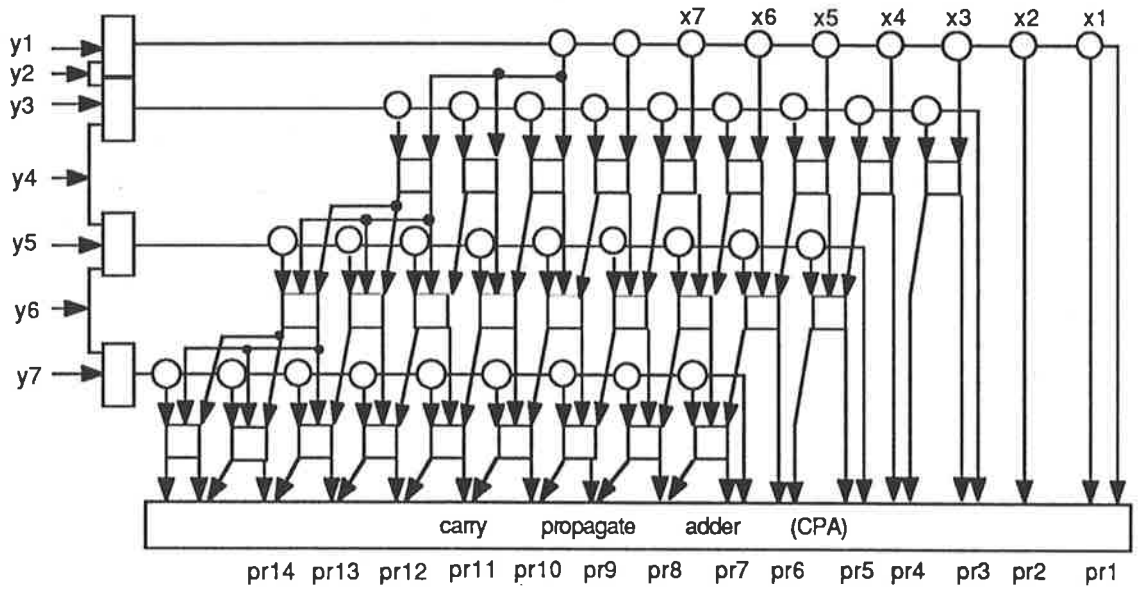


Figure 8.10: MBA Multiplier: block diagram. The rectangles represent the Booth Encoder modules (BEN), the circles represent the Partial Product Generator modules (PPG) and the squares represent Full Adder modules (FA).

For correct operation the partial sum must be sign extended at each level of summation, as discussed above. This causes the fan-out of the FA modules in the leftmost column to be three times as much as the fan-out of the remaining FA modules. In order to equalize the delays more load driving capability is introduced into these modules.

In order to estimate the delay of the MBA multiplier the following approximations for the delays of the various modules are made: the encoding operation can be performed with a delay (t_{BEN}) of two logic gates, and it also takes approximately two gate delays (t_{PPG}) to form a partial product. The delay (t_{MBA}) of the MBA multiplier for unsigned n-bit (n odd) operands can then be approximately given by:

$$\begin{aligned} t_{MBA} &= t_{BEN} + t_{PPG} + [(n+1)/2]t_s + t_{CPA,n}, \\ &= 2\tau + 2\tau + (n+1)\tau + t_{CPA,n}, \\ &= (n+5)\tau + t_{CPA,n}, \end{aligned} \tag{8.17}$$

and for n even:

$$\begin{aligned} t_{MBA} &= t_{BEN} + t_{PPG} + (n/2+1)t_s + t_{CPA,n}, \\ &= (n+6)\tau + t_{CPA,n}, \end{aligned} \tag{8.18}$$

The dimensions of the multiplier bounding box are given as follows (for unsigned

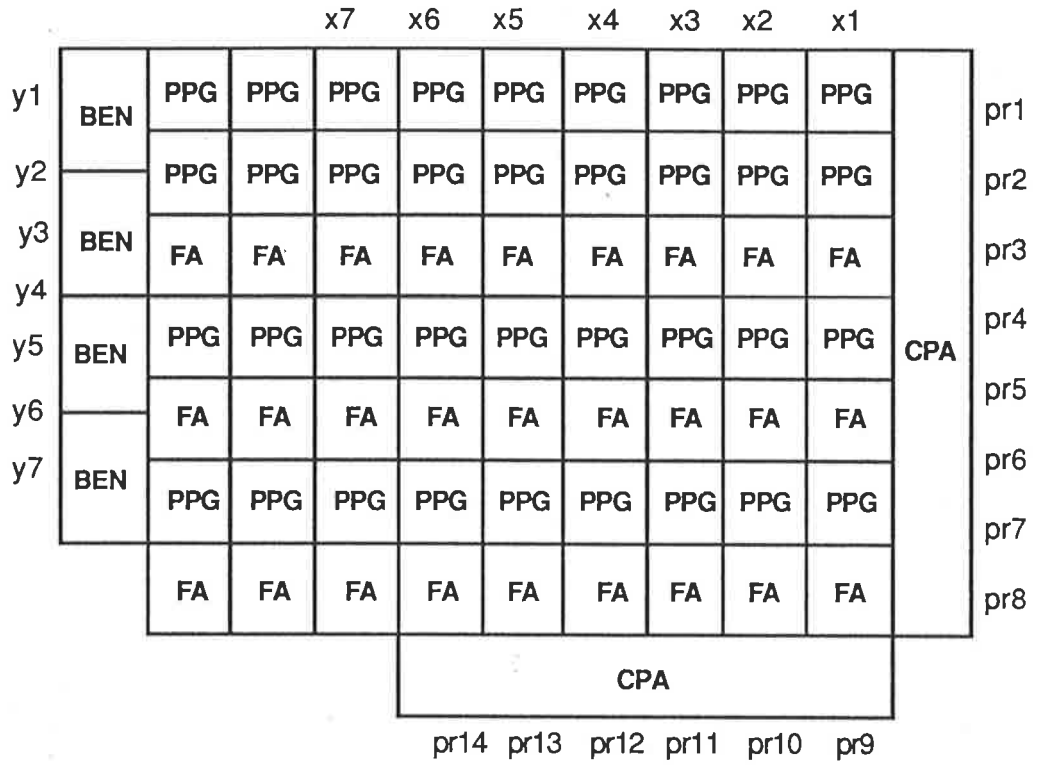


Figure 8.11: MBA multiplier: floorplan.

operands only). If n is odd:

$$\begin{aligned} A_{MBA} &= [(n+2)L_{FA} + L_{BEN}] \\ &\times \left[\frac{(n-1)}{2}W_{FA} + \frac{(n+1)}{2}W_{PPG}\right] + A_{CPA,2n}, \\ &= (n+3)L_{FA} \times nW_{FA} + A_{CPA,2n}, \\ &= n(n+3)A_{FA} + A_{CPA,2n}, \end{aligned} \tag{8.19}$$

where it is assumed that $L_{FA} = L_{BEN}$ and $W_{FA} = W_{PPG}$. If n is even the above result is given by:

$$\begin{aligned} A_{MBA} &= [(n+2)L_{FA} + L_{BEN}] \\ &\times \left[\frac{n}{2}W_{FA} + \left(\frac{n}{2}+1\right)W_{PPG}\right] + A_{CPA,2n}, \\ &= (n+3)W_{FA} \times (n+1)L_{FA} + A_{CPA,2n}, \\ &= (n+3)(n+1)A_{FA} + A_{CPA,2n}. \end{aligned} \tag{8.20}$$

A comparison of expressions (8.15) and (8.14) for the ACSA multiplier and expressions (8.17) and (8.19) shows that the speed increase of the MBA multiplier over the ACSA multiplier is achieved at the expense of increased design complexity (more modules need to be designed) and an increase in silicon area.

A parallel multiplier realizing the octal version of Booth's algorithm has been proposed

by [ReKM81]. The authors reported a 25% speed improvement and a 15% silicon area improvement over a parallel multiplier implementing the modified Booth algorithm. This saving is achieved at the expense of increased design complexity of the BEN and PPG modules, and an increase in module interconnect complexity.

8.2.6 Multipliers for Signed Binary Numbers

The design of array multipliers, such as the ACSA multiplier, becomes more difficult for operands in two's or one's complement form because the signs of the multiplier and the multiplicand are embedded in the representation of the operands.

A number of direct two's complement multiplier structures have been proposed, where the modules used are either more complex or the number of cells is significantly larger than the corresponding unsigned number multiplier array [MaKi71], [GiGi83]. Two schemes that reduce these problems are the array proposed by Pezaris [Peza71] and an array implementing the Baugh-Wooley algorithm [BaWo73].

The Pezaris multiplier, as originally proposed in [Peza74], uses two types of Full Adder modules. However, a multiplier based on the idea of Pezaris, but using only one type of Full Adder module, and structurally equivalent to the ACSA multiplier has been proposed in [BaVH83]. The implementation of the Baugh-Wooley algorithm on the other hand, although it requires only one type of Full Adder, requires the true and complemented forms of both operands [Hwan79].

The parallel multiplier based on the modified Booth algorithm remains, despite the need to design three different functional modules for its implementation, one of the most widely used architectures for the multiplication of two integer operands in two's complement notation.

8.3 New Parallel Integer Multiplier Designs

As a result of the study of parallel multiplier design discussed in the previous section two new parallel integer multiplier architectures have been developed by the author of this thesis. The first multiplier, denoted herein the BCSA multiplier, is an extension of the multiplier design proposed by Vuillemin [Vuil83]. Its implementation and the proposed modifications are the subject of Section 8.3.1. The second multiplier, denoted herein the DS multiplier, is based on the parallel multiplier architectures proposed in [Peza71] and uses the modified Booth algorithm, and is described in full detail in Section 8.3.2.

8.3.1 Binary Tree of Carry Save Adders (BCSA)

An iterative multiplier has been proposed by Vuillemin [Vuil83] which generates the product of two n-bit operands in time $O(\log_2 n)$. It also has a regular array structure and thus can be easily extended to any arbitrary size. Its implementation is based on the carry save adder principle and a binary tree interconnection of basic multiplier cells. A similar parallel multiplier architecture, to that presented in [Vuil83], is given in [TaYY85] and [HNNT87]. The partial products are also summed in time $O(\log_2 n)$ by an array of basic multiplier cells connected as a binary tree, but redundant binary representation is used in order to postpone carry assimilation to the final level, instead of the more usual carry save approach.

The problem of defining an area efficient floorplan for the BRCA multiplier (binary tree of ripple carry adders multiplier) also exists in the designs of [Vuil83], [TaYY85] and [HNNT87]. This is because the adders at different levels of addend summation are of unequal length, with the lengths of the adders at higher levels being longer, as described in Section 8.2.2. In an attempt to produce the most regular implementation of this multiplier, it is proposed in [Vuil83] that the adders at all levels of addend summation be of equal length of $2n$ bits. In the multiplier presented in [TaYY85] and [HNNT87] the redundant binary adders are of varying lengths, with the level-1 adders being $n+1$ bits long, the level-2 adders being $n+4$ bits long, and in general adders at level-i ($i = 1, \ldots, \log_2 n - 1$) being $n + 2^i$ bits long. The carry propagate adders, producing the final products, are $2n-1$ bits long in both multipliers.

In this work a module interconnect modification is presented which results in a realization of the Vuillemin multiplier containing adders which are only n bits long at all levels of addend summation. This allows for a floorplan containing no unnecessary modules and no "dead" space to be defined, thus saving valuable silicon area. In addition, no speed penalty is incurred as a result of this architectural modification. The new multiplier is denoted here the BCSA multiplier.

Before the proposed multiplier architecture modification and the resulting area saving are discussed, a description of the main architectural features of the original multiplier design of [Vuil83] is given to introduce the most important ideas. An implementation containing minimum length adders at each level of addend summation is assumed, in analogy to the multiplier presented in [TaYY85], since this leads more directly to the new architecture.

The basic multiplier cell (BMC) used in the implementation of the BCSA multiplier (and also of the Vuillemin multiplier [Vuil83]) is shown in Figure 8.12. It is composed of

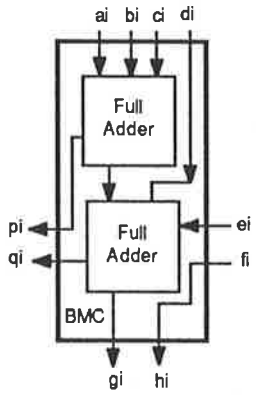


Figure 8.12: BCSA multiplier: implementation of the BMC module.

two Full Adder (FA) modules. The BMC module takes six 1-bit inputs, a_i, b_i, c_i, d_i, e_i and f_i, with significance 2^i and generates two 1-bit outputs, g_i and h_i, with significance 2^i and two 1-bit outputs p_i and q_i with significance $2^{(i+1)}$ such that:

$$2(p_i + q_i) + g_i + h_i = a_i + b_i + c_i + d_i + e_i + f_i. \tag{8.21}$$

The BMC modules implemented in this manner has twice the delay and occupies twice the area of one Full Adder module.

The BCSA multiplier is composed of such BMC modules interconnected as a binary tree $\log_2 n - 1$ levels deep, followed by a $(2n-1)$-bit carry propagate adder to reduce the last two addends to the final result. Figure 8.13 depicts schematically the way in which the product is formed for two 8-bit operands. In the following discussion the general case of two n-bit operands is assumed and a row of x BMC modules is referred to as an x-bit adder.

In the figure, the "stars" represent individual bits of the initial partial products, the "deltas" represent the individual bits of the outputs generated by the level-1 adders and the "plus" signs represent the individual bits of the outputs generated by the level-2 adder. The carry propagate adder is not shown. An empty space indicates that there is no output in that particular bit position. In the following discussion an input is regarded as a 1-bit signal.

Any level-1 adder reduces four n-bit partial products (compare this to the BRCA multiplier where any level-1 adder reduces *two* n-bit partial products) into two addends, each being a maximum of $(n+3)$ bits long. Note that an addend does not necessarily constitute a binary number because in some cases a number of bits are missing. Each level-1 adder needs to be n bits long, as shown in Figure 8.13. For example, the level-1 adder-1 has only one input in bit positions 1 and $n+3$ and two inputs in bit position 2 and thus, no reduction is necessary in these three bit positions. These inputs are di-

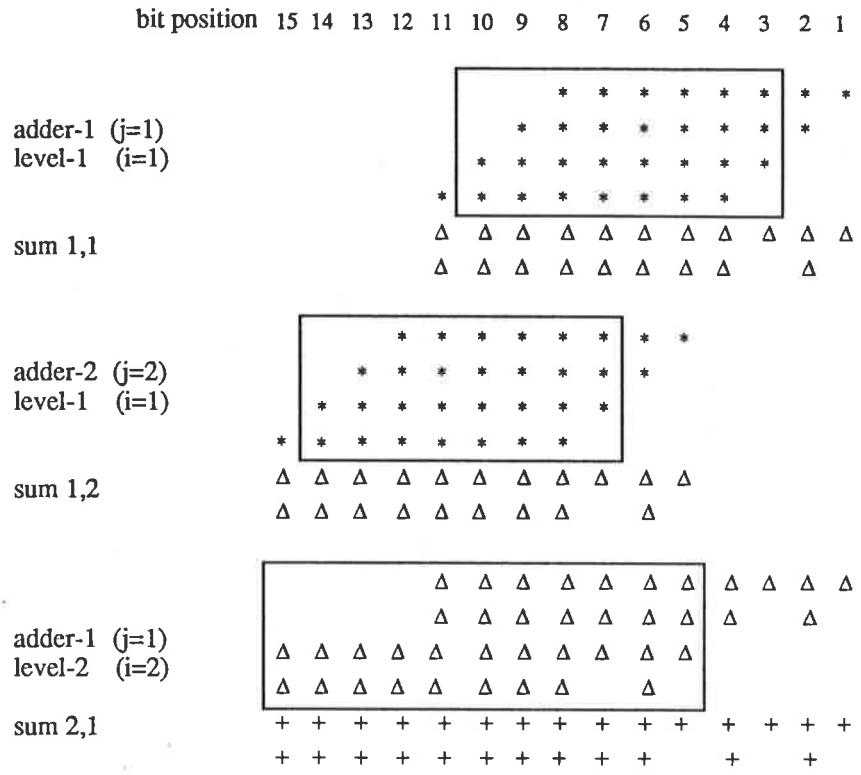


Figure 8.13: BCSA multiplier: method of partial product reduction using a binary tree of BMC modules defined in Figure 8.12, 8 × 8 bit case.

rectly connected to the respective inputs of the level-2 adder-1 or to the carry propagate adder.

At level-2 the successive addends are shifted by four bit positions with respect to each other. In order to reduce four level-2 addends into two level-3 addends $(n+3)$-bit adders are necessary. The results of the second level of reduction are $n/4$ addends, each of $(n+7)$ bits, with the successive addends shifted by eight bit positions with respect to each other. In general there are $n/2^{i+1}$ level-i adders $(i \geq 2)$, each reducing four $(n+2^i-1)$-bit addends into two $(n+2^{i+1}-1)$-bit addends.

In the example of Figure 8.13 there are two level-1 adders, each being 8 bits long, a single level-2 adder 11 bits long, followed by a 15-bit carry propagate adder (not shown in the figure).

The time (t_{BCSA}) required by the BCSA multiplier to compute the $2n$ bits of the product is given by:

$$\begin{aligned} t_{BCSA} &= t_{AND} + 2(\log_2 n - 1)t_{FA} + t_{CPA,3n/2-1}, \\ &= 4(\log_2 n - 1/2)\tau + t_{CPA,3n/2-1}, \end{aligned} \tag{8.22}$$

where, as before, $t_{FA} = \max$ of (t_s, t_c) and $t_{CPA,3n/2-1}$ is the delay of the $(3n/2-1)$-bit

section of the $(2n-1)$-bit carry propagate adder.

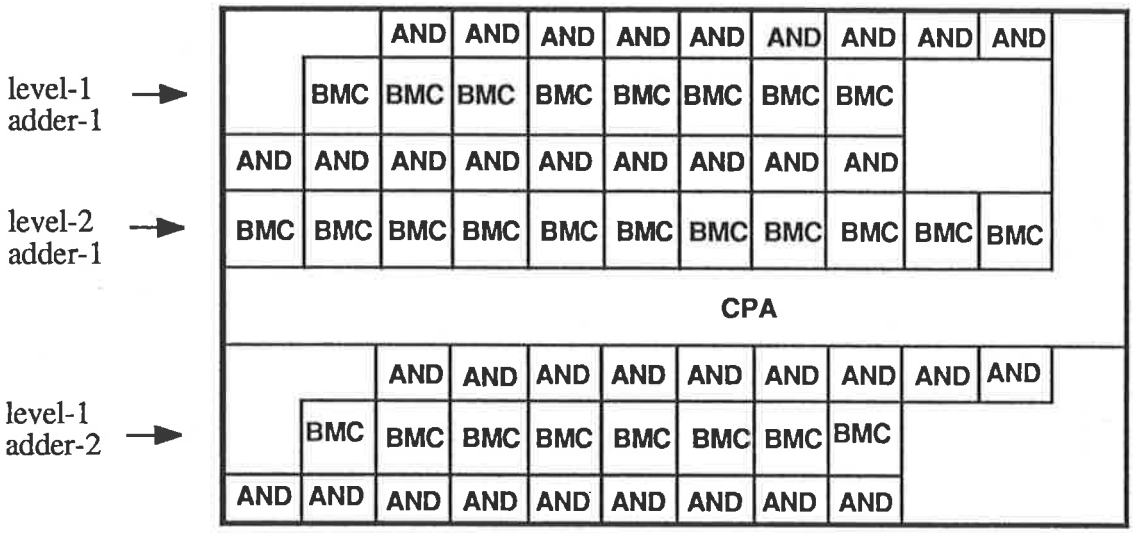


Figure 8.14: Vuillemin multiplier: area optimum floorplan for an 8×8 multiplier. AND denotes two 2-input AND gate modules (see Figure 8.4(b)).

The variable size of the adders at successive levels of addend summation results in an inefficient floorplan for the BCSA multiplier from the silicon area point of view, ie the floorplan contains a lot of "dead" space. A possible floorplan for an 8×8-bit Vuillemin multiplier corresponding to the block diagram of Figure 8.13 is analogous to the floorplan, shown in Figure 8.4, of the BRCA multiplier and it is given in Figure 8.14. The effective silicon area (A_{Vuil}) occupied by this multiplier is given by:

$$\begin{aligned} A_{Vuil} &= \left(\frac{3n}{2}-1\right) L_{FA} \\ &\times \left[2W_{FA}\left(\sum_{i=1}^{i=\log_2 n-1} \frac{n}{2^{i+1}}\right) + nW_{AND}\right] + A_{CPA,2n-1}, \\ &= \left(\frac{3n}{2}-1\right) L_{FA} \\ &\times \left[2W_{FA}\left(\frac{n}{2}-1\right) + nW_{AND}\right] + A_{CPA}, \\ &= \frac{3}{2}\left(n-\frac{2}{3}\right)(n-2)A_{FA} + n\left(\frac{3}{2}n-1\right)A_{AND} + A_{CPA,2n-1}. \end{aligned} \tag{8.23}$$

This is approximately 1.5 times the silicon area required by the ACSA multiplier. The module interconnection required in this implementation is not as regular as in the original Vuillemin multiplier, however, this implementation requires approximately 25% less silicon area, as a result of minimum length adders only being used. The redundant binary tree multiplier presented in [HNNT87] and [YaTT87] was implemented according

to the floorplan of Figure 8.4, however, the adders were $n + 2^i$ bits long at any level $i \geq 2$, rather than being $n + 2^i - 1$ bits long, as demonstrated here.

A modification, proposed in this work, is now presented that results in the implementation of the Vuillemin multiplier that requires the adders, at each level of addend summation, to be of *constant* length of n bits. This multiplier is termed here the BCSA multiplier. It is now possible to produce a floorplan, as shown in Figure 8.16, that does not contain any "dead" space and the maximum width of the overall bounding box is approximately equal to the width of the BMC module times n rather than times $3n/2$. This area reduction is achieved at some cost of interconnect complexity but no additional delay is incurred.

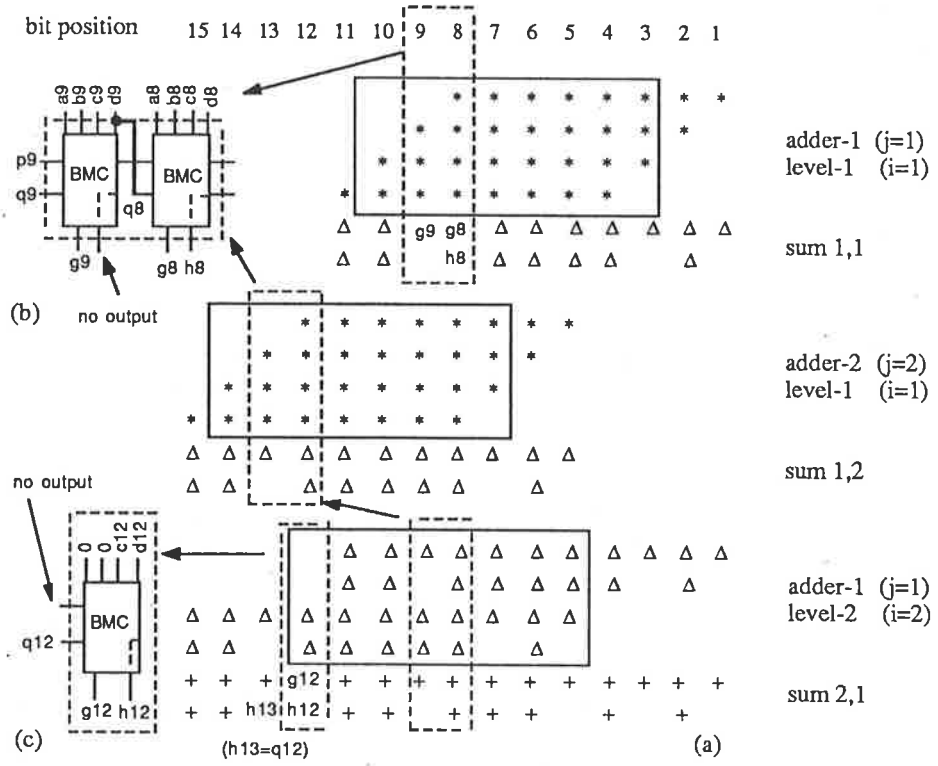


Figure 8.15: BCSA multiplier: (a) reduction of partial products, (b) module interconnection modification in bit position 9 and 13, (c) module outputs in bit position 13.

The reduction in the lengths of the adders at subsequent levels of the binary tree is explained using the diagram of Figure 8.15. Consider the level-1 adder-1. The BMC module in bit position 9 (ie $n+1$ in a general case) accumulates only three 1-bit inputs. If the unused input of this module is connected to the output $q8$ of the BMC module in bit position 8, as shown in Figure 8.15(b), then only one 1-bit output, namely g_9, will be generated in bit position 9. Similarly only one 1-bit output is generated by providing such a connection between the BMC modules in bit positions 12 and 13 (ie $n + 5$ and

$n + 6$ in a general case) of adder-2 on level-1.

The outputs from adder-1 and adder-2 on level-1 constitute inputs to adder-1 on level 2. Since there are a maximum of two inputs to adder-1 on level-2 in bit positions 1 to 4 no addend reduction is necessary for these positions and such inputs are directly connected to the carry propagate adder.

As described above there are only three inputs in bit position 9 $(n + 1)$ of adder-1 on level-2, namely the two outputs from adder-2 on level-1 combined with the single output g_9 of adder-1 on level-1. The unused input of the BMC module in bit position 9 of adder-1 on level-2, is connected to the output q_8 of the BMC module in bit position 8 of adder-1 on level-2, resulting in only one bit of output being generated in bit position 9 $(n + 1)$ of adder-1 on level-2. This modification is not necessary in this case but it would be required for a 16-bit multiplier, for example.

The level-2 adder-1 has only one bit of input in bit position 13 $(n + 5)$, since there is no output in this bit position from adder-1 on level-1, and there is only a single output from adder-2 on level-1, as a result of the modified connection between the BMC modules in bit positions 12 and 13 of adder-2 on level-1. In addition, adder-1 on level-2 has only two inputs in bit position 12 $(n + 4)$. Connecting these inputs into ports c_{12} and d_{12} of the BMC module in bit position 12, as indicated in Figure 8.15(c), ensures that the output p_{12} of this particular module will always $= 0$, ie there will be effectively only one output (q_{12}) into the next most significant bit position (bit position 13 in this case). This single output (q_{12}) combined with the single output g_{13} of adder-2 on level-1 results in only two bits of output in bit position 13. Consequently, there is no need for a BMC module in this bit position of adder-1 on level-2. Similarly, there is no need for a BMC module in bit positions 14 and 15. It follows from this argument that adder-1 on level-2 need only contain BMC modules in bit positions 5 to 12, ie be only 8 bits long.

A similar argument may be presented for any other level-2 or higher level adders which are necessary if $n > 8$, and thus it can be shown that *any* adder at *any* level of addend reduction needs only be n-bit long if the additional connections between adjacent BMC modules in appropriate bit positions are introduced.

There is no additional delay incurred by this scheme. This will now be demonstrated, again using the diagram of Figure 8.15. All outputs from the level-1 adder-1 in bit positions 3 to 8 are available after two Full Adder delays and the outputs in bit positions 9 to 11 are available after three Full Adder delays (ie the modified connection introduces an additional Full Adder delay in these bit positions). Similarly, outputs in bit positions 5 to 12 of adder-2 on level-1 are available after two Full Adder delays and the outputs in

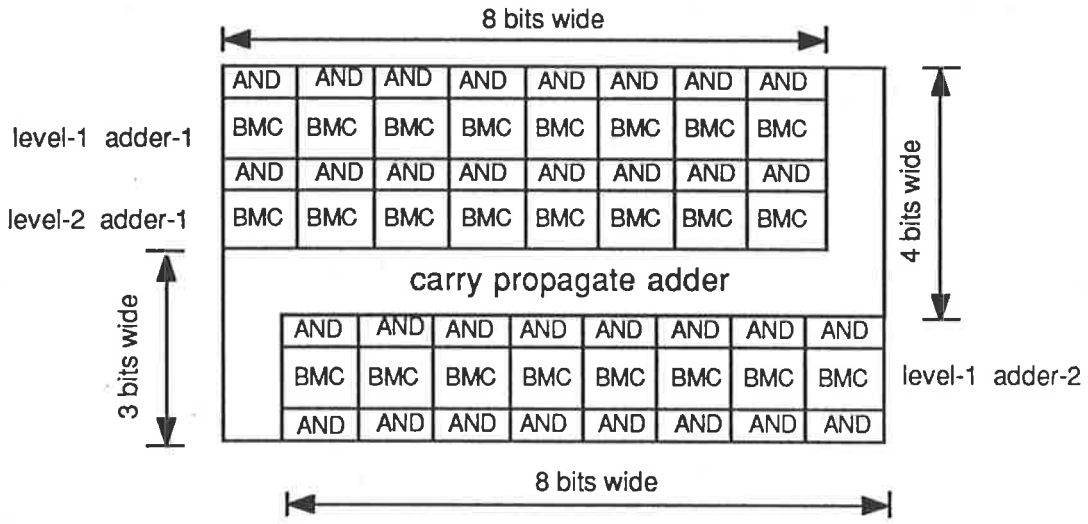


Figure 8.16: BCSA multiplier: floorplan for an 8×8 bit case.

bit positions 13 to 15 need an additional Full Adder delay to stabilize due to the modified connection. Using this simplified timing analysis, we conclude that the final level adder outputs in bit positions 1 to 8 $(= n)$ are available after a maximum of $2(\log_2 n - 1)$ Full Adder delays, and the outputs in bit positions 9 $(= n + 1)$ to 15 $(= 2n - 1)$ take an additional Full Adder delay to stabilize. These signals, however, constitute inputs to a carry propagate adder. There is, presumably, a finite delay due to the carry bit propagation, in the carry propagate adder, before the late inputs (ie inputs in bit positions $n+1$ to $2n-1$) start to effect the output. Since it is likely that by the time the carry bit signal reaches the first bit position in which a late signal is used, that signal will have stabilized and as a result no increase in the overall delay of the multiplier will result.

A possible floorplan of an 8-bit version of the BCSA multiplier architecture is shown in Figure 8.16. The layout is composed of a two-dimensional array of $n - 1$ rows of n columns of the basic multiplier cell. The layout is recursive with any level-i adder placed between two level-$(i-1)$ adders. The connections between successive adder levels increase with i but the wire lengths are equal for all bit positions. These interconnections must be passed across adders of other levels and the second layer of metal is used for their implementation so as not to introduce an additional area penalty.

The $(2n - 1)$-bit carry propagate adder is subdivided into three sections: $n/2$, n and $n/2 - 1$ bits long, with its middle section running through the centre of the array. By shifting the upper and the lower halves of the array an area efficient floorplan, not containing any "dead" space, is produced.

The silicon area of the BCSA multiplier is given by:

$$\begin{aligned} A_{BCSA} &= nL_{FA} \times \left[2W_{FA}\left(\frac{n}{2}-1\right)+nW_{AND}\right] + A_{CPA,2n-1}, \\ &= n(n-2)A_{FA} + n^2 A_{AND} + A_{CPA,2n-1}. \end{aligned} \tag{8.24}$$

Comparing expressions (8.23) and (8.24) shows that with the module interconnection modification proposed in this work a saving in silicon area of approximately 50% is achieved, compared to the original multiplier design of [Vuil83], and approximately 25% compare to the redundant binary adder tree multiplier of [TaYY85] and [HNNT87], at no degradation in multiplier performance.

8.3.2 "Double Stream" Multiplier (DS)

This parallel multiplier is based on the modified Booth algorithm multiplier discussed in Section 8.2.5 and the scheme proposed by Pezaris [Peza71].

The parallelism at the functional module level of the modified Booth algorithm multiplier is similar to the parallelism at the functional module level of a carry save adder array multiplier. In both cases only one row of Full Adder modules have generated a stable output at any instant of time. The modification proposed here is simple. The parallelism at the module level is doubled by interconnecting the Full Adders in such a way that there are effectively two parallel streams of partial product summation.

This modification is applied to the modified Booth algorithm, since by applying the modified Booth algorithm the number of Full Adder stages is halved, and the method proposed in this section effectively reduces the number of Full Adder stages by a further factor of 2. Note that in the multiplier of [Peza71] only the sum signals were made to propagate along a number of parallel streams and the carry signals were made to propagate in the conventional manner. Thus, as mentioned previously, that scheme will only have an advantage over the conventional ACSA multiplier if the sum delay is greater than the carry signal delay of the Full Adder module used.

In the DS multiplier of this work, the Full Adder module (to be discussed in Subsection 8.7.2.1) is designed in such a way that the critical paths for the carry and the sum signals are made of equal delay, and each includes only one inverter and one transmission gate.

Figure 8.17 illustrates the new idea applied to a 4×4 bit ACSA multiplier, rather than to a modified Booth algorithm multiplier, for simplicity. The method of producing the "matrix" of partial products remains the same as for the ACSA multiplier, but a new

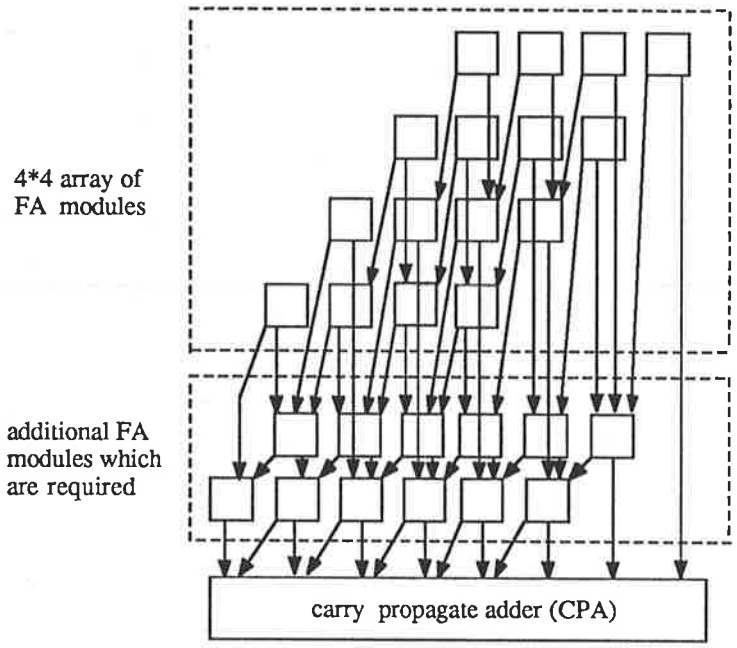


Figure 8.17: DS multiplier: block diagram.

and faster method of combining the partial products is proposed. In reference to the figure, if the rows of Full Adders were coloured red and black alternately then the sum and carry bit outputs from a red row of Full Adders are input to the next red row of the Full Adders, and similarly for the black row adders. At the bottom of the array there are four addends which need to be combined together (two red and two black) instead of two as in the case of the ACSA multiplier. The four addends are first reduced to two with the use of two additional rows of Full Adders, connected together as shown in Figure 8.12, before the signals are finally applied to a fast carry propagate adder. It can be seen from the figure that within the array of modules the maximum signal length includes only two Full Adders. As stated at the beginning of this section, by introducing this idea into the modified Booth algorithm (MBA) multiplier, the number of Full Adder stages, within the array, is reduced by a factor of 4 compared to the conventional ACSA multiplier. This architecture is termed the DS multiplier.

The interconnecting wires between successive rows of adders, in the DS multiplier, are approximately twice as long compared to the conventional MBA multiplier, since the sum output must be shifted four places to the right before being connected to the next row of adders of the same colour and the carry signals must be shifted three places to the right. Provided that $t_s = t_c$ for the FA module used, this interconnection scheme of modules will result in a faster and more regular multiplier than the Pezaris design and a faster multiplier with the same degree of regularity of interconnect as the MBA multiplier.

The overall delay of this multiplication scheme, assuming that the delays of the modules BEN and PPG are each 2τ, as before, is approximately given by:

$$\begin{aligned} t_{DS} &= t_{BEN} + t_{PPG} + \frac{n}{4}t_s + 2t_s + t_{CPA,n}, \\ &= 2\tau + 2\tau + \frac{n}{4}t_s + 2t_s + t_{CPA,n}, \\ &= \left(\frac{n}{2} + 8\right)\tau + t_{CPA,n}, \end{aligned} \tag{8.25}$$

and the silicon area (A_{DS}) occupied by this new multiplier is estimated using the diagrams of Figures 8.11 and 8.17 to be:

$$\begin{aligned} A_{DS} &= [(n+2)L_{FA} + 2L_{FA} + L_{BEN}] \\ &\times \left[\left(\frac{n+1}{2}\right)(W_{FA} + W_{PPG}) + 2W_{FA}\right] + A_{CPA,2n}, \\ &= (n+5)(n+4)A_{FA} + A_{CPA,2n}, \end{aligned} \tag{8.26}$$

assuming $L_{FA} = L_{PPG}$ and $W_{FA} = W_{PPG}$.

8.3.3 Discussion

The area requirements and the time delays of the two new parallel multiplier architectures, presented in this thesis, are compared with other multiplier designs for unsigned n-bit operands in Table 8.2.

After studying the maximum necessary adder lengths at each level of addend summation, within the Vuillemin multiplier, the BCSA multiplier architecture has been proposed. The BCSA multiplier has the same time delay as the Vuillemin multiplier but it occupies an area which is approximately 50% smaller than the area occupied by the Vuillemin multiplier, while maintaining the regular structure of the Vuillemin multiplier. The DS multiplier architecture is an extension of the Pezaris and the modified Booth algorithm multipliers. The FA module interconnection within the DS multiplier results in the longest signal path containing approximately half the number of FA modules of the modified Booth algorithm multiplier.

To a first approximation all multipliers occupy equal silicon areas, and have area complexity of $O(n^2)$. The ACSA and the DS multipliers have time complexity of $O(n)$ and the WT and the BCSA have time complexity $O(\log_2 n)$.

For operands with $n = 24$ the number of FA module stages within the WT and the BCSA multipliers is 7 and 8, respectively, while for the DS multiplier is 8. There are 23 FA module stages within a 24-bit ACSA multiplier and 12 FA module stages within the MBA multiplier. For $n = 53$ (ie mantissa length in the double precision floating format

Multiplier	Multiplier Silicon Area	Multiplier delay
ACSA	$(n-1)^2 A_{FA} + n^2 A_{AND}$ $+A_{CPA,n-1}$	$2n\tau$ $+t_{CPA,n-1}$
WT	$(n-1)(n-2)A_{FA} + n^2 A_{AND}$ $+A_{CPA,2n-2-1.71\log_2 n}$	$3.42(\log_2 n)\tau$ $+t_{CPA,3n/2-1}$
DS	$[(n+5)(n+4)A_{FA}$ $+A_{CPA,2n}$	$(\frac{n}{2}+8)\tau$ $+t_{CPA,3n/2-1}$
BCSA	$n(n-2)A_{FA} + n^2 A_{AND}$ $+A_{CPA,2n-1}$	$4(\log_2 n - 0.5)\tau$ $+t_{CPA,3n/2-1}$

Table 8.2: Area requirements and speed performance of the ACSA, WT, DS and BCSA multipliers.

of the IEEE Standard) the number of FA module stages within the WT, BCSA, DS, ACSA and MBA multipliers is 9, 10, 15, 52 and 27, respectively. The logarithmic time complexity of the WT and BCSA multiplier is clearly visible. The time complexity of the DS multiplier is the same as that of the ACSA and the MBA multipliers, ie $O(n)$, however, the proportionality constant between n and the multiplier delay is smaller for the DS multiplier than for the ACSA and the MBA multipliers. Thus, the delay of the DS multiplier is almost equal to the delays of the WT and BCSA multipliers for $n \leq 32$, and it is $\approx 50\%$ larger than that of the BCSA multiplier, and $\approx 45\%$ smaller than that of the MBA multiplier for 53-bit operands.

The layout of the WT multiplier is very complicated, whereas the layout of the BCSA is very regular by comparison, and easily extendable to any operand length. The layout of the DS multiplier is also regular but a number of different modules are required, compared to the WT and the BCSA multipliers which only require one module. However, for fabrication processes which provide only one layer of metallization the DS multiplier architecture may be preferred, if the silicon area is the critical factor. This is because the complexity of module interconnection increases with n in the WT and BCSA multipliers, while it remains constant for the DS multiplier. For this reason, more than one layer of metallization are desirable to reduce both the effective area occupied by interconnections and the delay introduced by long wires within the WT and BCSA multipliers.

8.4 Free Form Specification

The Floating Point Multiplier (FPM) system is implemented as a combinatorial static CMOS circuit, and it performs a floating point multiplication on two normalized operands represented in single precision format of the IEEE Floating Point Standard. Pathological cases are treated as described in Section 6.4.4. Because the maximum delay of the FPM system may potentially determine the clock period of the mP system, it is required that the maximum time delay of the FPM system be approximately the same as the maximum time delay of the FPA system, described in Chapter 7. In this way both the FPA and the FPM systems will be utilized at approximately the same efficiency within the environment of the mP system.

A diagram which shows the arrangement of input and output ports is given in Figure 8.18. The input and the output registers which are necessary for this asynchronous system to operate in the synchronous environment of the Mathematical Processor are not shown. The notation in this chapter is the same as the notation used for the corresponding quantities in Chapter 7, in the discussion of the FPA system. Any new symbols will be defined as they are introduced.

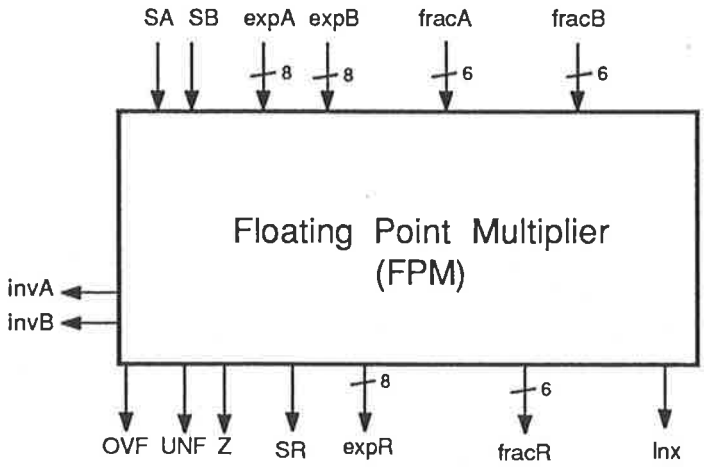


Figure 8.18: The Floating Point Multiplier system.

8.5 Floating Point Multiplication Algorithm

The usual hardware implementation performs the various tasks of the floating point multiplication algorithm in a sequential manner [WaMc82], [LeCT84], [WoLO84], [WiWi84], [WLWW84] and [SiOC86].

The following subsections present the conventional form of the floating point multiplication algorithm, an analysis of some of its numerical aspects, and a modification that results in an algorithm in which more of the originally sequential tasks are carried out concurrently, making it possible for the hardware realization to operate with a higher degree of parallelism at the module level.

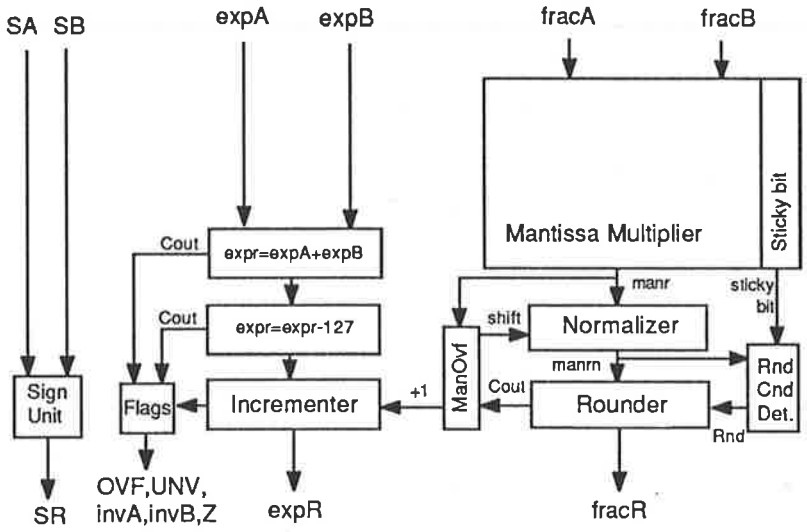


Figure 8.19: Block diagram of the conventional hardware realization of the floating point multiplication algorithm, after [WaFl82].

8.5.1 The Conventional Form

The conventional form of the algorithm for normalized floating point multiplication, [Flor63], [Hwan79], [Kuck81], [WaFl82], can be summarized as set output below, with all tasks shown as combined together normally being performed in parallel:

1. Set the sign of the result, compute the intermediate value of the resultant exponent, detect any invalid operands and set appropriate flags, perform a fixed point multiplication of the two mantissas;

2. Normalize the mantissa product, if necessary, and increment the intermediate value of the resultant exponent;

3. Round the normalized mantissa product, if necessary, by adding a one in the position of the first guard bit;

4. Check for mantissa overflow, normalize the mantissa, if required, and increment the intermediate value of the resultant exponent;

5. Check for exponent overflow and underflow and set the final result and the status flags accordingly.

A block diagram showing the conventional hardware realization (CFPM) of this form of the floating point multiplication algorithm is shown in Figure 8.19.

8.5.2 An Analysis of Normalized Floating Point Multiplication

Since both operand mantissas are normalized (the case of one or both operands being zero is treated separately later), ie in the range $1.0 \leq manA, manB < 2.0$, their product ($manr$) lies in the range $1.0 \leq manr < 4.0$. This means that the mantissa product can overflow and can require a normalization shift of one place to the right. The mantissa product, however, will never have any leading zeros and thus no normalization shifts to the left will ever be required.

There can only be one mantissa overflow in the process of floating point multiplication, either as a result of mantissa multiplication or as a result of mantissa rounding operation. This can be easily shown as follows. In order for the rounding operation to produce a mantissa overflow the n-bit normalized mantissa product ($manr_n$) to be rounded must have a value such that:

$$manr_n = 2 - 2^{-n}. \tag{8.27}$$

The maximum value of a mantissa product ($manr_{max}$) is given by:

$$manr_{max} = [2 - 2^{-(n-1)}]^2, \tag{8.28}$$

which following the normalization and prior to rounding has the maximum value given by:

$$\begin{aligned} manr_{n,max} &= manr_{max}/2, \\ &= 2 - 2^{-(n-2)} + 2^{-(2n-1)}, \\ &< 2 - 2^{-n}. \end{aligned} \tag{8.29}$$

Thus, the condition (8.27) cannot be satisfied.

8.5.3 The Modified Form

The combinatorial circuit presented in this chapter performs normalized floating point multiplication in a time delay which is only slightly longer than the time delay needed to perform the corresponding fixed point mantissa multiplication. This time difference

between the floating point multiplication and the corresponding fixed point mantissa multiplication is independent of the operand lengths (n). Floating point multiplication is performed, according to the *new* form of the algorithm in two steps, as set out below:

1. The sign of the result is determined, invalid operands are detected, the fixed point multiplication of the mantissas is performed and two versions of the rounded product are generated; one rounded under the assumption that the product overflows and one rounded under the assumption that the product does not overflow. In parallel, two resultant exponents and two versions of the various status flags, which depend on the value of the resultant exponent, are generated; one generated under the assumption that mantissa multiplication overflows and the other generated under the assumption that mantissa multiplication does not overflow.

2. When the mantissa product becomes available it is checked for overflow and on the basis of this test the correct versions of the resultant exponent, the rounded and normalized resultant mantissa, and the status flags are chosen and delivered as the final result.

In the first step of the algorithm the two rounded versions of the mantissa product become available a constant time delay following the time at which the mantissa product (output of the fixed point multiplier) becomes stable. This is explained in more detail in Section 8.7.5. The delay of the second step of the algorithm is simply the time for the signals to pass through a $2:1$ multiplexer; a negligible time delay compared to that of fixed point multiplication.

The new floating point multiplier architecture, ie the FPM system, is evaluated and compared with the CFPM system with respect to the silicon area requirements and performance in Section 8.9, where circuit simulation results are presented and discussed.

8.6 System Partitioning

The hardware architecture of the FPM system divides naturally into three composition modules:

1. *Sign*: determines the sign bit (SR) of the result;

2. *Exponent*: calculates the resultant exponent ($expR$), checks for invalid values of the operands and sets the status flags $invA$ and $invB$ accordingly, checks for result

overflow, underflow and zero and sets the status flags OVF, UNF and Z accordingly, and also generates the control signal $ManZ$, indicating that the constant 0 is to be delivered as the mantissa of the result;

3. *Mantissa*: calculates, rounds and normalizes the fixed point product of the operand mantissas, generates the inexact result (Inx) flag and a control signal $ExpOvf$, indicating mantissa product overflow.

The Sign module is completely independent of the other two modules. There is, however, a limited interaction between the Exponent and the Mantissa modules. Figure 8.20 depicts a block diagram of the FPM system in terms of the three modules described above.

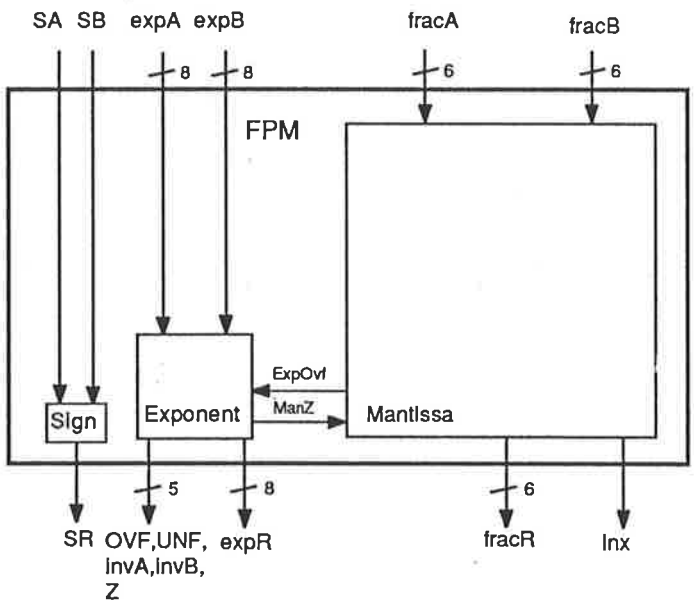


Figure 8.20: Floating Point Multiplier: block diagram at the first level of partitioning.

At this level of partitioning no formal functional or partitioning validation was performed and all validation steps of all system partitioning steps relied on informal hand calculation techniques and the results obtained from an interactive use of the USA functional simulator.

8.7 Physical Architecture

As mentioned in the introduction of this chapter, a number of limitations caused an implementation of a reduced circuit to be undertaken. In particular the Mantissa module

has been decreased in size from 24 to 7 bits. The implementation and performance estimation of the 24-bit mantissa multiplier are discussed in Section 8.9. The Exponent module has not been reduced in size and it operates on 8-bit inputs.

In an initial floorplan, closely corresponding to the block diagram of Figure 8.20(a), of the FPM system formulated at the first level of partitioning, the three modules are placed next to each other in the order of the corresponding three fields of the operands. The Exponent module is placed towards the bottom boundary of the Mantissa module, in order to minimize the lengths of the wires running between the two. The floorplan of the complete FPM system is strongly affected by the choice and the implementation of the parallel fixed point multiplier which forms a considerable part of the Mantissa module. The final floorplan will therefore be discussed after consideration of the physical implementation of the fixed point parallel multiplier module.

In the remainder of Section 8.7 the hardware design of the FPM system is elucidated by detailing the implementations of the various composition and functional modules.

8.7.1 Mantissa

The Mantissa composition module is further functionally partitioned into four modules, as shown in Figure 8.21 and described below:

1. *Mantissa Multiplier*: evaluates the 14-bit product (pr) of the two 7-bit operand mantissas. The resultant mantissa ($manr$) consists of the 9 most significant bits of the 14-bit mantissa product;

2. *Sticky Bit*: evaluates the "sticky bit" (S) necessary for proper rounding operation by performing a logical OR function on all bits of the mantissa product ($manr$) between bit positions 1 and 6 (7-bit version), where bit 1 is the least significant bit of the product;

3. *Rounding Condition Detector (RndCndDet)*: generates two control signals *RndA* and *RndB*. The *RndA* control signal specifies that if the mantissa product does not overflow it needs to be rounded, and the *RndB* control signal specifies that if the mantissa product does overflow it needs to be rounded;

4. *Mantissa Rounder A (ManRndA)*: generates a rounded mantissa product ($manrA$) according to the control signal *RndA*. The carry signal (cA) from the most significant bit position of this module constitutes one of the signals used to determine mantissa overflow;

5. *Mantissa Rounder B (ManRndB)*: generates a rounded mantissa product ($manrB$) according to the control signal *RndB*;

6. *Mantissa Overflow Detector (ManOvf)*: detects whether the operations of mantissa multiplication or product rounding result in mantissa overflow, generates the control signal *ManOvf* specifying which of $manrA$ or $manrB$ is to be delivered as the mantissa of the result, and generates the control signal *ExpOvf*, specifying the correct form of the exponent of the result and of the status flags;

7. *Mantissa Selector (ManSel)*: when the final result is different from infinity or zero, this module passes the correctly rounded and normalized mantissa product to the output. Bit 7 of the mantissa of the result is not delivered because it corresponds to the hidden bit which is always = 1 except for the special cases of the result being zero or infinity. In a case of the final result being either zero or infinity (a condition indicated by the *ManZ* control signal) a zero is delivered by the Mantissa Selector module, as the fractional field of the result.

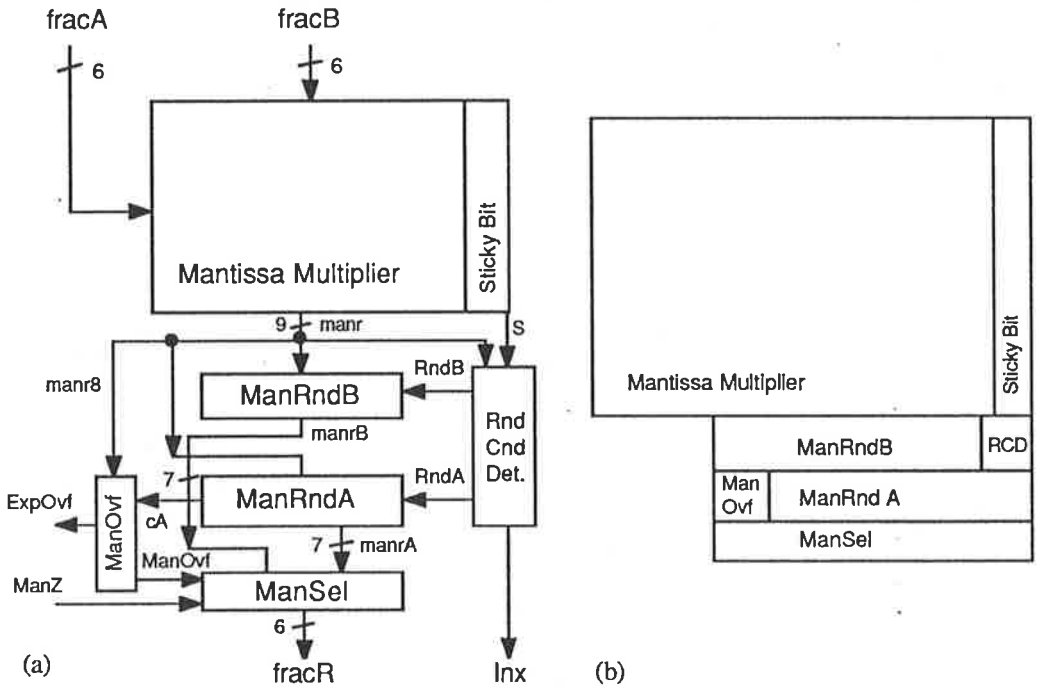


Figure 8.21: Mantissa: (a) block diagram, (b) proposed floorplan.

8.7.2 Mantissa Multiplier

This module has been implemented as a 7-bit version of the DS multiplier, discussed in Section 8.3.2.

For 24-bit operands (full single format version) the DS multiplier is similar in performance and silicon area requirements to the BCSA multiplier, discussed in Section 8.3.1. Two factors determined why this multiplier was chosen: firstly, the 24-bit version of the DS multiplier has a slight area advantage over the 24-bit version of the BCSA multiplier and secondly, a modified Booth algorithm multiplier had previously been implemented by the author (as a part of the learning process) and most modules were already available, with relatively few changes necessary to convert it to the DS multiplier. However, for two 64-bit floating point operands the mantissas are 53 bits long and the BCSA multiplier would be the preferred choice, because of its superior speed performance.

Some of the aspects of the physical implementation of the DS multiplier were discussed in Section 8.3.2. In this section the design of the 7-bit version is discussed in more detail.

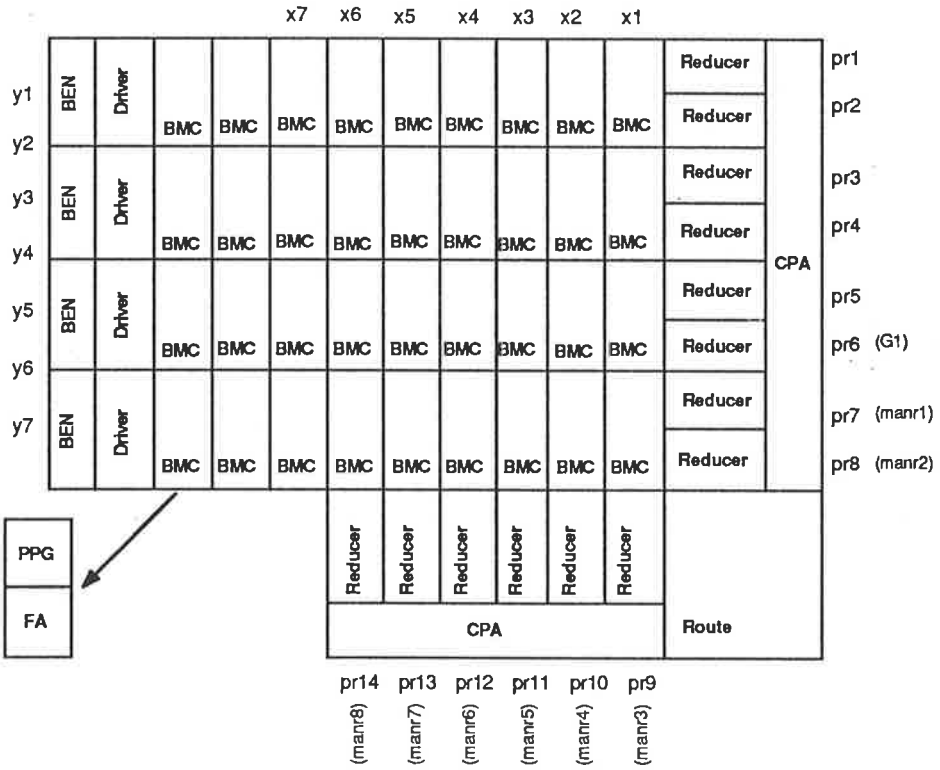


Figure 8.22: Mantissa Multiplier: floorplan.

Referring to Figure 8.21(a) the two operands to be multiplied are applied to the Mantissa Multiplier module along the top and the left hand side boundaries and the product is collected, after a time delay, along the right hand side and the bottom boundaries. In the case of two 7-bit unsigned operands the modified multiplier word, padded with two zeros to the left of its most significant bit, is 9 bits long and the multiplicand word, padded with one zero to the left of its most significant bit and one zero to the right

of its least significant bit, is also 9 bits long (see Section 8.2.5). The Both encoding process results in four 9-bit partial products, formed from the multiplicand, which must be summed together to obtain the final product result.

The structure of the 7-bit version of the Mantissa Multiplier is depicted in Figure 8.22. The Mantissa Multiplier module is subdivided into the following functional modules:

1. *Booth Encoder (BEN)*: encodes three successive multiplicand bits and defines the control signals necessary for the generation of a partial product,
2. *Basic Multiplier Cell (BMC)*: this module is further decomposed into:
   (a) *Full Adder (FA)*: a 1-bit full adder module,
   (b) *Partial Product Generator (PPG)*: forms the appropriate partial product from the multiplicand word in accordance with the control signals generated by the appropriate Booth Encoder module,
3. *Reducer*: a module composed of two Full Adder modules operating on signals of the appropriate polarity, which reduces the four addends generated by the array of the multiplier to two addends to be summed by a fast carry propagate adder,
4. *Carry Propagate Adder (CPA)*: a 14-bit fast carry propagate adder incorporating the carry bypass technique.

As discussed in Section 8.3.2, the BMC modules in the two top rows of the multiplier array need to contain the PPG modules only. It was found, however, that the connections between the BEN modules and the BMC modules in the leftmost column of the array, and between the Reducer modules and the BMC modules in the rightmost column of the array were greatly simplified (no routing channels were required) if the modules BMC along the top two rows also contained the FA modules, making them identical to the BMC modules in the remainder of the array. For this reason, the implementation containing identical BMC modules throughout the array was adopted.

The FA modules in the third row of the array operate on outputs of the PPG modules of that row, the sum outputs from the FA modules in the first row shifted by four bit positions and the carry outputs of the FA modules in the first row shifted by three bit positions. A corresponding connection exists between modules in the second and the fourth rows of the array. At the bottom boundary of the array and on its right hand side boundary the four addends produced by the modules of the array are first reduced to two addends by a row and a column of Reducer modules before being applied to the CPA (Carry Propagate Adder) module.

The control signals generated by the BEN modules run across the complete length of the Mantissa Multiplier module. The length of the necessary wires in the 24-bit version of the module will be approximately $1.5 \times 10^{-3}m$ long, with the multiplier implemented according to the $Orbit^{+}$ process design rules. In order to decrease the capacitive loading and the RC delay of such lines, the first layer of metal is used for their implementation. A typical load on each of these lines in the 24-bit version of the multiplier is approximately $2 \times 10^{-12}F - 2.5 \times 10^{-12}F$. Drivers of sufficient size (determined via circuit simulations) are included at the outputs of the BEN modules to propagate the control signals in a time approximately equal to the time delay of the FA module. The first level of metallization is used for the distribution of power supply within the multiplier module, while the second metal layer is used for running the vertical sections of the inter-row interconnections and the Mathematical Processor system buses.

The physical implementations of the modules, which are used in the composition of the module Mantissa Multiplier, are described in detail in the following subsections.

8.7.2.1 Full Adder (FA)

The delay characteristics of this module determine, to a large extent, the delay characteristics of the Mantissa Multiplier module. For best multiplier performance the carry and the sum delays of the FA module must be as small as possible and equal, since the two signals propagate in parallel across equal number of the FA module stages.

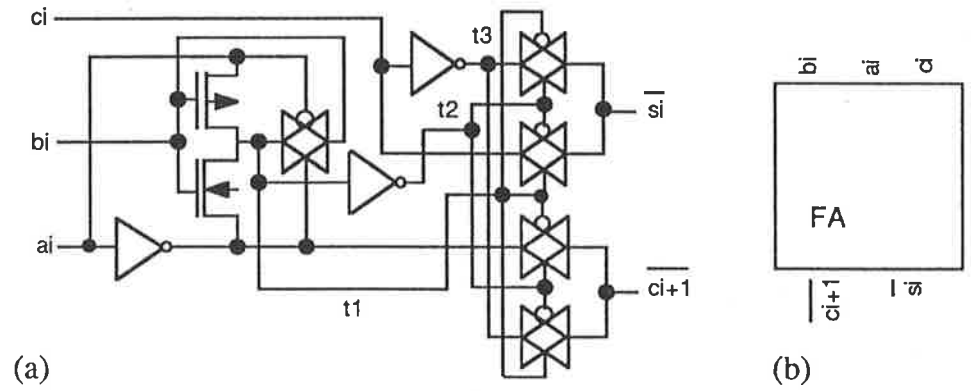


Figure 8.23: Full Adder: (a) logic implementation, (b) leaf module environment specification.

The full adder cell used is a modified version of the 24-transistor "transmission gate adder" given in [OhKD79] and [WeEs85]. The original logic implementation of the "transmission gate adder" has been modified, in this work, to decrease the critical path

delay and the transistor count. The logic implementation of the resulting FA functional module is shown in Figure 8.23(a), and the leaf module environment specification in Figure 8.23(b). The final symbolic layout is shown in Figure 8.24 and the result of compaction (mask domain representation) is presented in Figure 8.25.

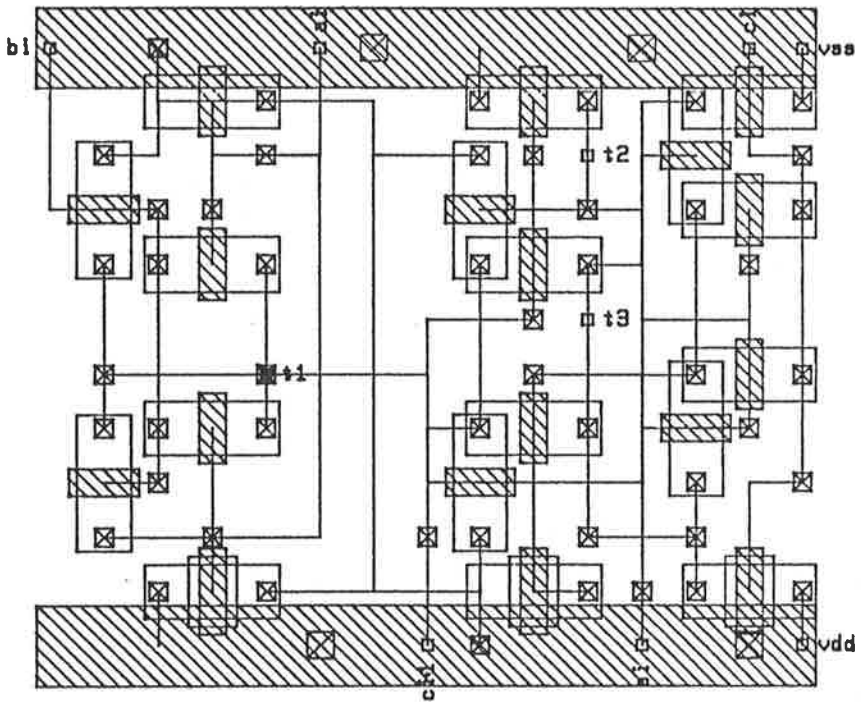


Figure 8.24: Full Adder: symbolic domain representation.

A speed increase has been achieved by removing one inverter from both the sum and the carry paths. This has resulted in an adder cell that when operating on true inputs produces inverted outputs, and when operating on inverted inputs produces true outputs. This scheme requires the use of a single type of full adder module inside the body of the array, thus, preserving the regularity of function and interconnect within the multiplier array, but it requires implementation of three different Reducer modules, each of which operates on inputs with different combinations of signal polarities. However, the decrease in delay and the saving in silicon area achieved as a result of these modifications of the transmission gate adder more than compensate, especially for large operands, for the addition design complexity. Additional saving in area has also been achieved by replacing parts of one of the exclusive-OR gates, present in the original design of the "transmission gate adder", by an inverter. The transistor count has been reduced from twenty four to eighteen, a saving of 25%.

The longest delays of the sum and carry paths are approximately equal, since both depend on the signal at node t_1, Figure 8.23. In specifying the logic implementation

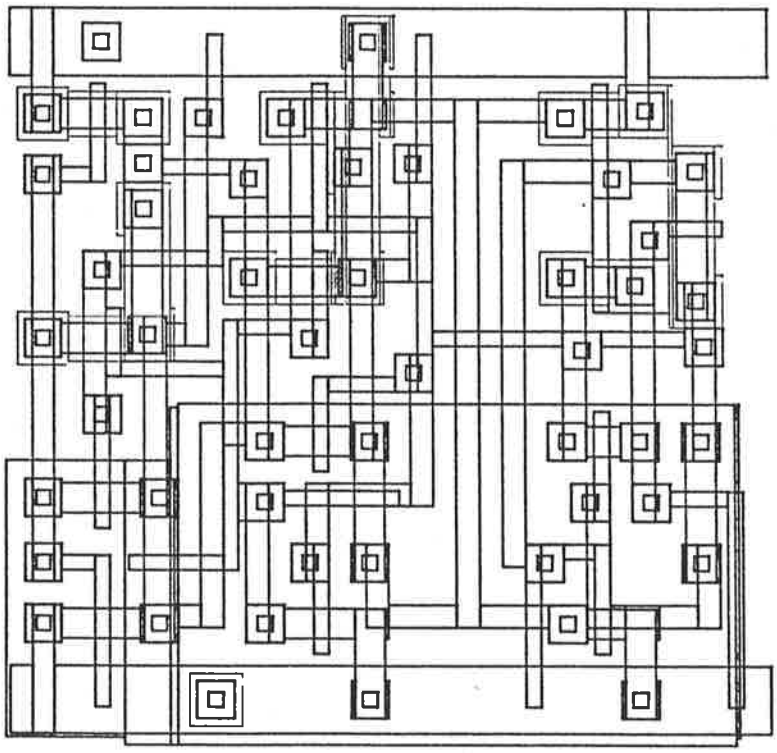


Figure 8.25: Full Adder: mask domain representation corresponding to the $Orbit^{+}$ 1-micrometer CMOS process design rules. The dimensions of the module bounding box are $93\lambda \times 87\lambda$, $\lambda = 0.5 \times 10^{-6}m$.

of this module, care was taken to ensure that when it was connected in the carry save, configuration within the Mantissa Multiplier module, no long signal paths consisting of unbuffered transmission gates could occur. It was determined, via circuit simulations using FACTS (see Figure 8.26), that the delay from input a_i to the node t_1 was larger (with signals on nodes b_i and c_i being held stable) than the delay from input b_i to node t_1 (with the signals on nodes a_i and c_i being held stable). Therefore, for better performance of the multiplier the FA input a_i is connected to the output p_i of the corresponding PPG module (see Figure 8.28), while the input b_i is connected to the s_i output of FA module in the previous row. This interconnection ensures that the input a_i of each FA module becomes stable prior to the input b_i becoming stable. Assuming that both inputs b_i and ci_i arrive simultaneously, the worst case delay of the FA module is composed of one inverter delay, to define the states at nodes t_1 and t_3, and one transmission gate delay to define the state of the two output nodes s_i and c_{i+1}, ie $= 2\tau$, with the delays of the inverter and the transmission gate each being $= \tau$.

The delay characteristics were investigated using circuit simulations by connecting the FA module in a *test frame* environment in which additional FA modules were connected to the inputs and to the outputs of the module under investigation, to ensure correct

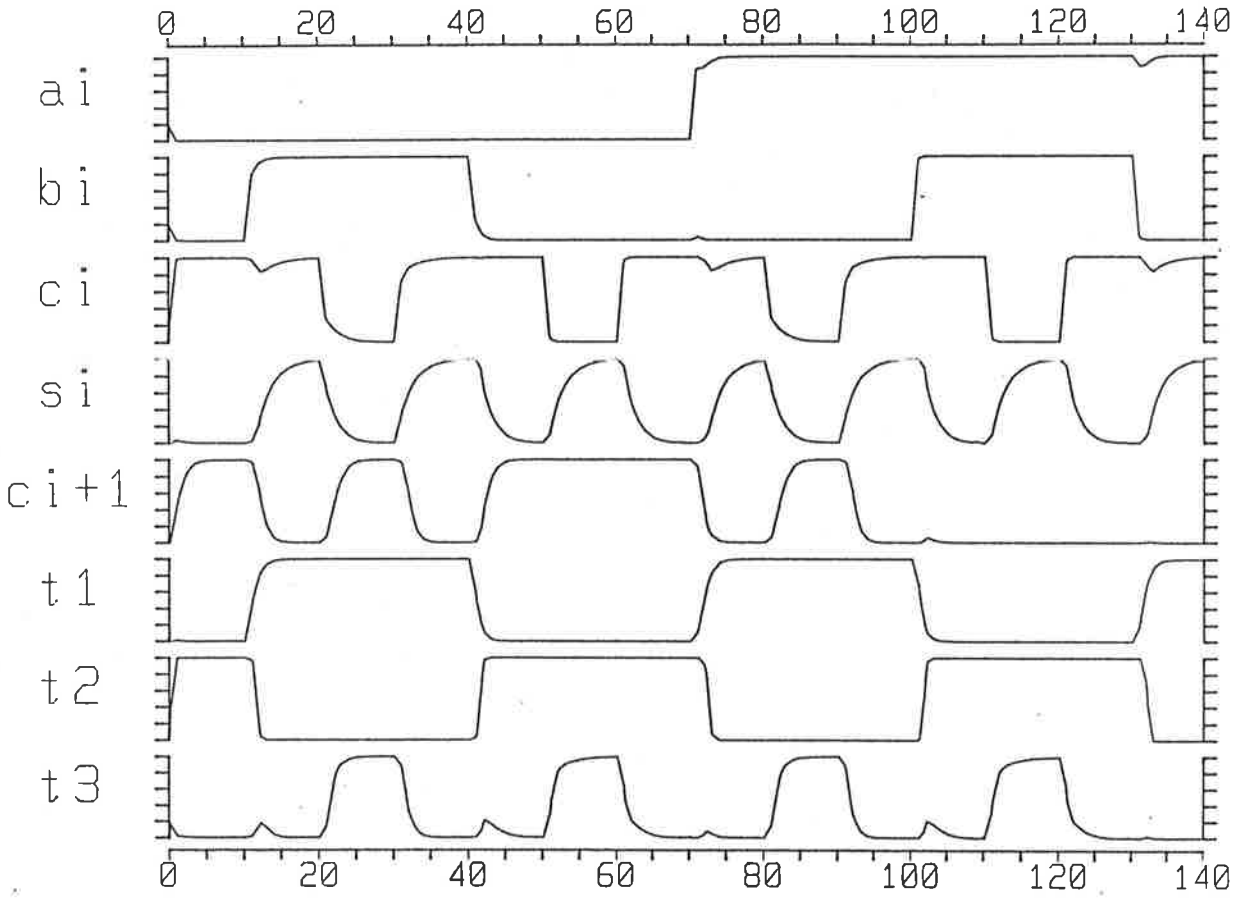


Figure 8.26: Full Adder: simulation results. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

input waveforms and correct output node loadings. The circuit simulation results for the $Orbit^{+}$ process parameters are given in Figure 8.26. It can be seen from the figure that the carry and sum signal delays are approximately equal, as intended.

8.7.2.2 Booth Encoder (BEN) and Partial Product Generator (PPG)

The logic designs of the Partial Product Generator module (PPG) and the Booth Encoder module (BEN) are closely related. The three input bits of the multiplicand y_{i-1}, y_i and y_{i+1} can be fully decoded into five control lines: add, subtract, X, $2X$, 0, (where X stands for the multiplier word). This results in a simple logic implementation of the PPG module at the expense of a more complicated logic implementation of the BEN module and the relatively large number of control lines which need to be routed within each row of the multiplier array, which lines require large signal drivers and occupy valuable silicon area.

By transferring some logic from the BEN module to the PPG module the number of control lines can be reduced to three, namely $ctl1$ (pass X to the output of the PPG

module), $ctl2$ (pass $2X$ to the output of the PPG module) and $ctl3$ (complement-do not complement the output of the PPG module). A logic level zero on all control lines is decoded by each PPG module as "pass zero to the output".

The necessary logic expressions for the three control lines are as follows:

$$\begin{aligned} ctl1_i &= y_i \oplus y_{i-1}, \\ ctl2_i &= \overline{\overline{y_i \oplus y_{i+1}} \cdot y_i \oplus y_{i-1}}, \\ ctl3_i &= y_{i+1}. \end{aligned} \tag{8.30}$$

A logic implementation of the BEN functional module is given in Figure 8.27(a) and its leaf cell environment specification in Figure 8.27(b).

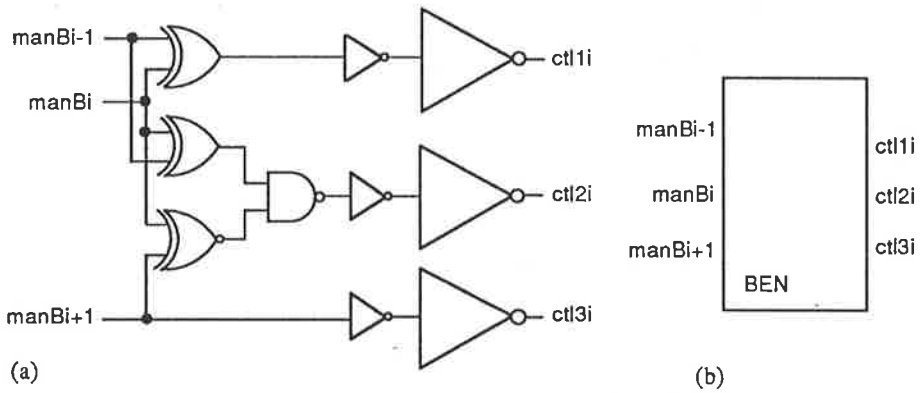


Figure 8.27: BEN: (a) logic implementation, (b) leaf module environment specification.

A similar diagram for the PPG module is given in Figure 8.28. Pseudo-nMOS circuit design technique was used in the implementation of this module and resulted in a reduction, by almost 50%, in transistor count over the full static CMOS circuit implementation (ie one using transmission gates). An additional inverter (inv2) is included to ensure that the signal goes through no more than one pass transistor before being restored to full signal level.

8.7.2.3 Reducer

The Reducer module is architecturally equivalent to the BMC module used in the implementation of the BCSA multiplier, ie it is a module composed of two FA modules.

Because of the alternating signal polarity scheme adopted in the design of the integer multiplier, due to the particular FA module implementation used as discussed in Subsection 8.7.2.1, a number of different Reducer modules are required to properly account

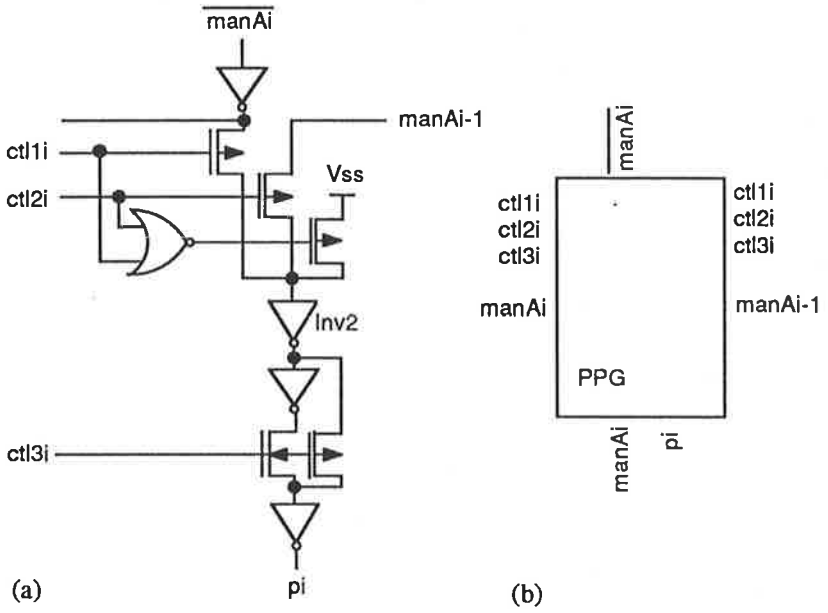


Figure 8.28: PPG: (a) logic implementation, (b) leaf module environment specification.

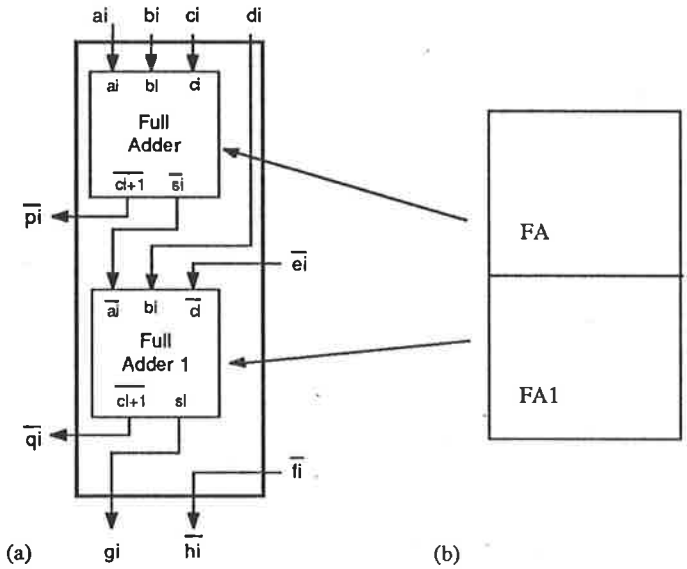


Figure 8.29: Reducer: (a) block diagram, (b) floorplan.

for the differences in signal polarities generated by the array of BMC modules of the Mantissa Multiplier. The different Reducer modules are implemented by modifying the transistor connections within the FA module.

For example, Figure 8.29(a) shows the block diagram of the Reducer module which is used along the bottom edge of of the multiplier array, as shown in Figure 8.22. The top module of this Reducer module is the FA module whose logic implementation is given in Figure 8.23(a). The bottom module (FA1) is a modified version of the FA module of

Figure 8.23(a). The FA1 module operates on one input with true polarity and the two other inputs with inverse polarities, and generates a sum signal with true polarity and the carry signal output with inverse polarity.

8.7.2.4 Carry Propagate Adder (CPA)

The optimization of the carry path delay is of primary concern in the design of this module. The carry bypass adder used in the implementation of the Mantissa Adder, shown in Figure 7.22, is also used to implement the Carry Propagate Adder module. The particular 1-bit full adder (Bitadd) modules used need to be modified, however, to take into account the fact that at any significance the two inputs a_i and b_i, generated are of opposite polarities.

8.7.3 Sticky Bit

The "sticky bit", required for a correct rounding of the mantissa product, is generated by a logical OR function of all the mantissa product bits between the least significant bit position and the $n-2$ bit position ($n-2=5$ in the reduced case example and $n-2=22$ in the 24-bit version of the multiplier).

The logical implementation of this functional module is the same as that of the Stickybit module, shown in Figure 7.19, used in the the FPA system, with the transistors involving the shift control signals ($shift_i$) omitted since they are not required in this case.

8.7.4 Round Condition Detector (RndCndDet)

This functional module generates the control signals *RndA* and *RndB*, as well as the inexact result flag. As defined in Section 8.7.1, the signal *RndA* indicates that if the mantissa product does not overflow then a rounding operation will be required, and the signal *RndB* indicates that if the mantissa product does overflow then a rounding operation is required.

The inputs to this module consist of the output S ("sticky bit") of the Sticky Bit module, the guard bit (G_1) and the first two least significant bits ($manr_1$ and $manr_2$) of the output of the Mantissa Multiplier module. The logic expressions for the two control signals generated by this module are given by:

$$RndA = G_1 \cdot (manr_1 + S), \quad RndB = manr_1 \cdot (manr_2 + G_1 + S). \tag{8.31}$$

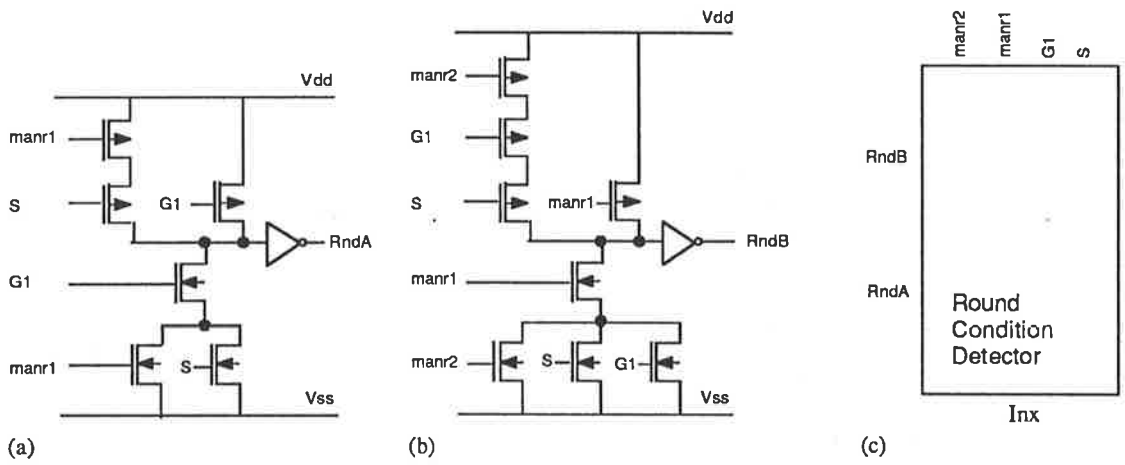


Figure 8.30: Round Condition Detector: (a) logic implementation for $RndA$, (b) logic implementation for $RndB$, (c) leaf module environment specification (the two input OR gate implementing the Inx flag is not shown).

The inexact result status flag is given by:

$$Inx = G_1 + S. \tag{8.32}$$

The RndCndDet module is implemented with static CMOS combinatorial logic gates. Its logic implementation is shown in Figure 8.30(a) and (b), and its leaf module environment specification in part (c) of the figure.

8.7.5 Mantissa Rounder (ManRndA and ManRndB)

As described previously, the speed of floating point multiplication is increased in this design, apart from a number of other modifications, by generating two versions of the mantissa product; one rounded under the assumption of no mantissa product overflow and the other under the assumption that mantissa product overflow takes place. Two mantissa incrementing modules (ManRndA and ManRndB) are used to implement these functions. These modules are realized in the same way as the ManInc module, shown in Figure 7.33, of the FPA system.

By using two mantissa incrementer modules to generate two results, it is possible to produce a correctly rounded mantissa only a small delay (independent of the operand sizes) after the mantissa product has stabilized. This result can be demonstrated by considering a system composed of an n-bit adder, ie the CPA module in the FPM system, directly connected to an n-bit incrementer, ie the ManRnd module of the FPM system. The longest time delay of such a system results when a carry $= 0$ is made to

propagate across the complete length of the n-bit adder and a carry $= 0$ or $= 1$ is made to propagate across the complete length of the n-bit incrementer. This time delay is equal to the time delay for the carry bit to propagate across the complete length of the adder plus the time delay of a 1-bit incrementer functional module, rather than the time for the carry to propagate across the complete length of the adder plus the time for the carry to propagate across the complete length of the n-bit incrementer. This is true since, as soon as the least significant bit of the adder output becomes stable the n-bit incrementer commences to generate its stable output, with the consecutive output bits from the incrementer lagging behind the corresponding output bits from the adder by approximately the delay of a 1-bit incrementer functional module.

In the FPM system the inputs of the two incrementers, ManRndA and ManRndB, are directly connected to the output of the CPA module of the Mantissa Multiplier, as shown in Figure 8.21(a). It follows from the above discussion that the outputs from the modules ManRndA and ManRndB stabilize a constant time delay following the time at which the output of the CPA module becomes stabile. The required resultant mantissa is then selected with a 2:1 multiplexer module, whose delay is constant and independent of the operand size.

The approach of using two mantissa incrementing modules, rather than only one as is the case in the CFPA system, provides a saving in the delay of the FPM system, compared to the delay of the CFPA system, approximately equal to the maximum delay of a 24-bit mantissa incrementer module. This result will be further illustrated in Section 8.9, where the results of circuit simulations are presented and discussed.

8.7.6 Mantissa Overflow Detector (ManOvf)

In order to detect a possible mantissa product overflow it is simply necessary to check the $14th$ bit (the $48th$ bit in the 32-bit version of the FPM system) of the Mantissa Multiplier output, ie bit pr_{14} (see Figure 8.22), which bit constitutes the $8th$ bit of the resultant mantissa ($manr$) prior to normalization and rounding. The control signal $ManOvf$, indicating mantissa product overflow, is then simply given by:

$$ManOvf = manr_8. \tag{8.33}$$

In order to correctly calculate the exponent of the result, it is also necessary to determine if the rounding operation results in an overflow. This is done by checking the carry bit out (cA), from the most significant bit position of the ManRndA module. The control signal $ExpOvf$ which indicates that mantissa multiplication or mantissa rounding results

in overflow is given by:

$$ExpOvf = ManOvf + cA, \tag{8.34}$$

which is simply realized with a 2-input OR gate. This signal is used, within the Exponent module (Section 8.7.8), to select the correct versions of the resultant exponent and the status flags.

8.7.7 Mantissa Selector (ManSel)

The mantissa of the final result is chosen from the two outputs, $manrA$ and $manrB$, generated by the modules ManRndA and ManRndB, respectively. The control signal $ManOvf$, defined in the previous section, indicates which of these two outputs is to be delivered as the mantissa of the result.

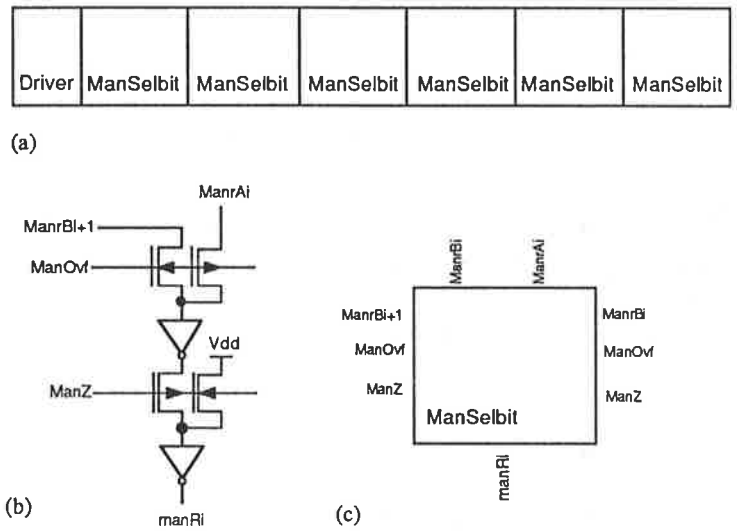


Figure 8.31: Mantissa Selector: (a) floorplan, (b) logic implementation, (c) leaf module environment specification.

If the final result to be delivered is the mantissa product rounded under the assumption that a mantissa overflow was produced, ie $manrB$, then it must be renormalized by shifting it by one bit position to the right. The shift is generated by appropriate interconnections between the 1-bit functional modules (ManSelbit) of the Mantissa Selector module, as shown in Figure 8.31(b).

Note that when the mantissa product rounded under the assumption of no mantissa overflow, ie $manrA$, is to be delivered as the mantissa of the result, and the rounding operation itself produces overflow, then the final result is still correct, since the rounded

(and overflowed) resultant mantissa has zeros in bit positions $manr_1$ to $manr_6$. These six bits together with the implicit bit ($manr_7$), which is assumed = 1 but not delivered as a bit of the output, constitute the correct result.

A number of special cases that need to be considered separately occur when one, (or both) of the operands is zero, infinity, unnormalized or NAN. These are detected by examining the exponents of the operands in the Exponent module (described in the following section). If any of the above cases occur the final result which is generated by the FPM system is either the Standard's representation of zero or infinity, as discussed in Section 6.4.4. In all such cases the fractional part ($fracR$) of the result = 0. This is indicated by the control signal $ManZ$ which is generated by the Exponent module, as described in the following section.

8.7.8 Exponent

The exponent arithmetic is complicated in floating point multiplication because of the exponent bias. In order to obtain the correct value of the intermediate resultant exponent ($expr$), the exponent bias must be subtracted from the sum of the operand exponents. In the case of a mantissa overflow, the intermediate value of the resultant exponent needs to be incremented by one to compensate for the right shift of the mantissa. The functions, giving the resultant exponent $expR$, are described mathematicaliy by the following arithmetic expression:

$$\begin{aligned} expR &= expA + expB - 127 + ExpOvf, \\ &= expr + ExpOvf. \end{aligned} \tag{8.35}$$

In general, it is not known until the mantissa product has been rounded whether the intermediate resultant exponent needs to be incremented by one or not. Thus, in the usual hardware realization of floating point multiplication, the exponent of the result is incremented, if necessary, following the operation of mantissa product rounding. As a result the resultant exponent ($expR$) becomes available a time delay after the final resultant mantissa becomes available. This additional time delay is approximately given by the maximum time delay required to increment by one the 8-bit intermediate value of the resultant exponent ($expr$).

In order to eliminate this additional time delay, in the FPM system, two versions of the intermediate resultant exponent are generated in parallel with the mantissa multiplication operation; one ($expr1$) given by expression (8.35) with $ExpOvf = 0$ and the other ($expr2$) given by that expression with $ExpOvf = 1$. Following mantissa rounding the

control signal *ExpOvf* is evaluated and the correct version of the resultant exponent is chosen, and delivered as *expR*, with an additional delay of a 2:1 multiplexer. Since the resultant mantissa also passes through a similar 2:1 multiplexer, the resultant mantissa and the resultant exponent should become available at approximately the same time.

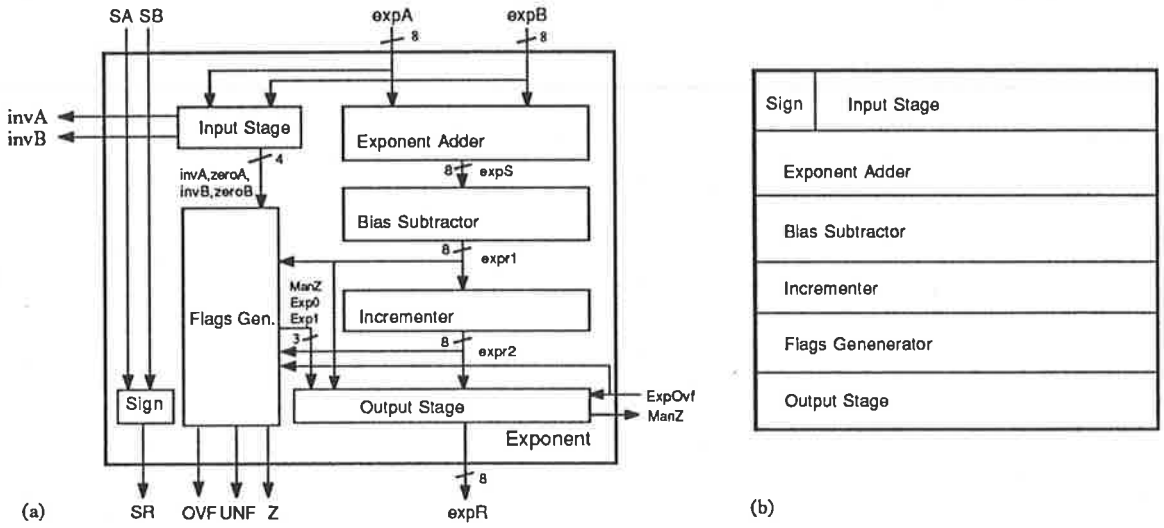


Figure 8.32: Exponent: (a) block diagram, (b) floorplan.

The module Exponent is functionally partitioned into the following modules:

1. *Input Stage*: checks the exponents of the operands for reserved values of 255 and 0, generates the invalid operand flags and two control signals ($zeroA$ and $zeroB$) accordingly;
2. *Exponent Adder*: calculates the 8-bit sum $expS$ ($= expA + expB$) of the operand exponents;
3. *Bias Subtracter*: calculates the intermediate resultant exponent by subtracting 127 from the sum of the operand exponents, ie $expr = expr1 = expS - 127$;
4. *Exponent Incrementer*: increments by one the output of the Bias Subtracter module, $expr2 = expr1 + 1$;
5. *Flags Generator*: detects result overflow, result underflow and result zero cases and sets the appropriate flags, generates the control signal *ManZ* specifying that the mantissa of the result has the value of zero, and generates two control signals, *Exp0* and *Exp1*, specifying that 0 or 255, respectively, is to be delivered as the resultant exponent ($expR$); and
6. *Output Stage*: selects the correct value of the exponent using the control signals *ExpOvf*, *Exp0* and *Exp1*.

A block diagram and a floorplan at this level of partitioning are given in Figure 8.32.

8.7.9 Input Stage

The logic implementation of this functional module is similar to the logic implementation of the Exponent Input Stage module, shown in Figure 7.12, of the FPA system.

As discussed in Section 6.4.4, the denormalized operands are treated as zero and the operands which are NAN are treated as infinity by the FPM system. Thus, when at least one of the operand exponents is zero, ie the corresponding operand is either zero or denormalized, the delivered result is automatically set to zero, and when at least one of the operand exponents is 255, ie the corresponding operand is either infinity or NAN, the Standard's representation of infinity is delivered as the result, and an overflow flag is set. The function of this module is to detect operands which cause the final result delivered by the FPM system to be set to zero or to infinity.

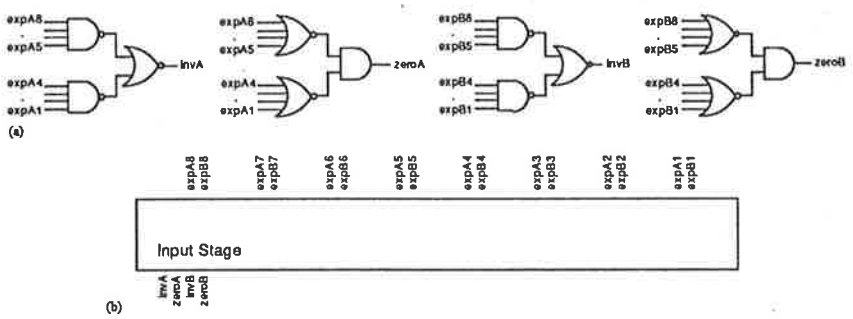


Figure 8.33: Input Stage: (a) logic implementation, (b) leaf module environment specification.

Two invalid operand flags, $invA$ and $invB$, indicating that $expA = 255$ and $expB = 255$, respectively, are generated within the Input Stage module according to expressions 8.36 given below:

$$\begin{aligned} invA &= expA_8 \cdot expA_7 \cdots expA_2 \cdot expA_1, \\ invB &= expB_8 \cdot expB_7 \cdots expB_2 \cdot expB_1. \end{aligned} \tag{8.36}$$

In addition, two signals, $zeroA$ and $zeroB$, indicating that $expA = 0$ and $expB = 0$, respectively are generated according to the following logic expressions:

$$\begin{aligned} zeroA &= \overline{expA_8} \cdot \overline{expA_7} \cdots \overline{expA_2} \cdot \overline{expA_1}, \\ zeroB &= \overline{expB_8} \cdot \overline{expB_7} \cdots \overline{expB_2} \cdot \overline{expB_1}. \end{aligned} \tag{8.37}$$

The logic implementation of the Input Stage module is shown in Figure 8.33(a) and the corresponding leaf module environment specification in part (b) of the figure. The logic gates are limited to four inputs for faster operation [WeEs85].

8.7.10 Exponent Adder

This is an 8-bit ripple carry adder partitioned into 1-bit full adder functional modules (ExpAddbit), as shown in Figure 8.34(a). No carry bypass circuit is employed because its benefits are very limited over the short word lengths of the operands, as was also the case with the Exponent Subtracter and the Exponent Incrementer modules of the FPA system. Instead, buffers are used after the first four adder stages to improve the speed of the carry signal path.

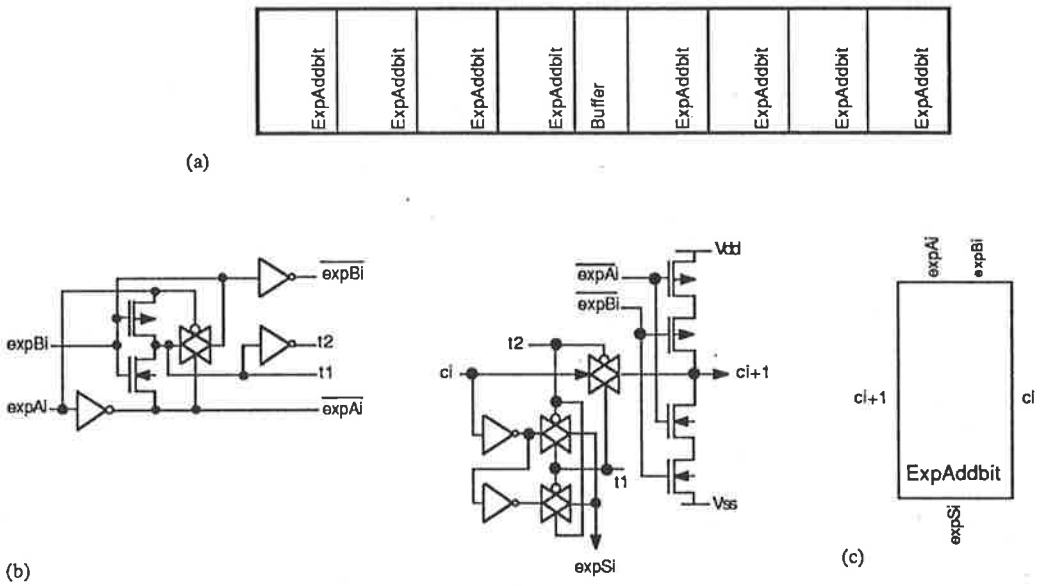


Figure 8.34: Exponent Adder: (a) floorplan, (b) ExpAddbit module logic implementation, (c) ExpAddbit module environment specification.

The logic implementation (obtained by suitably modifying the ExpDiffbit module shown in Figure 7.14) of the ExpAddbit module is shown in Figure 8.34(b) and the leaf cell environment specification in Figure 8.34(c).

8.7.11 Bias Subtracter

The implementation of this module is exactly the same as the implementation of the Exponent Adder module, except that one of its 8-bit inputs is hard wired to the two's

complement representation of -127. The other input is derived from the output ($expS$) of the Exponent Adder module.

8.7.12 Exponent Incrementer

This is an 8-bit incrementer module that increments by one the output from the Bias Subtracter module. Its implementation is the same as the implementation of the Exponent Incrementer module in the FPA system (Section 7.7.13.2).

8.7.13 Flags Generator

Two sets of flags corresponding to the incremented and not incremented versions of the intermediate resultant exponent are generated, the appropriate set being chosen by the Output Stage module subject to the control signal ($ExpOvf$) generated by the Mantissa module. In this way no additional time delay is incurred due to the flags generating logic since the flags are available prior to the evaluation of the $ExpOvf$ control signal.

A result overflow ($expR \geq 255$) occurs, assuming no mantissa overflow, when the most significant bits, $expA_8$ and $expB_8$, of the operand exponents are both $= 1$ and the most significant bit of the quantity $expr1$ (where $expr1 = expA + expB - 127$) is $= 0$, or when $expr1 = 255$. A similar condition applies for a mantissa overflow case. The OVF flag is then given by the following logic expression:

$$OVF = OVF_1 \cdot \overline{ExpOvf} + OVF_2 \cdot ExpOvf, \tag{8.38}$$

where

$$OVF_1 = expA_8 \cdot expB_8 \cdot \overline{expr1_8} + expr1_8 \cdot expr1_7 \cdots expr1_2 \cdot expr1_1 \tag{8.39}$$

$$OVF_2 = expA_8 \cdot expB_8 \cdot \overline{expr1_8} + expr2_8 \cdot expr2_7 \cdots expr2_2 \cdot expr2_1. \tag{8.40}$$

According to the above expressions $OVF = OVF_1$ when there is no mantissa overflow, indicated by $ExpOvf = 0$, and $OVF = OVF_2$ when mantissa overflow occurs, ie $ExpOvf = 1$.

A result underflow ($expR \leq 0$) occurs, assuming no mantissa overflow, when the most significant bits, $expA_8$ and $expB_8$, of the operand exponents are both $= 0$ and the most significant bit of the quantity $expr1 = 0$, or when $expr1 = 0$. A similar condition applies for a mantissa overflow case. The UNF flag is then given by the following logic expression:

$$UNF = UNF_1 \cdot \overline{ExpOvf} + UNF_2 \cdot ExpOvf, \tag{8.41}$$

where

$$UNF_1 = \overline{expA_8} \cdot \overline{expB_8} \cdot expr1_8 + \overline{expr1_8} \cdot \overline{expr1_7} \cdots \overline{expr1_2} \cdot \overline{expr1_1} \quad (8.42)$$

$$UNF_2 = \overline{expA_8} \cdot \overline{expB_8} \cdot expr1_8 + \overline{expr2_8} \cdot \overline{expr2_7} \cdots \overline{expr2_2} \cdot \overline{expr2_1} \quad (8.43)$$

According to the above expressions, $UNF = UNF_1$ when there is no mantissa overflow ($ExpOvf = 0$), and $UNF = UNF_2$ when mantissa overflow occurs ($ExpOvf = 1$).

The result zero flag (Z) indicates that the delivered value of the result is zero, ie $expR = 0$ and $fracR = 0$, and it is given by:

$$Z = zeroA + zeroB + UNF. \quad (8.44)$$

When one (or both) of the operands is infinity or NAN the final result is the Standard's representation of infinity, ie $expR = 255$, $fracR = 0$. The control signal $Exp1$, indicating that the constant 255 is to be delivered as the resultant exponent, is given by:

$$Exp1 = invA + invB + OVF. \quad (8.45)$$

When one (or both) of the operands is zero or denormalized the final result is the Standard's representation of zero, ie $expR = 0$, $fracR = 0$. The control signal $Exp0$, indicating that the constant 0 is to be delivered as the resultant exponent, is given by:

$$Exp0 = zeroA + zeroB + UNF. \quad (8.46)$$

This definition of $Exp0$ produces, however, an ambiguous result when one of the operands is zero (or denormalized) while the other is infinity (or NAN). In such cases the desired (according to the Standard) result is a quiet NAN. Since it has been decided in this work to treat the quantities which are NAN in the same way as infinity, the Standard's representation of infinity is generated by the FPM system. Expression (8.46) is therefore modified to read:

$$Exp0 = (zeroA + zeroB + UNF) \cdot \overline{Exp1}. \quad (8.47)$$

The control signal $ManZ$, indicating that the constant 0 is to be delivered as the fractional field of the result, is given by the following logic expression:

$$ManZ = Z + Exp1. \quad (8.48)$$

This module is implemented with a combination of static complementary CMOS logic and pseudo-nMOS logic, as shown in Figure 8.35(a). The leaf module environment specification is shown in part (b) of the figure.

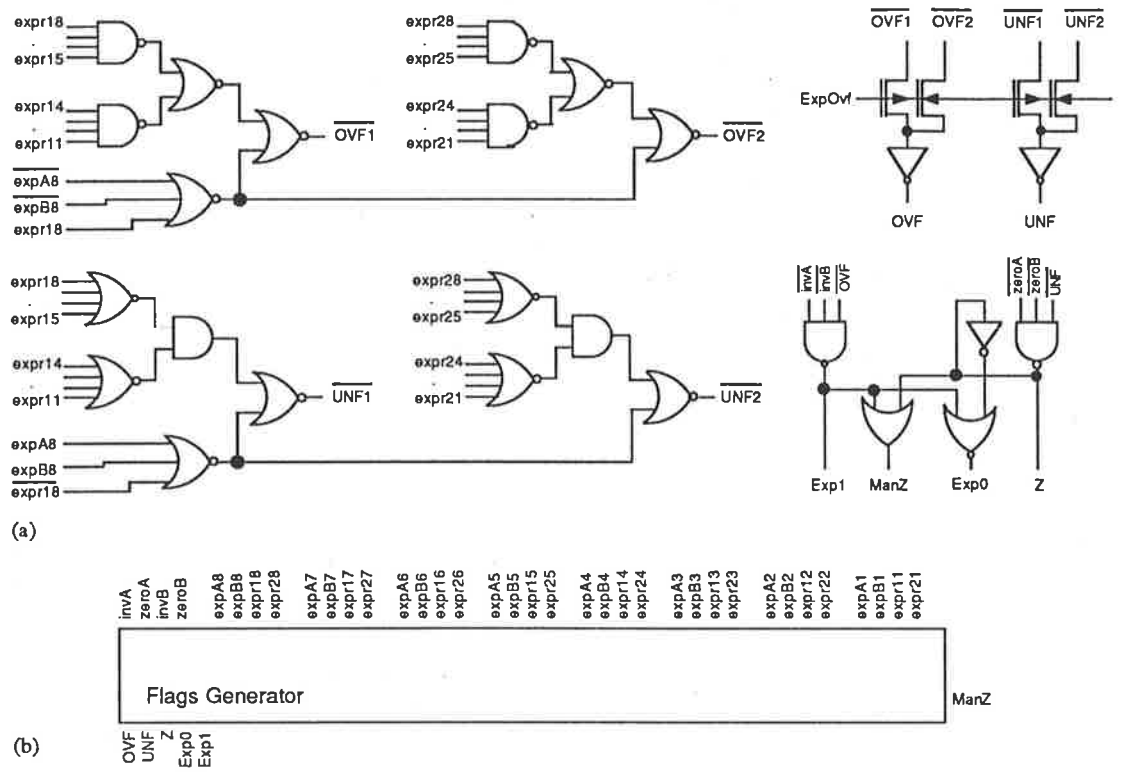


Figure 8.35: Flags Generator (a) logic implementation, (b) leaf module environment specification.

8.7.14 Output Stage

The function of this module is to select the correct version (either $expr1$ or $expr2$) of the resultant exponent or to generate the constant 255 or 0. It is implemented as a bit sliced 8-bit 4:1 multiplexer realized using the pseudo-nMOS circuit design style, as shown in Figure 8.36.

8.7.15 Sign

The sign of the result (SR) is given by the exclusive-OR logic function of the signs of the two operands. This functional module is included in the Input Stage module.

8.8 Final Floorplan

Having described the implementation of all modules of the FPM system, the reasons for generating the floorplan of Figure 8.37 become evident. It has been stated in Sec-

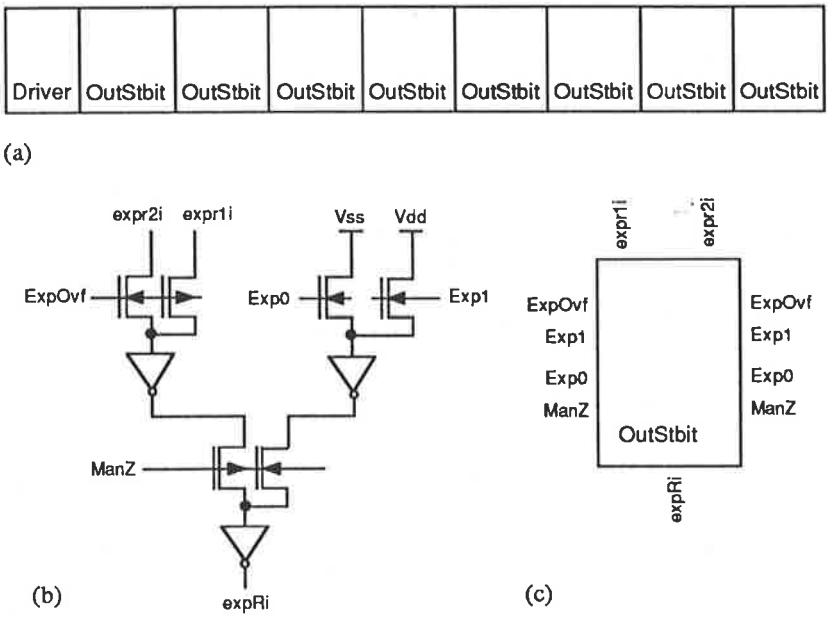


Figure 8.36: Output Stage: (a) floorplan, (b) OutStbit functional module logic implementation, (c) OutStbit functional module environment specification.

tion 8.7 that the floorplan of the complete FPM system was strongly influenced by the implementation of the Mantissa Multiplier module. This can be clearly seen from Figure 8.37.

The realization of the DS multiplier, used for the implementation of the Mantissa Multiplier (see Figure 8.21), is such that there is an unoccupied space in the bottom left hand side corner of the module's bounding box. In addition, the width of this free space is extended by the presence of the ManRndA, ManRndB and Mantissa Selector modules. The Exponent module is placed therefore in that area, eliminating the initially present "dead" space in the bounding box of the Mantissa module.

8.9 Design Evaluation

The area requirements and the speed performance of the Floating Point Multiplier (FPM) are compared to those of the conventional hardware realization (CFPM) of floating point multiplication.

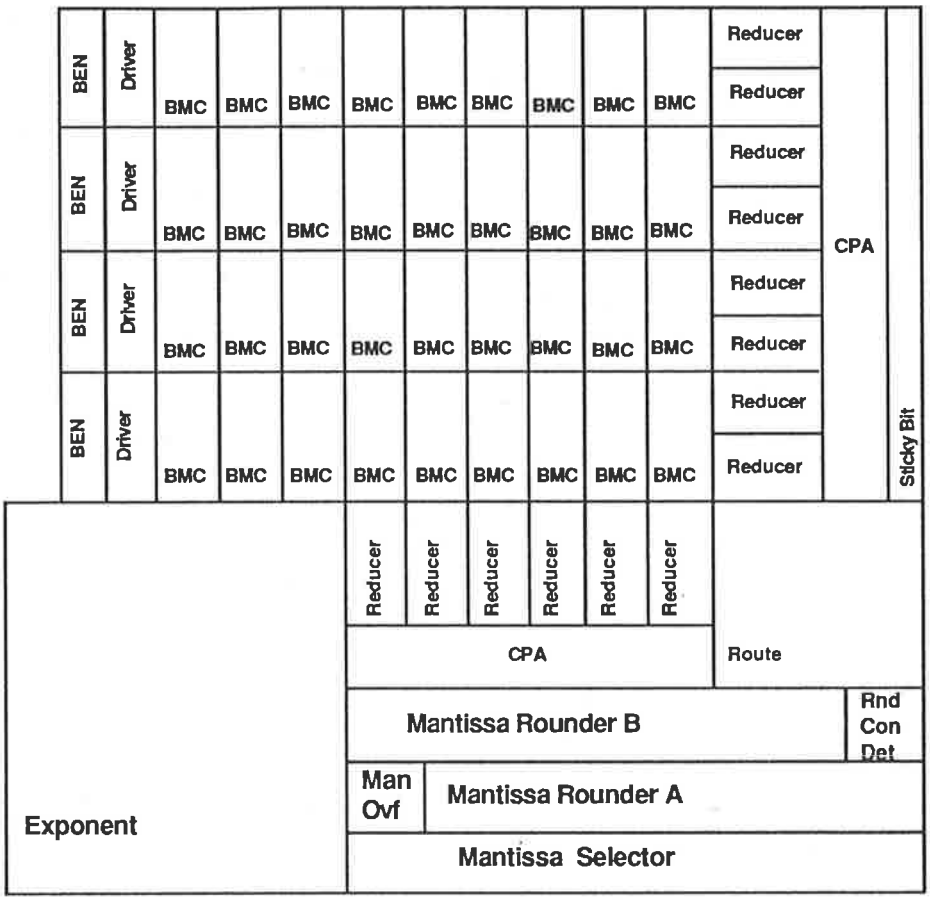


Figure 8.37: Floating Point Multiplier: floorplan

8.9.1 Area Requirements

The complete plot, in the symbolic domain, of the reduced version of the FPM system is shown in Figure 8.38. It corresponds exactly to the floorplan of Figure 8.37. A full compaction using the Phase2 compactor, SYMPACK, was performed according to the design rules of the hypothetical $Orbit^+$ process, defined in Section 4.6. The dimensions of the overall bounding box were found to be $L_{FPM} = 1582\lambda$ and $W_{FPM} = 1604\lambda$. From this result the dimensions of the 32-bit bit version, ie a 24-bit mantissa and an 8-bit exponent, of the FPM system are estimated to be $L_{FPM} = 3552\lambda$ and $W_{FPM} = 3260\lambda$, or $L_{FPM} = 1.78 \times 10^{-3}m$ and $W_{FPM} = 1.63 \times 10^{-3}m$ ($\lambda = 0.5 \times 10^{-6}m$), giving an effective area of $2.90 \times 10^{-6}m^2$.

The bounding box of the corresponding hardware implementation (CFPM) of the conventional algorithm would have the same length $L_{CFPM} = 3552\lambda$ but the width of the bounding box would be smaller by approximately the width of the bounding box of one Mantissa Rounder (ManRnd) module, ie $W_{CFPM} = 3260\lambda - 284\lambda = 2976\lambda$.

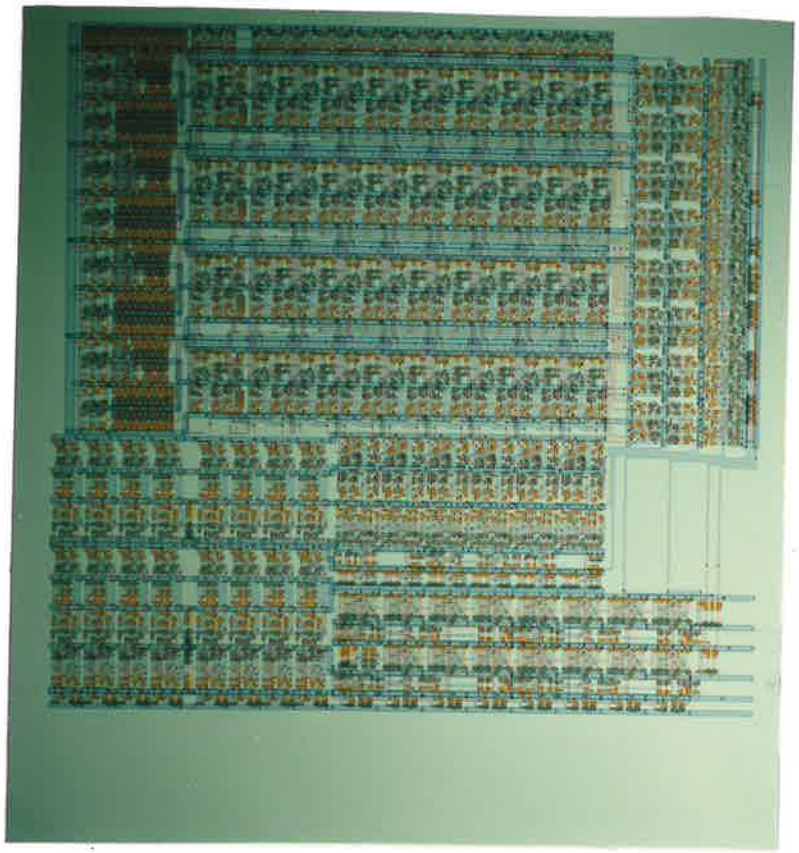


Figure 8.38: Floating Point Multiplier: symbolic domain representation.

Set No.	manA	expA	manB	expB	manR	expR
set 1	01111111	01111111	01000001	01111111	1000000	10000000
set 2	01111111	01111111	01000000	01111111	1111111	01111111
set 3	01111111	01111111	01000001	01111110	1000000	01111111
set 4	01111110	01111111	01000001	10000000	1000000	01111111

Table 8.3: Simulation test vectors for the FPM system.

Thus, the new architecture of the floating point multiplier requires an extra silicon area of $3552\lambda \times 284\lambda$, or approximately 9.5%.

8.9.2 Speed Performance

The performance of the FPM system was investigated using circuit simulations on the reduced system, and also by simulating the critical path of the full 32-bit version of the design. In total, 40 different sets of test vectors were applied to the reduced system with

correct result. A simulation run with one set of vectors on a VAX785/VMS[1] machine took on average of 24 cpu minutes.

Four sets of results are shown in two figures 8.39 and 8.41, with the corresponding test vectors given in Table 8.3. Results from two sets of test vectors are presented in each figure. In Figure 8.39 the simulation time corresponding to set 1 of the input vectors is $0ns$ to $50ns$ with the time $50ns$ to $100ns$ corresponding to set 2 of the input vectors. The simulation results for set 3 and set 4 of the input vectors are similarly presented in Figure 8.41.

As discussed previously, in the reduced system, the 7-bit resultant mantissa ($manr$) corresponds to bits 7 to 13 ($pr_7 - pr_{13}$) of the Mantissa Multiplier module output. The most significant bit of the full mantissa product (pr_{14}) is denoted $manr_8$ and it corresponds to the most significant bit of an overflowed resultant mantissa.

Example 1

The operation of the circuit will be discussed on the basis of the results presented in Figure 8.39. Set 1 of the test vectors ($t = 0ns$ to $t = 50ns$) is such that a carry $= 1$ is generated in the module CPA in the bit position corresponding to $manr_1$, and it propagates to the most significant bit position ($manr_8$) of that module. The last bit ($manr_6$, waveform manr6) of the output of the Mantissa Multiplier stabilizes at $t = 33ns$. The propagation of the carry signal becomes evident by examining the waveforms on nodes $manr_1$ to $manr_8$. Bits $manr_7$ and $manr_8$ stabilize earlier than bit $manr_6$, as a result of the carry Bypass module being present across bit positions $manr_3$ to $manr_6$ of the CPA module. The two control signals, $RndA$ and $RndB$, are generated early (at $t = 22ns$), as a result of the least significant part of the mantissa product (bits pr_1 to pr_6) becoming stable much earlier than bits pr_7 ($= manr_1$) to pr_14 ($= manr_8$). Thus, the modules ManRndA and ManRndB operate in parallel with the module CPA.

Once the most significant bit of the overflowed mantissa product ($manr_8$) has stabilized ($t = 29ns$), the control signals $ManOvf$ and $ExpOvf$ can be evaluated (expressions (8.33) and (8.34)). $ManOvf$ becomes available at $t = 30ns$ and $ExpOvf$ becomes available at $t = 31ns$. These signals define the outputs of the modules Mantissa Selector and Output Stage. The final resultant mantissa stabilizes at $t = 36ns$ (waveform fracR5) and the resultant exponent stabilizes at $t = 33ns$ (waveform expR8).

[1]VAX and VMS are trademarks of Digital Equipment Corporation

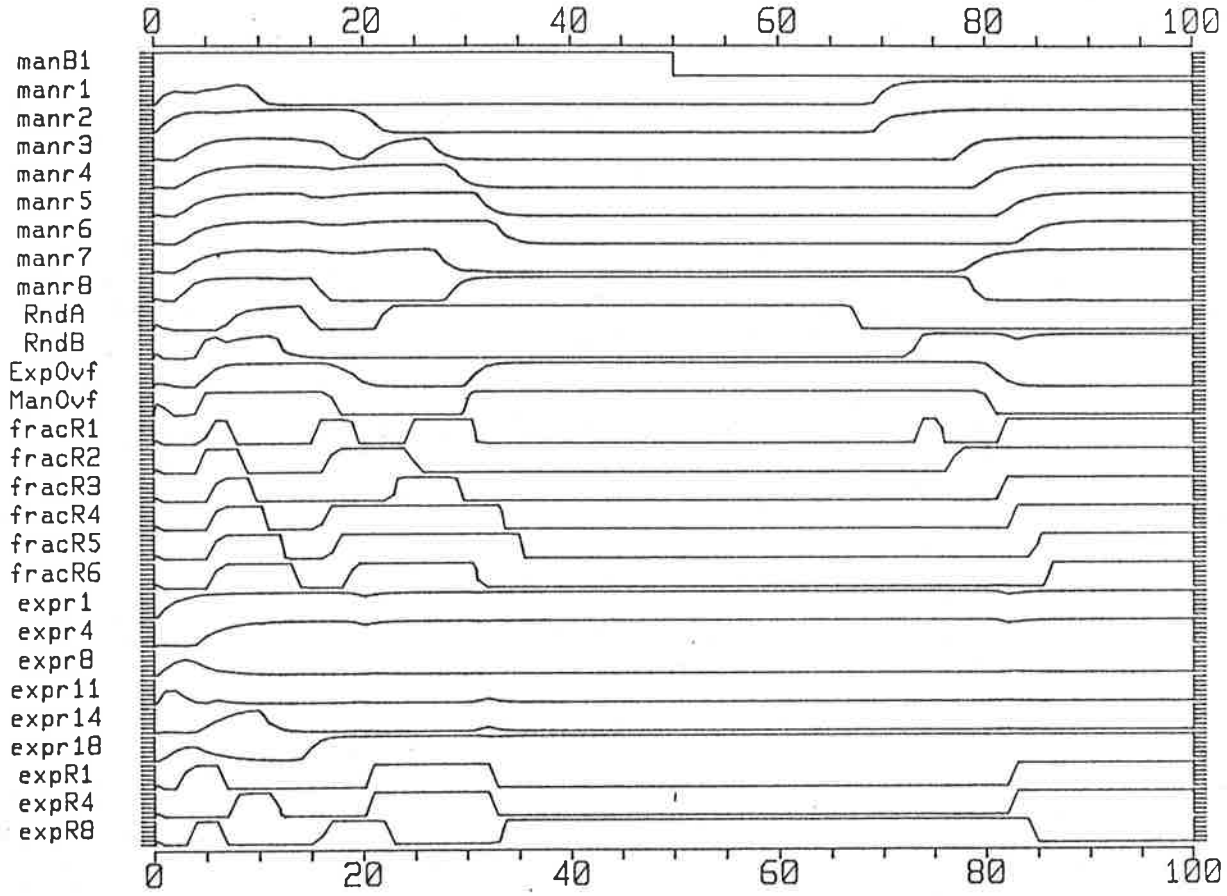


Figure 8.39: Floating Point Multiplier: FACTS simulation results corresponding to set 1 and set 2 of the input vectors given in Table 8.9. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

Note that for this set of the input vectors mantissa multiplication results in overflow ($manr_8 = 1$), requiring that the intermediate value of the exponent be incremented by one. The process of exponent incrementing results in a carry = 1 being propagated across the complete length of the Exponent Incrementer. However, since in this design both possible versions of the resultant exponent are evaluated simultaneously, the process of exponent incrementing does not introduce any additional delay, as intended. In fact the resultant exponent becomes available *before* the resultant mantissa becomes available. Note also that, as intended, the resultant mantissa (waveform fracR5) stabilizes only a small delay after the mantissa product (waveform manr6) becomes stable. This small delay, as discussed previously, is approximately equal to the time it takes for the signal to propagate through a 1-bit incrementer module and the 1-bit functional module of the Output Stage, and this delay is *independent* of the operand word length.

The delay of the FPM system for set 1 of the input vectors given in Table 8.3 can be

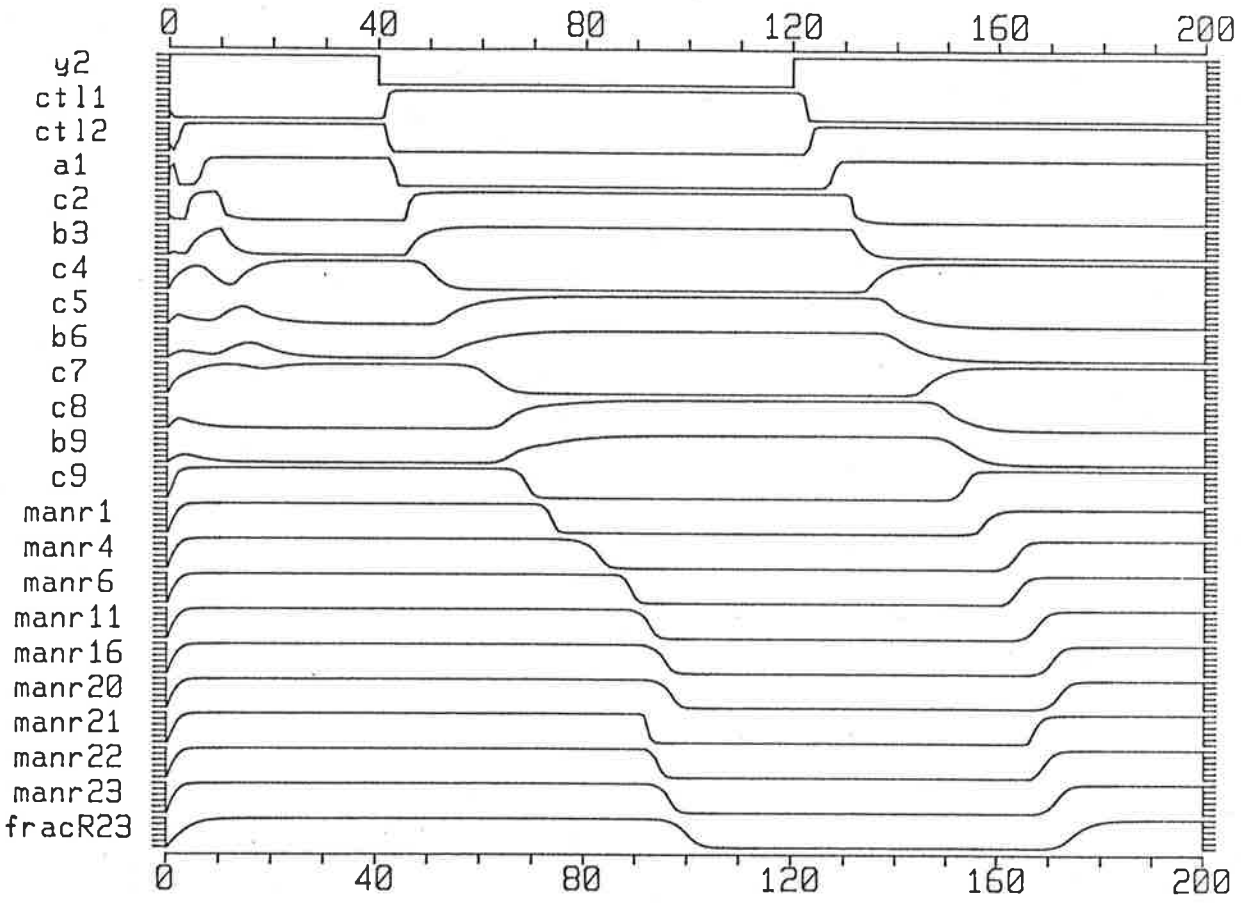


Figure 8.40: Floating Point Multiplier: FACTS simulation results of the critical data signal path of the 32-bit FPM system. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

expressed in the form:

$$\begin{aligned} t_{FPM,set1} &= t_{mult} + t_{round} + t_{mux}, \\ &= t_{mult} + t_{const}, \\ &= 36ns. \end{aligned} \tag{8.49}$$

Where t_{mult} is the time required to evaluate the mantissa product ($t_{mult} = 33ns$), t_{round} is the time delay introduced by the 1-bit functional module of the Mantissa Rounder module and t_{mux} is the time delay introduced by the Output Stage module, ie $t_{const} = t_{round} + t_{mux} = 3ns$.

The method of extrapolating the delays obtained from the circuit simulations of the reduced system to the delays of the full 32-bit system is based on the simulation results of the critical path of the 32-bit FPM system, shown in Figure 8.40. The signal path of interest is composed of the following modules: BEN, PPG, 8 stages of FA, 24-bit CPA, 24-bit ManRnd and ManSelbit. A consideration of the slowest output of the 24-bit CPA module (waveform manr23) gives an estimate of the delay of the 24-bit Mantissa

Multiplier module of $t_{mult} = 96ns - 40ns = 56ns$ (the inputs are applied at $t = 40ns$). The delay $t_{const} = 3ns$ is obtained by taking the difference between the time ($t = 99ns$) at which the most significant bit of the fractional field of the result (fracR23) and the time ($t = 96ns$) at which the most significant bit of the mantissa product (waveform manr23) becomes stable. The overall delay of the 32-bit FPM system for set 1 of the input vectors is therefore $t_{FPM} = 59ns$.

The delay of the corresponding 32-bit CFPM system is established as follows. In the CFPM system the calculation of the resultant exponent commences when the signal *ExpOvf* (also given by (8.34)) becomes stable. This happens as soon as the most significant bit ($manr24$), indicating mantissa overflow, becomes stable ($t_{mult} = 56ns$). Thus, the delay of the CFPM system, for this case of the input vectors, is approximately given by:

$$\begin{aligned} t_{CFPM,1} &= t_{mult} + t_{mux} + t_{exp} \\ &= 56ns + 1ns + 18ns = 75ns. \end{aligned} \tag{8.50}$$

For this set of the input vectors the FPM system performs approximately 21% better that the corresponding CFPM system.

Example 2

The maximum speed advantage of the FPM system over the CFPM system is demonstrated with set 2 of the input vectors (Figure 8.39 for $t = 50ns - 100ns$). The internal state of the circuit prior to the application of this set of inputs is defined by the final state of the circuit corresponding to the first set of input vectors.

In this case a carry = 0 is propagated across the CPA module. The control signals *ManOvf* and *ExpOvf* stabilize at $t = 81ns$ and $t = 82ns$, respectively. The resultant exponent (waveform expR8) becomes available at $t = 84ns$, ie shortly after *ExpOvf* stabilizes. (The time difference $84ns - 82ns$ corresponds to the time delay of the the Output Stage module). The mantissa product is rounded with the rounding operation resulting in an overflow. The resultant mantissa becomes available at $t = 87ns$, a delay $t_{const} = 3ns$ after the mantissa product (waveform manr6) becomes available. The resultant exponent again stabilizes before the time at which the resultant mantissa stabilizes. Interpolating these results to the 32-bit FPM system gives:

$$\begin{aligned} t_{FPM,set2} &= t_{mult} + t_{const}, \\ &= 56ns + 3ns, \\ &= 59ns. \end{aligned} \tag{8.51}$$

The serial nature of the CFPM system becomes apparent in this case. The mantissa product overflowed for set 1 of the input vectors, thus initially *ManOvf=1* and, since this signal is also used to normalize an overflowed mantissa product, the mantissa product is initially shifted one bit position to the right. As a result, an incorrect value of the rounding control signal is initially determined, ie initially $manr_1 = manr_2 = 1$ causing $Rnd = 1$, resulting in the mantissa product being incremented. This initial mantissa incrementing results in an mantissa overflow, causing the resultant exponent to be incremented by one.

The correct value of $ManOvf\,(= 0)$ and the correct rounding condition $(Rnd = 0)$ are determined, when the output of the CPA module becomes stable $(t = 84ns)$. This then causes the correct value $(= 0)$ of the carry signal to propagate across the complete length of the mantissa rounding module, and presumably in parallel the resultant exponent is also generated.

The 32-bit CFPM system, therefore, experiences a time delay given by:

$$\begin{aligned} t_{CFPM,set2} &= t_{mult} + \max(t_{rnd} + t_{round}, t_{exp}), \\ &= 56ns + 3ns + 22ns = 81ns. \end{aligned} \tag{8.52}$$

The FPM system is approximately 27% faster than the corresponding CFPM system, at a cost of an additional silicon area of approximately 10%.

8.10 Summary

A novel VLSI implementation of a floating point multiplier circuit (the FPM system) has been discussed. An analysis of the conventional algorithm led to the formulation of the new architecture which operates with a higher degree of parallelism than the commonly adopted implementations. A maximum time delay reduction of approximately 27% has been achieved, at the expense of an approximate 10% increase in the silicon area. Circuit simulation results, obtained using the modified FACTS program, indicate that the 32-bit version of the circuit fabricated with a CMOS 1-micrometer process will have a maximum delay of approximately $59ns$. The new architecture experiences a time delay which is only a constant time delay larger than the corresponding time delay of the integer multiplier module.

Two new fixed point multiplier architectures have also been described. One being a modification of the modified Booth algorithm multiplier, where the signals propagate across two parallel streams of Full Adder modules. This modification results in the

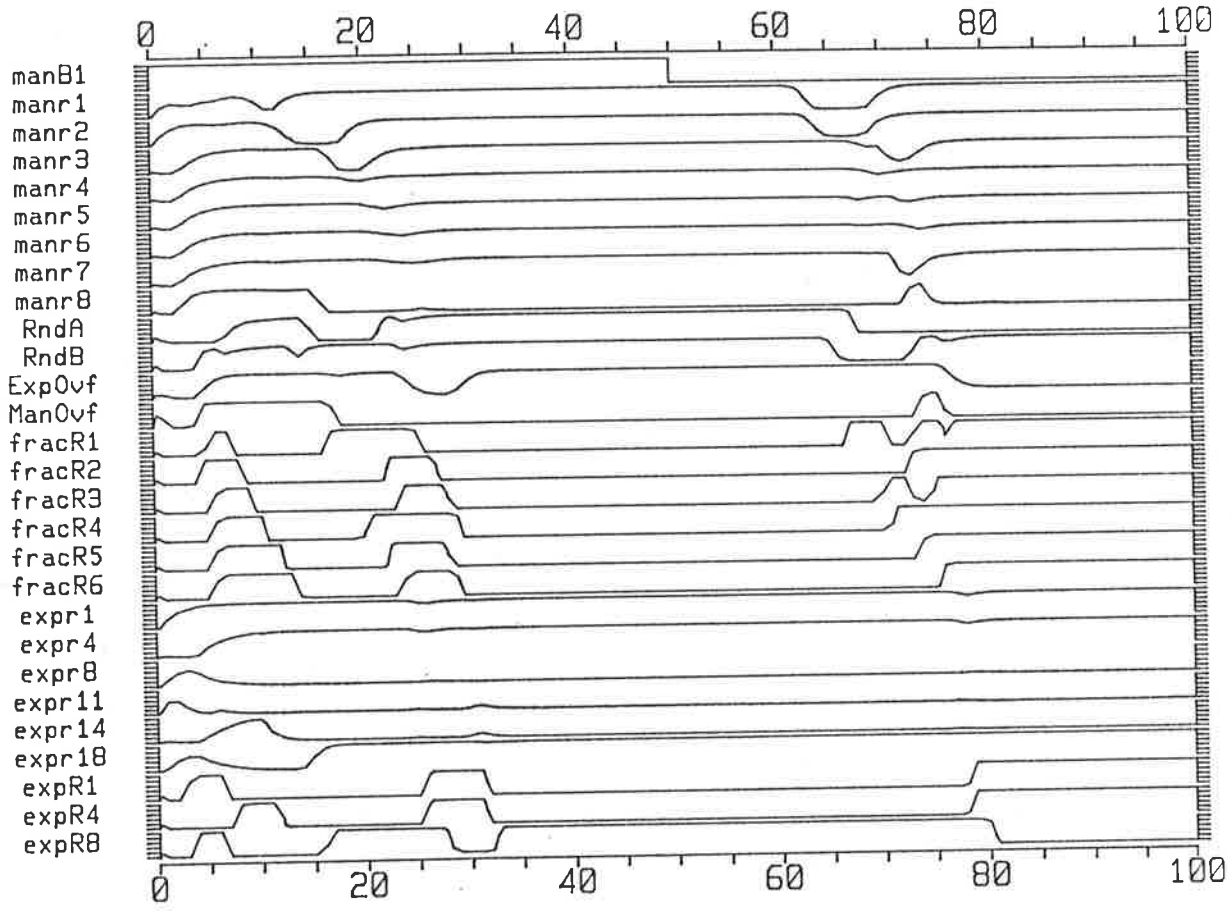


Figure 8.41: Floating Point Multiplier: FACTS simulation results corresponding to set 3 and set 4 of the input vectors given in Table 8.9. Time is in nanoseconds and the range of the output waveforms is 0V to 5V.

critical signal path, within the multiplier array, containing half as many Full Adder modules, as in the modified Booth algorithm multiplier. The multiplier still has time complexity $O(n)$ and area complexity $O(n^2)$. The other multiplier architecture results in a reduction of approximately 50% in the silicon area requirements of the multiplier design proposed in [Vuil83] in which the basic multiplier cells, each composed of two Full Adder modules, are connected as a binary tree. This area saving has been achieved by modifying the interconnections between some modules, resulting in the lengths of the adders at all levels of the binary tree being constant and equal to n, the operand length. No addition time delay is introduced by the proposed architectural modification. The maximum delay of this multiplier is within one Full Adder delay of the Wallace tree multiplier and its regular structure allows a much more regular and easily expandable layout compared to the Wallace tree multiplier.

Chapter 9

Summary and Conclusions

With the large number of transistors ($\approx 10^5$) provided on a single silicon chip by present day VLSI technology, it is possible to realize high performance, application specific systems. The high performance is made possible by the large degree of concurrency that VLSI circuits can exhibit. The main objective of the research work described in this thesis has been to improve the efficiency of certain computations, by designing architectures that have a higher degree of parallelism than is currently present in current systems used for these computations.

The particular task that has been examined herein is that of numerical MOS transistor modelling, requiring a solution of three partial differential equations, commonly known as the basic semiconductor equations. This is a task task requiring large computer resources both in terms of cpu performance and memory storage capabilities, and is characterized by the need to perform large number of floating point arithmetic operations. To a large degree, the computations involved in the solution of these equations are highly regular and may be carried out with a high degree of parallelism on a two-dimensional array processor composed of identical PEs.

An examination of the numerical algorithms, such as the SOR, and the Chebychev and the conjugate gradient acceleration methods, which need to be employed in order to achieve acceptable convergence performance for three-dimensional device simulators to be practical tools, has shown that a class of computations involved in such algorithms cannot be carried out efficiently on the two-dimensional processor array. These computations involve the calculation of the stopping test an adaptive estimation of the spectral radius of the iteration matrix, which quantity is needed for the calculation of the optimum relaxation parameter or Chebychev acceleration parameters. The common factor in the abovementioned calculations, as well as in the calculation of iteration parameters

in the conjugate gradient method, is the evaluation of a vector inner product.

A vector inner product can be evaluated with a larger degree of parallelism on the TIME Machine, an architecture proposed in this thesis, than on a two-dimensional array processor. The new parallel processor architecture is composed of a two-dimensional array of PEs, augmented by a number of additional PEs embedded within the array, and connected as a binary tree. The number of extra processors required is only 6.25% of the number of PEs within the two-dimensional array. The new processor architecture efficiently combines the characteristics of the two-dimensional array processor and the binary tree processor, making it a more flexible architecture than either of the two structures.

On the TIME Machine, it is possible to calculate the sum of n_p numbers (a computation that forms the main part of a vector inner product calculation), stored one per each PE within the two-dimensional array, at a cost of $\log_2 n_p$ additions and $\log_2 n_p$ move operations, compared to $\log_2 n_p$ additions and n_p move operations required on the conventional two-dimensional array processor.

At the chip level the logical and physical architectures of a single chip PE, the Mathematical Processor, have been described. The logical architecture includes full hardware implementation of a floating point adder and a floating point multiplier, and two bus switches operating under the control of the program. These switches provide means of increasing the parallelism at the chip level, by allowing the setting up of the necessary data paths (including data feedback paths) between the data handling elements on the chip, for optimum execution of computational tasks required of the mP, such as the sum of products computation, vector by scalar multiplication and the evaluation of first order recurrence relations associated with an efficient solution of tridiagonal systems of linear equations. Area estimations of a number of the largest modules indicate that the Mathematical Processor should be realizable in 1-micrometer, double metal CMOS technology on a square chip of $10^{-2}m$ side.

A major problem that has been completely dealt with in this thesis is the design of the Floating Point Adder and the Floating Point Multiplier modules, both of which constitute two most complex functional blocks of the Mathematical Processor. The conventional algorithms for these two operations have been analysed and modified, and as a result novel VLSI architectures have been presented that perform a number of tasks, normally executed in a sequential manner, concurrently, thus, increasing the speed with which the final results are obtained. Both these circuits have been implemented in the symbolic domain using static combinatorial CMOS and pseudo-nMOS circuit design techniques, in order to evaluate the new architectures at the transistor level and compare

them to the conventional architectures of such circuits.

A substantial part of the Floating Point Adder system has been fabricated together with a number of ring oscillators, using a 2-micrometer CMOS process. Using the experimental results obtained from these fabricated circuits, together with circuit simulations, the performance of the Floating Point Adder and Multiplier have been projected to a 1-micrometer CMOS process. In the case of the Floating Point Adder the maximum delay of $66ns$ has been achieved. This represents a 36% decrease on the maximum delay of a corresponding circuit implementing the conventional floating point addition algorithm. In the case of the Floating Point Multiplier the maximum delay was found to be $59ns$, and this represents a 27% decrease on the maximum delay of the corresponding circuit implementing the conventional form of the floating point multiplication algorithm. These performance increases have been achieved at a cost of approximately 10% increase in the silicon area, in both circuits.

As part of the Floating Point Multiplier architecture, two novel integer parallel multiplier designs also have been proposed. The DS multiplier is based on the design given by Pezaris. The improvement in performance has been achieved, as a result of increased parallelism within a carry save multiplier architecture obtained by interconnecting full adder modules in such a way that two, instead of one, signal paths are available within the multiplier array. This effectively halves the number of full adder stages within the critical signal path. By also incorporating the modified Booth algorithm, the number of full adder stages is decreased by a further factor of two.

The second multiplier design is based on the multiplier proposed by Vuillemin, which performs the multiplication in time $O(\log_2 n)$. The proposed architecture includes a modification to module interconnections which allow the adders at all levels of addend summation to be of equal length, resulting in a decrease in the silicon area of $\approx 50\%$ over the original architecture. No performance penalty is incurred by this modification and the area saving is achieved at the expense of increased complexity of module interconnection.

The successful designs of these two complex (although not VLSI scale) floating point arithmetic systems has been achieved by the application of a design methodology that effectively manages design complexity. This methodology has been based on the structured design methodology that employs techniques of hierarchical decomposition, abstraction, regularity and modularity. The available software tools also have had an influence in formulating the design methodology employed in this work. In particular it has been demonstrated how, in the opinion of this author, the best use can be made of symbolic design tools such as VIVID or Phase 2, and how these can be effectively aided

by the use of a functional simulator such as USA.

The highly accurate numerical models of MOS transistors cannot be employed in circuit simulation programs, such as FACTS. Such programs still use analytical transitor models for computational efficiency. Because of the poor accuracy of the analytical MOS transistor models initially embedded within the FACTS program, it was necessary to develop an improved model in order for the program to be of useful value in the undertaken design work.

An analytical MOS transistor model, with geometry dependent parameters, has been proposed which predicts the current-voltage characteristics of MOS transistors with a wide range of channel lengths and widths for a particular fabrication process. The main characteristic of this model is that once technology dependent parameters have been determined all that is necessary to predict the current-voltage characteristics of a given transistor are the channel length and width. In addition, this model incorporates a novel expression for the threshold voltage of a short channel MOS transistor, which expression has been derived by combining the two most commonly used approaches in deriving such expressions, ie the approach based on the a priori charge sharing concept and the approach based on the a priori potential distribution assumption. The model accuracy has been demonstrated by comparing its predictions with the experimental measurements performed on a number of test transistors fabricated with two different processes. The analytical model has been incorporated into the FACTS program by using a table took-up technique combined with linear interpolation. This resulted in an accurate circuit simulator capable of simulating digital circuits of up to 10000 transistors. The FACTS simulator has been extensively used to evaluate and validate the designs of this work.

As the active transistor dimensions are further reduced, higher functional densities on a chip are made possible which allow for larger and more complex digital systems to be designed. At the same time the single transistor becomes a more complex device itself. As a result, the demand for cpu performance, memory requirements and throughput, in the development of VLSI chips will continue to increase. However, while the discipline of VLSI design produces the need for supercomputers, it is the design of new and high performance VLSI architectures that can provide this computing power. The work reported in this thesis has attempted to make a contribution towards realizing such systems.

Appendix A

The Transform and Filter Brick

The design of the Transform and Filter Brick (TFB) chip has been undertakes as a group project by postgraduate and undergraduate students in the Department of Electrical and Electronic Engineering at Adelaide University, under the supervision of Dr. Kamran Eshraghian. Various members of the team were responsible for different aspects of the project. Broadly speaking the logical architecture was specified by David Fensom, Rod Bryant and Bill Cowley, the physical architecture was specified by Alex Dickinson, John Rockliff and the author, the control store and the decoders were designed by Michael Pope and the execution controller was designed by Paul Franzon who was also responsible for the work on yield analysis and redundancy. John Rockliff was responsible for the testability issues [Rock86]. The input-output processors were designed by Eddy Savio [Savi84] and the Data Memory pointers and buffers were designed by Neil Murry [Murr84] both under the supervision of the author of this thesis. The design of the multiplier-dividers [ZyEs85], and of the arithmetic-logic unit were the sole responsibility of the author.

Although the complete chip was never finished, the project provided a vehicle, for the author, for gaining initial knowledge and experience in most areas of VLSI design. The most important outcome of the project, for the author, was the original design of the multiplier-divider circuit briefly described in both papers included in this appendix. The first paper, to follow, gives a general overview of the TFB chip and the second paper gives a brief description of the multiplier-divider circuit.

Eshraghian, K., Bryant, R.C., Dickinson, A., Fenson, D.S., Franzon, P.D., et al., (1986) The transform and filter brick: a new architecture for signal processing. In Hörbst, E., (ed.), *VLSI 85: VLSI Design for Digital Systems*, North-Holland, pp. 135-144.

NOTE:
This publication is included in the print copy
of the thesis held in the University of Adelaide Library.

Eshraghian, K., and Zyner, G., (1985) Multiplier/divider designs for VLSI signal processing system.
20th International Electronics Convention & Exhibition (IREECON International), Melbourne, Victoria, pp. 253-256.

NOTE:
This publication is included in the print copy
of the thesis held in the University of Adelaide Library.

Appendix B

Model Parameters Extraction

B.1 Determination of V_{th}, N_A and V_{FB}

The threshold voltage of a transistor may be determined from the I_{DS} versus V_{GS} characteristics measured at a low drain bias, eg $V_{DS} \approx 0.1V$. The value of the threshold voltage is obtained by extrapolating the straight line segment of such a curve to the horizontal axis (ie $I_{DS} = 0$). The abscissa intercept is approximately given by $V_{th} + V_{DS}/2$ from which V_{th}, "the extrapolated threshold voltage", is found. This is the most common definition of the threshold voltage used in literature. This procedure is typically performed on a number of transistors of the same mask dimensions and the average value taken.

The body effect parameter (K) given by (B.2) is obtained by using threshold voltage values, determined as described above for a large geometry transistor, corresponding to two different substrate bias voltages. Assuming an initial estimate for $2\phi_F = 0.6V$ the value for K is calculated from:

$$K = \frac{V_{th}(V_{SB1}) - V_{th}(V_{SB2})}{\sqrt{2\phi_F + V_{SB2}} - \sqrt{2\phi_F + V_{SB2}}}. \tag{B.1}$$

Using this value for K an estimate for the effective substrate doping N_A is calculated from the definition:

$$K = \frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}}, \tag{B.2}$$

which is then used to find an improved estimate for ϕ_F. This procedure converges after two or three iterations. An example of this calculation for the P1 process (described in Section 4.4) is presented in Table B.1. The transistor with the largest available mask dimensions ($L_m = 25 \times 10^{-6} m$ and $W_m = 25 \times 10^{-6} m$) was used for this calculation. The threshold voltage for $V_{SB} = 0$ was found to be $1.05V \pm 0.05$ and the threshold voltage

iteration index i	$2\phi_F^i$ $[V]$	K^i V^{-1}	N_A^i m^{-3}
0	0.600	1.508	8.62×10^{21}
1	0.704	1.549	9.10×10^{21}
2	0.707	1.550	9.10×10^{21}

Table B.1: Extraction of the body effect parameter K.

for $V_{SB} = 5V$ was found to be $3.45V \pm 0.04V$. (The value of the gate capacitance C_{ox} is normally obtained from capacitance-voltage measurements [Pier83]. This parameter was obtained directly from the fabrication house.)

The value of V_{FB} is then simply calculated from expression (3.32) applied to a large geometry transistor.

B.2 Determination of y_{S0}, σ, α and δ

The short channel effect according to expressions (4.12) and (4.13) is given by:

$$\Delta V_{TS} = \frac{y_{S0}}{L}\left(1 + \sigma V_{SB}^{1/2}\right)\frac{1}{C_{ox}}\left[2\epsilon_s q N_A(2\phi_F + V_{SB})\right]^{1/2}. \tag{B.3}$$

Dividing both sides by $[2\epsilon_s q N_A(2\phi_F + V_{SB})]^{1/2}$ gives:

$$\frac{\Delta V_{TS}}{[2\epsilon_s q N_A(2\phi_F + V_{SB})]^{1/2}} = \frac{y_{S0}}{LC_{ox}} + \frac{y_{S0}}{LC_{ox}}\sigma V_{SB}^{1/2}, \tag{B.4}$$

where $\Delta V_{TS} = V_{th}(V_{SB}) - V_{th}(V_{SB} = 0)$. A straight line is fitted (Note that linear regression is always employed where a straight line can be fitted through the experimental points) through the points obtained by plotting $\Delta V_{TS}[2\epsilon_s q N_A(2\phi_F + V_{SB})]^{-1/2}$ versus $V_{SB}^{1/2}$ for a number of transistors with large channel widths and various channel lengths. The value of y_{S0} is determined from the ordinate intercept and the parameter σ is obtained from the slope of the plot. The corresponding value for r_j can then be obtained, if desired, from (4.8). Using this procedure the value of r_j corresponding to the P1 process was found to be $= 1.85 \times 10^{-6}m$. This falls well within the range of values for this physical parameter supplied by the fabrication house, ie $1.7 \times 10^{-6}m - 1.9 \times 10^{-6}m$.

The narrow width effect is given by equation (4.11). The narrow width effect parameter δ_W is obtained from the slope of the plot $\Delta V_{TN} = V_{th}(W) - V_{th}(W = large)$ versus W^{-1} for a number of transistors with a constant large channel length (to minimize the short channel effect) and various channel widths.

The parameter α is determined from the slope of the plot of $\Delta V_{th} = V_{th}(V_{DS}) - V_{th}(V_{DS} = 0)$ versus V_{DS}. The threshold voltage at a given drain bias was determined in this case from the ordinate intercept of the linear portion of the plot of $\sqrt{I_{DS}}$ versus V_{GS} at saturation. By plotting $\ln \alpha$ against $\ln L$ it was found that the dependence of this parameter with geometry was of the form:

$$\alpha = \alpha_0 / L^{1.5}, \tag{B.5}$$

a result previously suggested in [Klas78] and [KlGr80].

B.3 Determination of μ^o, θ, L and W

For small values of the drain to source bias the expression (4.33) for the drain current in the linear mode reduces to:

$$I_{DS} = \frac{\mu^o \frac{W}{L} C_{ox} (V_{GS} - V_{th} - 0.5 V_{DS}) V_{DS}}{1 + \theta \left[V_{GS} - V_{th} - 0.5 V_{DS} + 2 \frac{\gamma}{C_{ox}} \sqrt{2 \epsilon_s q N_A (2\phi_F + V_{SB})} \right]}, \tag{B.6}$$

where all other terms are neglected. Expression (B.6) can be written as:

$$\begin{aligned} & \left[\frac{V_{DS}}{I_{DS}} (V_{GS} - V_{th} - 0.5 V_{DS}) \right], \\ = \; & \frac{1}{\beta_0} + \frac{\theta}{\beta_0} \left[V_{GS} - V_{th} - 0.5 V_{DS} + 2 \frac{\gamma}{C_{ox}} \sqrt{2 \epsilon_s q N_A (2\phi_F + V_{SB})} \right]. \end{aligned} \tag{B.7}$$

A plot of the left hand side of the above expression (performed using measured values of I_{DS}) against the quantity enclosed in the square brackets and on the right hand side of the above equation gives a straight line. The low field transistor gain $\beta_0 = \mu^o C_{ox} W/L$ is given as the reciprocal of the ordinate intercept and the mobility degradation parameter θ is determined from the slope using the relation $\theta/\beta_0 = slope$. A single straight line is obtained for a given transistor even when different values of V_{SB} are used in expression (B.7). This is in agreement with the findings of [WhWL80] that the carrier mobility decreases with increasing substrate bias. Note that expression (3.173) indicates that the carrier mobility increases with increasing substrate bias and its use in expression (B.6) results in a family of straight lines being produced, one for every substrate bias voltage which in turn gives a number of different values for β_0 for a single transistor. Clearly the mobility model given by expression (3.179) is the correct one. The two components making up θ, ie θ_0 and θ_1 (θ_2 was found to be negligible), are determined by plotting θ versus W/L for a number of transistors with various mask dimensions.

The *effective* (also known as electrical) transistor channel length (L) is calculated as a difference between the specified mask channel length (L_m) and a ΔL_m term combining

geometrical variations in the photolithographical process and lateral diffusion. From the definition of β_0 given earlier in this section:

$$L = L_m - \Delta L_m = \frac{\mu^o W C_{ox}}{\beta_0}. \tag{B.8}$$

Thus a plot of β_0^{-1} versus L_m for a family of devices with the same channel width is a straight line. ΔL_m is obtained as the abscissa intercept and the slope of the plot is given by $(\mu^o W)^{-1}$ from which the value for the low field mobility (μ^o) is determined.

The effective channel width (W) is calculated as the difference between the mask channel width (W_m) and a ΔW_m term due to geometrical variations in the photolithographical process. This parameter is determined as the abscissa intercept of the straight line plot of β_0 versus W for a family of transistors with the same channel length.

B.4 Determination of η

The expression for the drain current in the triode region of operation can be written as:

$$I_{DS} = \frac{\mu^o C_{ox} \frac{W}{L} \left[(V_{GS} - V_{FB} - 2\phi_F + kT/q + \alpha V_{DS}) V_{DS} - \frac{1}{2} V_{DS} - f + g) \right]}{1 + \frac{\theta}{V_{DS}} (V_{GS} - V_{FB} - 2\phi_F - \alpha V_{DS}) V_{DS} + f + \eta \frac{V_{DS}}{L}} \tag{B.9}$$

where the following symbol assignment is used for simplicity:

$$f = \gamma \frac{2}{3} \frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}} \left[(V_{DS} + 2\phi_F - kT/q + V_{SB})^{3/2} - (2\phi_F - kT/q + V_{SB})^{3/2} \right],$$

$$g = \gamma \frac{kT}{q} \frac{2}{3} \frac{\sqrt{2\epsilon_s q N_A}}{C_{ox}} \left[(V_{DS} + 2\phi_F - kT/q + V_{SB})^{1/2} - (2\phi_F - kT/q + V_{SB})^{1/2} \right].$$

The parameter η may be determined in the following manner. Define:

$$t_1 = t_2 + \eta, \tag{B.10}$$

where

$$t_1 = \frac{\mu^o W C_{ox} [(V_{GS} - V_{FB} - 2\phi_F + kT/q + \alpha V_{DS}) V_{DS} - \frac{1}{2} V_{DS} - f + g]}{I_{DS} V_{DS}} \tag{B.11}$$

$$t_2 = \frac{L}{V_{DS}} \left\{ 1 + \frac{\theta}{V_{DS}} \left[(V_{GS} - V_{FB} - 2\phi_F + \alpha V_{DS}) V_{DS} - \frac{1}{2} V_{DS} + f \right] \right\}. \tag{B.12}$$

A plot of t_1 versus t_2 for a chosen value of V_{GS} is a straight line with a unit slope and its abscissa intercept gives η.

The value of the channel length modulation parameters (η_1) has been found by "trial and error" approach by matching the measured and calculated drain current values in

the saturation region. No geometry dependence has been found to be necessary for this parameter.

In the conclusion of this section it should be pointed out that normally the process of parameter extraction is performed using an automated system that collects the necessary data and calculates model parameters subject to minimizing a given cost function [ChYS83].

Appendix C

ISSL Code Examples

C.1 Functional Module Specification

This section gives an example of ISSL code describing a functional module used in functional simulation, as discussed in Chapter 2. The ISSL code given in this section represents the formal functional description of the scaled FPA system for the case of addition (the code simulating the subtraction is similar but much larger and is not presented). This constitutes the Functional System Description module of Figure 2.8 for the module based validation of the FPA system. Before giving the actual ISSL description, some of the ISSL language commands are briefly explained.

The ports are declared in the STATE section of the code and their characteristics are defined in the PORTS part of the code. The INIT section allows for initial conditions of the internal state of a functional module to be defined. The section of code between the keywords FUNCTION and ENDFUNCTION describes the behaviour of the functional module. In this example the Boolean expressions for the addition of two one bit binary numbers are included. The values present at the input ports of the module are made available to that module by the READ statement. The new outputs generated by the module are made available at the output ports with the use of the WRITE statements. A DELAYEDWRITE statement is also possible with an additional argument defining the internal delay of this particular circuit.

```
DEFINITION FPA8Fun;
 STATE
  manA, manB, manR,        : ARRAY[0..7] of PortType;
  -- the mantissas are 5 bits but need 3 extra bits for G1, G2 and S
  expA, expB, expR         : ARRAY[0..3] of PortType;
```

```
 SA, SB, SR              : PortType;
 manAv, manBv, manRv,              -- v indicates an internal code
 expAv, expBv, expRv,              -- variable having the value of
 SAv, SBv, SRv,                    -- the given port
 shift                   : INTEGER -- the alignment shift amount
 shifted                 : REAL;   -- unnormalized mantissa
PORTS
 DeclarePort(SA,BIT,INP);      -- operand A sign bit
 DeclareBus(expA,INP);         -- operand A exponent
 DeclareBus(manA,INP);         -- operand A mantissa
 DeclarePort(SB,BIT,INP);      -- operand B sign bit
 DeclareBus(expB,INP);         -- operand B exponent
 DeclareBus(manB,INP);         -- operand B mantissa
INIT
FUNCTION
 manAv := READBUS(manA);       -- read the inputs
 expAv := READBUS(expA);
 manBv := READBUS(manB);
 expBv := READBUS(expB);

 -- Determine the alignment shift amount first
 IF((expBv-expAv) >= 0) THEN
  shift   := expBv - expAv;
  expRv   := expBv;
  shifted := manAv;          -- mantissa A will be unnormalized
 ELSE
  shift   := expAv - expBv;
  expRv   := expAv;
  shifted := manBv;          -- mantissa B will be unnormalized
 END;
 --Saturate the shift amount
 IF shift > 7 THEN
  shift := 7;
 END;
 -- Unnormalize one of the mantissas
 -- Simulate the ''sticky bit''
 IF(shift = 7) AND (shifted # 0.0) THEN
  shifted := 1.0;
```

```
ELSIF(shift = 6) AND (shifted # 0.0) THEN
 IF(shifted > 128.0) THEN
  shifted := 3;
 ELSE
  shifted := 2;
 END;
ELSIF(shift = 5) AND (shifted # 0.0) THEN
 IF(shifted > 192.0) THEN
  shifted := 7;
 ELSE
  shifted := 6;
 END;
ELSIF(shift = 4) AND (shifted # 0.0) THEN
 IF(shifted > 224) THEN
  shifted := 15;
 ELSE
  shifted := 14;
 END;
ELSIF(shift <= 3) THEN
 WHILE(shift > 0) DO
  shifted := shifted / 2.0;
  shift   := shift - 1;
 END;
END;
IF(expRv = expBv) THEN
 manAv := TRUNC(shifted);
ELSE
 manBv := TRUNC(shifted);
END;
-- Add the two mantissas
manRv := manAv + manBv;
-- Check for overflow
IF (manRv > 255) THEN
 manRv := manRv/2;
 expRv := expRv+1;
END;
-- Check if rounding required
IF FLOAT(manRv)/8.0-FLOAT(TRUNC(FLOAT(manRv)/8.0))) > 0.5 THEN
```

```
   manRv := manRv+4;
  END;
  IF FLOAT(manRv)/8.0-FLOAT(TRUNC(FLOAT(manRv)/8.0)) = 0.5 THEN
   IF TRUNC(FLOAT(manRv)/8.0) MOD 2 = 1 THEN    -- case of a tie
    manRv := manRv+4;                           -- round to even
   END;
  END;
  --Check for overflow
  IF (manRv > 255) THEN
   manRv := manRv/2;
   expRv := expRv+1;
  END;
  -- Output the result
  WRITEBUS(manR,manRv);
  WRITEBUS(expR,expRv);
 Pause(1);
ENDFUNCTION
END FPA8Fun;
```

C.2 Composition Module Description

The description of a composition module is given in terms of its elements and their interconnections. The section of code between the keyword TOPOLOGY and ENDTOPOLOGY gives description of instantiations of previously defined functional or topological (in this case functional) modules, information on how those modules are interconnected and how they are connected to the external ports of the composition module being defined.

The ISSL code given below represents the topological description of the scaled FPA system, according to the diagram of Figure 7.5.

```
DEFINITION FPA8;
 STATE
  expA, expB, expR          : ARRAY[0..3] OF PortType;
  fracA, fracB, fracR       : ARRAY[0..4] OF PortType;
  SA, SB, SR, OP, Inx, Z
  OVF, UNF, invA, invB      : PortType;
 PORTS
```

```
 DeclarePort(OP,BIT,INP);    -- add/subtract opcode
 DeclarePort(SA,BIT,INP);    -- operand A sign bit
 DeclarePort(SB,BIT,INP);    -- operand B sign bit
 DeclarePort(SR,BIT,INP);    -- sign bit of the result
 DeclareBus(expA,INP);       -- operand A exponent
 DeclareBus(expB,INP);       -- operand B exponent
 DeclareBus(expR,OUTP);      -- exponent of the result
 DeclareBus(fracA,INP);      -- operand A fraction
 DeclareBus(fracB,INP);      -- operand B fraction
 DeclareBus(fracR,OUTP);     -- fractional part of the result
 DeclarePort(OVF,OUTP);      -- result overflow flag
 DeclarePort(UNF,OUTP);      -- result underflow flag
 DeclarePort(Inx,OUTP);      -- inexact result flag
 DeclarePort(Z,OUTP);      -- zero flag
 DeclarePort(invA,OUTP);     -- invalid operand A
 DeclarePort(invB,OUTP);     -- invalid operand B
TOPOLOGY
 -- Instantiate modules
 MakeInstance(AlignUnit,AlignUnit);
 MakeInstance(Operate,Operate);
 MakeInstance(OutStage,OutStage);
 -- Connect signals to input and output ports
 Connect(AlignUnit/OP & OP);
 Connect(AlignUnit/SA & SA);
 Connect(AlignUnit/SB & SB);
 Connect(AlignUnit/invA & invA);
 Connect(AlignUnit/invB & invB);
 Connect(AlignUnit/expA & expA);
 Connect(AlignUnit/expB & expB);
 Connect(AlignUnit/fracA & fracA);
 Connect(AlignUnit/fracB & fracB);
 Connect(OutStage/SR & SR);
 Connect(OutStage/expR & expR);
 Connect(OutStage/fracR & fracR);
 Connect(OutStage/OVF & OVF);
 Connect(OutStage/UNF & UNF);
 Connect(OutStage/Inx & Inx);
 Connect(OutStage/Z & Z);
```

```
  -- Connect AlignUnit, Operate and OutStage
  Connect(AlignUnit/EOP & Operate/EOP);
  Connect(AlignUnit/X & Operate/X);
  Connect(AlignUnit/Y & Operate/Y);
  Connect(AlignUnit/expr & OutStage/expr);
  Connect(Operate/manr & OutStage/manr);
  Connect(Operate/MSBP & OutStage/MSBP);
  Connect(Operate/PrG1 & OutStage/PrG1);
  Connect(Operate/PrS & OutStage/PrS);
  Connect(Operate/Fnd & OutStage/Fnd);
 ENDTOPOLOGY
END FPA8;
```

Bibliography

[Adam85] L. Adams, "m-Step Preconditioned Conjugate Gradient Methods", *SIAM J. Sci. Stat. Comp.*, vol. 6, no. 2, pp. 452-463, (1985).

[AdOr82] L. Adams, J. Ortega, "A Multi Color SOR Method for Parallel Computation", *Proc. Int. Conf. Parallel Processing*, pp. 53-56, (1982).

[ADSP86] Analog Devices Technical Data, "DSP Microprocessor ADSP-2100", (1986).

[Aker81a] L. A. Akers, "Threshold Voltage of a Narrow Width MOSFET", *Electronics Letters*, vol. 17, no. 1, pp. 49-50, (1981).

[Aker81b] L. A. Akers, "An Analytical Expression for the Threshold Voltage of a Small Geometry MOSFET", *Solid-State Electronics*, vol. 24, pp. 621-627, (1981).

[AkBC81] L. A. Akers, M. M. E. Begulwa, F. Z. Custode, "A Model of a Narrow Width MOSFET Including Tapered Oxide and Doping Encroachment", *IEEE Trans. Electron Dev.*, vol. ED-28, no. 12, pp. 1490-1495, (1981).

[AKNS82] T. Adachi, H. Kitasawa, M. Nagatani, T. Sudo, "Hierarchical Top down Layout Design Method for VLSI Chip", *9th Design Automation Conference*, paper 44.2, (1982).

[AkPo80] L. A. Akers, W. M. Portnoy, "Numerical Analysis of the Steady State Behaviour of a MOSFET", *Int. J. Num. Meth. Eng.*, vol. 15, pp. 1221-1238, (1980).

[AlFe87] A. Aliphas, J. A. Feldman, "The Versatility of Digital Signal Processing Chips", *IEEE Spectrum*, pp. 40-45, June (1987).

[Alle83] J. Allen, "VLSI Architectures for Signal Processing", in "VLSI Architecture", B. Randell, P. C. Treleaven (Editors), Prentice-Hall, (1983).

[AnRe82] F. Anceau, R. A. Reis, "Complex Integrated Circuit Design Strategy", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 3, pp. 459-464, (1982).

[AnSm79] H. E. B. Anderson, D. H. Smith, "The Evaluation of MOSFET Model Parameters for Computer Simulation", *IREE Proceedings Aust.*, pp. 148-154, December (1979).

[ArAb66] E. Arnold, G. Abowitz, "Effect of Surface States on Electron Mobility in Silicon Surface Inversion Layers", *Applied Physics Letters*, vol. 9, no. 9, pp. 344-346, (1966).

[Asen82] A. M. Asenov, "Simple Model for Threshold Voltage of a Nonuniformly Doped Short Channel MOS Transistor", *Electronics Letters*, vol. 18, no. 11, pp. 481-482, (1982).

[Ahli63] A. C. Ahlin, "A Bivariate Generalization of Hermite's Interpolation Formula", *Math. Comp.*, vol. 18, pp. 264-273, (1963).

[Axel85] O. Axelsson, "A Survey of Preconditioned Iterative Methods for Linear Systems of Algebraic Equations", *BIT*, no. 25, pp. 166-187, (1985).

[AxGu79] O. Axelsson, I. Gustafson, "A Modified Upwind Scheme for Convective Transport Equations and the use of a Conjugate Gradient Method for the Solution of Nonsymmetric Systems of Equations", *J. Inst. Appl. Math.*, vol. 23, pp. 321-337, (1979).

[BaBe70] G. Baum, H. Beneking, "Drift Velocity Saturation in MOS Transistors", *IEEE Trans. Electron Dev.*, vol. ED-17, pp. 481-481, (1970).

[BaKo77] W. R. Bandy, D. P. Kokalis, "A Simple Approach for Accurately Modelling the Threshold Voltage of Short Channel MOSTS", *Solid-State Electronics*, vol. 20, pp. 675-679, (1977).

[BaRF83] R. Bank, D. J. Rose, W. Fichtner, "Numerical Methods for Semiconductor Device Simulation", *IEEE Trans. Electron Dev.*, vol. ED-30, no. 9, pp. 1031-1041, 1983.

[Barr72] M. B. Barron, "Low Level Currents in IGFETs", *Solid-State Electronics*, vol. 15, pp. 293-302, (1972).

[Barr84] H. G. Barrow, "Proving the Correctness of Digital Hardware Designs", *VLSI Design*, pp. 64-77, July (1984).

[BaVH83] N. Bandeira, K. Vaccaro, J. A. Howard, "A Two's Complement Array Multiplier Using True Values of the Operands", *IEEE Trans. Computers*, vol. C-32, no. 8, pp. 745-747, (1983).

[BaWo73] C. R. Baugh, B. A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm", *IEEE Trans. Computers* vol. C-22, no. 12, pp. 1045-1047, (1973).

[BBKK68] G. H. Barnes, R. M. Brown, M. Kato, D. J. Kuck, D. L. Slotnick, R. A. Stokes, "The ILLIAC IV Computer", *IEEE Trans. Computers*, vol. C-7, no. 8, pp. 746-757, (1968).

[Bell72] D. A. Bell, "Improved Formulation of Gummel's Algorithm for Solving the Two Dimensional Current-Flow Equations in Semiconductor Devices", *Electron. Letters*, vol. 8, no. 22, pp. 536-538, 1972.

[BlWi82] C. L. Wilson, J. L. Blue, "Two Dimensional Analysis of Semiconductor Devices using General Purpose Interactive PDE Software" *IEEE Trans. Electron Dev.*, vol. ED-30, no. 9, pp. 1056-1069, (1982).

[Boot51] A. D. Booth, "A Signed Binary Multiplication Technique", *Quart. J. Mech. Appl. Math.*, vol. 4, Part 2, (1951).

[Brau63] E. L. Braun, "Digital Computer Design", Academic Press, New York, 1963.

[Brew78] J. R. Brews, "A Charge Sheet Model of the MOSFET, *Solid-State Electronics*, vol. 21, pp. 345-355, (1978).

[BRGC86] G. Baccarani, M. Rudan, R. Guerrieri, P. Ciampolini, "Physical Models for Numerical Device Simulation", in "Process and Device Modeling", W. L. Engl, (Editor), North-Holland, (1986).

[BrKu81] R. P. Brent, H. T. Kung, "The Area-Time Complexity of Binary Multiplication", *J. ACM,* vol. 28, no. 3, pp. 521-534, (1981).

[BrKu83] M. A. Breuer, A. Kumar, "A Methodology for Custom VLSI Layout", *IEEE Trans. Circuits Systems*, vol. CAS-30, no. 6, pp. 358-363, (1983).

[Brod81] R. W. Brodersen, "Signal Processing Using MOS-VLSI Technology", in "VLSI Electronics: Microstructure Science", N. G. Einspruch (Editor), Academic Press, (19811).

[Brow53] W. L. Brown, "n-Type Conductivity on p-Type Germanium", *Physical Review*, vol. 91, no. 3, pp. 518-527, (1953).

[BrTF83] H. Brown, C. Tong, G. Foyster, "Palladio: An Exploratory Environment for Circuit design", *IEEE Computer*, pp. 41-55, December, (1983).

[BCGS81] E. M. Burtula, P. E. Cottrell, B. M. Grossman, K. A. Salsburg, "Finite Element Analysis of Semiconductor Devices: the FIELDAY Program", *IBM J. Res. Dev.,* vol. 25, pp. 218-230, (1981).

[BuCo80] E. M. Burtula, P. E. Cottrel, "Simulation of Semiconductor Transport using Coupled and Decoupled Solution Techniques", *Solid-State Electronics*, vol. 23, pp. 331-334, (1980).

[Buss82] D. D. Buss, "Technology and Design Challenges of MOS VLSI", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 3, pp. 442-448, (1982).

[CaTh67] D. M. Caughney, R. E. Thomas, "Carrier Mobilities in Silicon Empirically Related to Doping and Field, *IEEE. Proceedings,* vol. 55, pp. 2192-2193, (1967).

[CCMG80] J. G. J. Chern, P. Chang, R. F. Motta, N. Gordinho, "A New Method to Determine MOSFET Channel Length", *IEEE Electron Dev. Letters*, vol. EDL-1, no. 9, pp. 170-173, (1980).

[CFHR82] C. W. Cryder, P. M. Flanders, D. J. Hunt, S. F. Reddaway, "The Solutions of Linear Complementary Problems on an Array Processor", *J. Comp. Physics*, vol. 47, pp. 258-280, (1982).

[ChFi83] T. L. Chang, P. D. Fisher, "High-Speed Normalization and Rounding Circuits for Pipelined Floating-Point Processors", *IEEE Trans. Acoustics, Speech and Sig. Proc.,* vol. ASSP-31, no. 6, pp. 1403-1408, (1983).

[ChGK75] B. R. Chawla, H. K. Gummel, P. Kozak, "MOTIS-An MOS Timing Simulator", *IEEE Trans. Circuits Systems*, vol. CS-22, no. 12, pp. 513-517, (1975).

[ChLa84] Y. C. Cheng, P. T. Lai, "An Analytical Model for the Threshold Voltage of a Narrow Width MOSFET, *IEEE Trans. Electron Dev.,* vol. ED-31, no. 12, pp. 1814-1822, (1984).

[ChLT81] P. K. Chatterjee, J. E. Leiss, G. W. Taylor, "A Dynamic Average Model for the Body Effect in Ion Implanted Short Channel MOSFETs", *IEEE Trans. Electron Dev.,* vol. ED-28, no. 5, pp. 606-607, (1981).

[Choo72] S. C. Choo, "Theory of a Forward Biased Diffused Junction P-L-N Rectifier-Part I: Exact Numerical Solutions, *IEEE. Trans. Electron Dev.,* vol. ED-19, no. 8, pp. 954-957, (1972).

[ChSu73] Y. C. Cheng, E. A. Sullivan, "On the Role of Scattering by Surface Roughness in Silicon Inversion Layers", *Surface Science*, vol. 34, pp. 717-731, (1973).

[Clen62] C. W. Clenshaw, "Chebychev Series for Mathematical Functions", *Mathematical Tables*, vol. 5, Her Majesty's Stationery Office, London, (1962).

[CMMO74] C. Canali, G. Manjni, R, Minder, G. Ottaviani, "Electron Drift Velocity Measurements in Silicon at High Temperatures and Their Empirical Relations to Electric Field", *Electronics Letters*, vol. 10, no. 24, pp. 523-524, (1974).

[CMMO75] C. Canali, G. Manjni, R, Minder, G. Ottaviani, "Electron and Hole Drift Velocity Measurements in Silicon and Their Empirical Relations to Electric Field and Temperature", *IEEE. Trans. Electron Dev.,* vol. ED-22, no. 11, pp. 1045-1046, (1975).

[Cobb66] R. S. C. Cobbolt, "MOS Transistor as a 4-Terminal Device, *Electronics Letters*, vol. 2, no. 6, pp. 189-190, (1966).

[CoBN77] D. J. Coe, H. E. Brockman, K. H. Nicholas, "A Comparison of Simple and Numerical Two-Dimensional Models for the Threshold Voltage of Short Channel MOSTs", *Solid-State Electronics*, vol. 20, pp. 993-998, (1977).

[COCM86] K. M. Cham, S. Y. Oh, D. Chin, J. L. Moll, "Computer Aided Design and VLSI Device Development", Kluwer Academic Publisher, (1986).

[Coel84] D. R. Coelho, "Behavioural Simulation of LSI and VLSI Circuits", *VLSI Design*, February, (1984).

[Colg82] L. H. Colgan, "Iterative Methods for Solving Large Sparse Linear Systems", pp. 367-396, in "Numerical Solutions of Partial Differential Equations", J. Noye (Editor), North-Holland, (1982).

[CoMu80] R. W. Coen, R. S. Muller, "Velocity of Surface Carriers in Inversion Layers on Silicon", *Solid-State Electronics*, vol. 23, pp. 35-40, (1980).

[Coon81] J. T. Coonen, "An implementation Guide to a Proposed Standard for Floating Point Arithmetic", *Computer*, pp. 68-79, January (1980).

[CoWe50] E. Conwell, V. F. Weisskopf, "Theory of Impurity Scattering in Semiconductors", *Phys. Review*, vol. 77, no. 3, pp. 388-390, (1950).

[Craw67] R. H. Crawford, "MOSFET Theory in Circuit Design", McGraw-Hill, New York, (1967).

[CrZi79] B. L. Crowder, S. Zirinsky, "1μm MOSFET VLSI Technology: Part 7-Metal Silicide Interconnection Technology-A Future Perspective", *IEEE Trans. Electron Dev.*, vol. ED-26, no. 4, (1979).

[CSPD79] P. W. Cook, S. E. Schuster, J. T. Parrish, V. di Lonardo, D. R. Freedman, "1μm MOSFET VLSI Technology: Part 3-Logic Circuit Design Methodology and Applications", *IEEE Trans. Electron Dev.*, vol. ED-26, no. 4, pp. 333-345, (1979).

[Dang77a] L. M. Dang, "Drain Voltage Dependence of IGFET Turn On Voltage", *Solid-State Electronics*, vol. 20, pp. 825-830, (1977).

[Dang77b] L. M. Dang, "A One-Dimensional Theory on the Effects of Diffusion Current and Carrier Velocity Saturation on E-Type IGFET Current-Voltage Characteristics", *Solid-State Electronics*, vol. 20, pp. 781-788, (1977).

[Dang79] L. M. Dang, "A Simple Current Model for Short Channel IGFET and its Application to Circuit Simulation, *IEEE J. Solid-State Circuits*, vol. SC-14, no. 2, pp. 358-367, (1979).

[DaGw82] M. E. Daniel, C. W. Gwyn, "CAD Systems for IC Design", *IEEE Trans. Comp. Design Integrated Circuits*, vol. CAD-1, no. 1, pp. 2-12, (1982).

[DaRo55] G. C. Dacey, I. M. Ross, "The Field Effect Transistor", *Bell Sys. Tech. J.*, vol. 30, pp. 175-178, (1955).

[Das68] M. B. Das, "Physical Limitations of MOS Structures", *Solid-State Electronics*, vol. 12, pp. 305-336, (1968).

[DaSh83] R. Davis, H. Shrobe, "Representing Structure and Behaviour of Digital Hardware", *IEEE Computer*, pp. 75-8, October, (1983).

[DeMa68] A. De Mari, "An Accurate Numerical One-Dimensional Solution of the p-n Junction under Arbitrary Transient Conditions", *Solid-State Electronics* vol. 11, pp. 1021-1053, (1968).

[DeMa79] H. J. De Man, "Computer-aided Design for Integrated Circuits: Trying to Bridge the Gap", *IEEE J. Solid-State Circuits*, vol. SC-11, no. 3, pp. 613-621, (1979).

[DGYR74] R. H. Dennard, F. H. Gaensslen, Hwa-Nien Yu, V. L. Rideout, E. Bassous, A. R. LeBlanc, "Design of Ion-Implanted MOSFETs with Very Small Physical Dimensions", *IEEE J. Solid-State Circuits*, vol. SC-9, no. 5, pp. 256-267, (1974).

[Disn88] A. G. Dickinson, "Complexity Management and Modelling of VLSI Systems", Ph.D. Thesis, The University of Adelaide, Adelaide, (1988).

[DCML82] K. McDonough, E. Caudel, S. Magar, A. Leigh, "Microcomputer with 32-bit Arithmetic does High-Precision Number Crunching", *Electronics*, pp. 105-110, February 24, (1982).

[DoLe81] J. M. Dorkel, P. Leturcq, "Carrier Mobilities in Silicon Semi-empirically Related to Temperature, Doping and Injection Level, *Solid-State Electronics*, vol. 24, no. 9, pp. 821-825, (1981).

[Duff76] M. J. B. Duff, "CLIP4: A Large Scale Integrated Circuit Array Parallel Processor", *Proc. JCPR-3*, vol. 4, pp. 728-733, (1976).

[DuGR79] P. F. Dubois, A. Greenbaum, G. H. Rodrigue, "Approximating the Inverse of a Matrix for use in Iterative Algorithms on Vector Processors", *Computing*, vol. 22, pp. 257-268, (1979).

[EiES83] S. C. Eisenstat, H. C. Elman, M. H. Schultz, "Variational Iterative Methods for Nonsymmetric Systems of Linear Equations", *SIAM J. Numer. Anal.,* vol. 20, pp. 345-357, (1983).

[EnDM83] W. L. Engl, H. K. Dirks, B. Meinerzhagen, "Device Modelling", *IEEE Proceedings,* vol. 71, no. 1, pp. 10-33, (1983).

[Engl86] W. L. Engl (Editor), "Process and Device Modeling", North-Holland, (1986).

[EvTr73] A. J. Evensen, J. L. Troy, "Introduction to the Architecture of a 288 Element PEPE", *Proc. Segamore Conf. Parallel Processing*, pp. 162-169, (1973).

[Ferg64] J. Ferguson, "Multivariable Curve Iterpolation", *J. Association for Computing Machinery*, vol. 11, no. 2, pp. 221-228, (1964).

[Ferr76] D. K. Ferry, "Optical and Intervalley Scattering in Quantized Inversion Layers in Semiconductors", *Surface Science*, vol. 67, pp. 218-228, (1976).

[Ferr85] D. K. Ferry, "Interconnection Lengths and VLSI", *IEEE Circuits and Devices*, vol. 1, no. 4, pp. 39-42, (1985).

[Fike68] C. T. Fike, "Computer Evaluation of Mathematical Functions", Prentice-Hall, Inc. Englewood Cliffs, New Jersey, (1968).

[FiPo79] W. Fichtner, H. W. Pötzl, "MOS Modelling by Analytical Approximations. 1. Subthreshold Current and Threshold Voltage", *Int. J. Electronics*, vol. 46, pp. 33-55, (1979).

[FiRB83] W. Fichtner, D. J. Rose, R. Bank, "Semiconductor Device Simulation", *IEEE Trans. Electron Dev.,* vol. ED-30, no. 9, 1983.

[FKMD84] A. L. Fisher, H. T. Kung, L. M. Monier, Y. Dohi, "The Architecture of a Programmable Systolic Chip", *J. VLSI and Computer Systems*, vol. 1, no. 2, pp. 153-169, (1984).

[Flet75] R. Fletcher, "Conjugate Gradient Methods for Indefinite Systems", in Proc. Dundee Biennial Conf. on Numerical Analysis, G. A. Watson, (Editor), Springer Verlag, New York, (1975).

[Flor63] I. Flores, "The Logic of Computer Arithmetic", Prentice Hall, Englewood CLiffs, (1963).

[FNPP85] W. Fichtner, L. W. Nagel, B. R. Penumalli, W. P. Peterson, J. L. D'Arcy, "The Impact of Supercomputers on IC Technology Development and Design", *IEEE Proceedings,* vol. 72, no. 1, pp. 96-112, (9185).

[Forr72] A. R. Forrest, "On Coons and other Methods for the Representation of Curved Surfaces", *Computer Graphics and Image Processing*, vol. 1, pp. 341-359, (1972).

[FoWa60] G. E. Forsythe, W. R. Wasow, "Finite Difference Methods for Partial Differential Equations", New York: Wiley, (1960).

[FoOt84] G. C. Fox, S. W. Otto, "Algorithms for Concurrent Processors", *Physics Today*, pp. 50-59, May, (1984).

[FrGr69] D. Frohman-Bentchkowsky, A. S. Grove, "Conductance of MOS Transistors in Saturation", *IEEE Trans. Electron Dev.,* vol. ED-16, no. 1, pp. 108-113, (1969).

[Froh68] D. Frohman-Bentchkowsky, "On the Effect of Mobility on MOS Device Characteristics", *IEEE Proceedings*, vol. 56, pp. 217, (1968).

[Fung77] L. W. Fung, "A Massively Parallel Processing Computer", in "High Speed Computing and Algorithm Organization", D. J. Kuck, D. H. Lawrie, A. H. Sameh (Editors), Academic Press, New York, (1977).

[FuOk80] M. Fukuma, Y. Okuto, "Analysis of Short Channel MOSFETs with Field Dependent Carrier Drift mobility", *IEEE Trans. Electron Dev.,* vol. ED-27, no. 11, pp. 2109-2114, (1980).

[GaSo87] S. L. Garverick, C. Sodini, "A Simple Model for Scaled MOS Transistors that Includes Field Dependent Mobility", *IEEE J. Solid-State Circuits*, vol. SC-22, no. 1, pp. 111-114, (1987).

[GaOx84] D. Garde, J. Oxaal, "16 Bit Slice Family Creates Ultrafast Digital Signal Processors", *Electronic Design*, pp. 136-143, May 17, (1984).

[Gear71] C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, Englewood, NJ (1971).

[Gibb67] J. F. Gibbons, "Papers on Carrier Drift Velocities in Silicon at High Electric Field Strengths", *IEEE Trans. Electron Dev.,* vol. ED-14, no. 1, pp. 37-49, (1967).

[GiGi75] J. A. Gibson, R. W. Gibbard, "Synthesis and Comparison of Two's Complement Parallel Multipliers", *IEEE Trans. Computers*, pp. 1020-1027, (1975).

[GrDu80] J. A. Greenfield, R. W. Dutton, "Nonplanar VLSI Device Analysis Using the Solution of Poisson's Equation", *IEEE Trans. Electron Dev.,* vol. ED-27, no. 8, 1980.

[Grov67] A. S. Grove, "Physics and Technology of Semiconductor Devices, New York: Wiley 1967.

[GRWW77] F. H. Gaensslen, V. L. Rideout, E. J. Walker, J. J. Walker, "Very Small MOSFETs for Low-Temperature Operation", *IEEE Trans. Electron Dev.,* vol. ED-24, no. 3, pp. 218-229, (1977).

[Gumm64] H. K. Gummel, "A Self Consistent Iterative Scheme for One Dimensional Steady State Transistor Calculations", *IEEE Trans. Electron Dev.,* vol. ED-11, pp. 455-465, (1964).

[Gust78] I. Gustafsson, "A Class of First Order Factorization Methods", *BIT*, no. 18, pp. 142-156, 1978.

[HaCD82] H. I. Hanafi, L. H. Camnitz, A. J. Dally, "An Accurate and Simple MOSFET Model for Computer Aided Design", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 5, pp. 882-891, (1982).

[Hage81] L. A. Hageman, "Applied Iterative Methods", Academic Press, Inc. (London) Ltd. (1981).

[HaLY80] L. A. Hageman, F. T. Luk, D. M. Young, "On the Equivalence of Certain Iterative Acceleration Methods", *SIAM J. Numer. Anal.,* vol. 17, no. 6, 1980.

[HAMM87] S. Hanamura, M. Akoi, T. Masuhara, O. Minato, Y. Sakai, T. Hayashida, "Low Temperature CMOS 8 × 8 bit Multiplier with Sub 10ns Speeds", *IEEE Trans. Electron Dev.,* vol. ED-34, no. 1, pp. 94-99, (1987).

[HaRP82] C. D. Hartgring, B. A. Rosario, J. M. Pickett, "High Speed Low Power Silicon MESFET Parallel Multipliers", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 1, pp. 69-73, 1982.

[HaSa81] G. D. Hatchel, A. L. Sangiovanni-Vioncentelli, "A Survey of Third Generation Simulation Techniques", *IEEE Proceedings*, vol. 69, no. 10, pp. 1264-280, (1198).

[Hask83] M. R. Haskard, "A Simple Method for Determining SPICE MOS Transistor Model Static Parameters", *J. IREE Aust.,* vol. 3, no. 3, pp. 232-233, (1983).

[HaWi70] A. Habibi, P. A. Wintz, "Fast Multipliers", *IEEE Trans. Computers,* pp. 153-157, (1970).

[HeSt52] M. R. Hestenes, E. L. Stiefel, "Methods of Conjugate Gradients for Solving Linear Systems", *Nat. Bur. Std. J. Res.,* vol. 49, pp. 409-436, (1952).

[Hils74] C. Hilsum, "Simple Empirical Relationship Between Mobility and Carrier Concentration", *Electronics Letters*, vol. 10, no. 31, pp. 259-260, (1974).

[HKBB85] W.P.Hays, R. N. Kershaw, L. E. Bays, J. R. Boddie *et. al.,* "A 32-bit VLSI Digital Signal Processor", *IEEE J. Solid-State Circuits*, vol. SC-20, no. , pp. 998-1004, (1985).

[HKMT83] Y. Hagiwara, Y. Kita, T. Miyamoto, Y. Toba, H. Hara, T. Akazawa, "A Single Chip Digital Signal Processor and Its Application to Real Time Speech Analysis", *IEEE Trans. Acous. Speech, Sig. Proc.,* vol. ASSP-31, no. 1, pp. 339-345, (1983).

[HKSH83] T. Hoshino, T. Kawai, T. Shirakawa, J. Higashino, A. Yamaoka, H. Ho, T. Sato, K. Sawada, "PACS: A parallel Microprocessor Array for Scientific Computations", *ACM Trans. Comp. Sys.,* vol. 1, no. 3, pp. 129-221. (1983).

[HNNT87] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, N. Takai, "High Speed Multiplier Using a Redundant Binary Tree", *J. Solid-State Circuits*, vol. SC-22, no. 1, pp. 28-33, (1987).

[HoMe72] B. Hoeneisen, C. A. Mead, "Fundamental Limitations in Microelectronics-1. MOS Technology", *Solid-State Electronics*, vol. 15, pp. 819-829, (1972).

[HuCh82] A. Husain, S. G. Chamberlain, "Three Dimensional Simulation of VLSI MOSFETs: The Three Dimensional Simulation Program WATMOS", *IEEE Trans. Electron Dev.,* vol. ED-29, no. 4, pp. 631-638, (1982).

[Hurs85] S. L. Hurst, "Custom-Specific Integrated Circuits", Marcel Dekker, Inc., (1985).

[Hwan79] K. Hwang, "Computer Arithmetic: Principles, Architecture and Design", John Wiley and Sons, (1979).

[InMo64] H. K. Inhantola, J. L Moll, "Design Theory of a Surface Field Effect Transistor", *Solid-State Electronics*, vol. 7, pp. 423-430, (1964).

[IEEE85] "IEEE Standard for Floating-Point Arithmetic", Institute of Electrical and Electronics Engineers, Inc., (1985).

[IMGB82] A. C. Ipri, L. B. Medwin, N. Goldsmith, F. W. Brehm, "Two Dimensional Dynamic Analysis of Short Channel Thin Film Transistors Using Minicomputer", *IEEE Trans. Electron Dev.,* vol. ED-29, no. 4, pp. 618-625, (1982).

[Irvi62] J. C. Irvin, "Resistivity of Bulk Silicon and of Diffused Layers in Silicon", *Bell System Technical Journal,* vol. 41, no. 2, pp. 387-410, (1962).

[ISD86] Integrated Silicon Design Pty. Ltd., "Phase 1 VLSI Design Suite", User Manual, Version 1.00, (1986).

[ISD87] Integrated Silicon Design Pty. Ltd., "Phase 2 VLSI Design Suite", User Manual, Version 1.00, (1987).

[ISD88] Integrated Silicon Design Pty. Ltd., "Universal Systems Analyser", User Manual, Version 1.00, (1988).

[Jant82] O. Jantsch, "A Geometrical Model of the Threshold Voltage of Short and Narrow Channel MOSFETs", *Solid-State Electronics*, vol. 25, no. 1, pp. 59-61, (1982).

[JCOQ77] C. Jacobini, C. Canali, G. Ottaviani, A. A. Quartani, "A Review of Some Charge Transport Properties of Silicon, *Solid-State Electronics*, vol. 20, pp. 77-89, (1977).

[Jepp75] K. O. Jeppson, "Influence of the Channel Width on the Threshold Voltage Modulation in MOSFETs", *Electronics Letters*, vol. 11, no. 14, pp. 297-298, (1975).

[JiSa83a] C. R. Ji, C. T. Sah, "Two Dimensional Numerical Analysis of the Narrow Gate Effect in MOSFET", *IEEE Trans. Electron Dev.,* vol. ED-30, no. 6, pp. 635-646, (1983).

[JiSa83b] C. R. Ji, C. T. Sah, "Analysis of the Narrow Gate Effect in Submicrometer MOSFETs", *IEEE Trans. Electron Dev.,* vol. ED-30, no. 12, pp. 1672-1677, (1983).

[JLLR73] F. S. Jenkins, E. R. Lane, W. W. Lattin, "MOS-Device Modeling for Computer Implementation", *IEEE Trans. Circuit Theory*, vol. CT-20, no. 6, pp. 655-657, (1973).

[KeMu73] D. P. Kennedy, P. C. Murley, "Steady State Mathematical Theory of the IGFET", *IBM J. Res. Dev.,* vol. 17, pp. 2-112, (1973).

[KeOB69] D. P. Kennedy, R. R. O'Brien, "Two Dimensional Mathematical Analysis of a Planar Type Junction Field Effect Transistor", *IBM J. Res. Dev.,* vol. 31, pp. 662-674, (1969).

[Kers77] D. S. Kershaw, "The Incomplete Choleski-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations", *J. Comp. Physics*, no. 26, pp. 43-65, 1978.

[Keye81] R. W. Keyes, "Limitations of Small Devices and Large Systems" in "VLSI Electronics: Microstructure Science" N. G. Einspruch (Editor), Academic Press, (198).

[Klas77a] F. M. Klassen, "Review of Physical Models for MOS Transistors", in "Process and Device Modeling for Integrated Circuit Design", F. van de Wiele, W. L. Engl, P. G. Jespers (Editors), Nordhoff-Leyden, (1977).

[Klas77b] F. M. Klassen, "Characterization and Measurements of MOS Devices", in "Process and Device Modeling for Integrated Circuit Design", F. van de Wiele, W. L. Engl, P. G. Jespers (Editors), Nordhoff-Leyden, (1977).

[Klas78] F. M. Klassen, "Design and Performance of Micron Size Devices", *Solid-State Electronics*, vol. 21, pp. 565-571, (1978).

[KlGr80] F. M. Klassen, W. C. J. de Groot, "Modelling Scaled Down MOS Transistors", *Solid-State Electronics*, vol. 23, pp. 237-242, (1980).

[Koll88] P. Kollaritsch, private communications, (1988).

[KrAc76] K. E. Kroell, K. Ackermann, "Threshold Voltage of Narrow Channel Field Effect Transistor", *Solid-State Electronics*, vol. 19, pp. 77-81, (1976).

[Kuck81] D. J. Kuck, "The Structure of Computers and Computations", John Wiley and Sons, (1981).

[KYYS82] R. Kasai, K. Yokoyama, A. Yoshii, T. Sudo, "Threshold Voltage Analysis of Short and Narrow Channel MOSFETs by Three Dimensional Computer Simulation", *IEEE Trans. Electron Dev.,* vol. ED-29, no. 5, pp. 870-875, (1982).

[Lanc60] G. N. Lance, "Numerical Methods for High Speed Computers", London: Iliffe and Sons Ltd., (1960).

[Lawr75] D. H. Lawrie, "Access and Alignment of Data in an Array Processor", *IEEE Trans. Computers* vol. C-24, pp. 1145-1155, (1975).

[LBBR81] W. W. Lattin, J. A. Bayliss, D. L. Budde, J. R. Rattner, W. S. Richardson, "A Methodology for VLSI Chip Design", *Lambda*, Second Quarter, pp. 34-39, (1981).

[LeCT84] F. Lee, P. Chiu, F. Toth, "16 by 16 Bit Multipliers Fabricated in CMOS Rival the Speed of Bipolars", *Electronic Design*, pp. 311-315, June 14, (1984).

[Lee73] H. S. Lee, "An Analysis of the Threshold Voltage for Short Channel IGFETs", *Solid-State Electronics*, vol. 16, pp. 1407-1417, (1973).

[LeGD81] J. P. Leburton, H. Gesch, G. Dorda, "Analytical Approach of Hot Electron Transport in Small Size MOSFETs", *Solid-State Electronics*, vol. 24, pp. 763-771, (1981).

[LeGS65] O. Leistiko, A. S. Grove, C. T. Sah, "Electron and Hole Mobilities in Inversion Layers on Thermally Oxidized Silicon Surfaces", *IEEE Trans. Electron Dev.,* vol. ED-12, pp. 248-254, (1965).

[LeGS87] J. Y. Lee, H. L. Garvin, C. W. Slayman, "A High Speed High Density Silicon 8 × 8 bit Parallel Multiplier", *IEEE J. Solid-State Circuits*, vol. SC-22, no. 1, pp. 35-40, 1987.

[Lipp83] H. M. Lipp, "Methodical Aspects of Logic Synthesis", *IEEE Proceedings,* vol. 71, no. 4, pp. 88-97, (1983).

[LiTh77] S. S. Li, W. R. Thurber, "The Dopand Density and Temperature Dependence of Electron Mobility and Resistivity in n-Type Silicon, *Solid-State Electronics*, vol. 20, pp. 609-616, (1977).

[LoLa80] A. D. Lopez, H. S. Law, "A Dense Gate Matrix Layout Method for MOS VLSI", *IEEE J. Solid-State Circuits*, vol. SC-15, no. 4, pp. 736-740, (1980).

[LyCY65] L. A. Lyusternik, O. A. Chervonenkis, A. R. Yanpolskii, "Handbook for Computing Elementary Functions", Pergamon Press, (1965).

[MaKi71] J. C. Majithia, R. Kitai, "An Iterative Array for Multiplication of signed Binary Numbers", *IEEE Trans. Computers*, vol. C-20, no. 4, pp. 214-216, (1971).

[MaNK79] H. Masuda, M. Nakai, M. Kubo, "Characteristics and Limitations of Scaled-Down MOSFETs due to Two-Dimensional Field Effects", *IEEE Trans. Electron Dev.,* vol. ED-26, pp. 980-986, (1979).

[Mans82] Y. El-Mansy, "MOS Device and Technology Constraints in VLSI", *IEEE Trans. Electron Dev.,* vol. ED-29, no. 4, pp. 567-573, (1982).

[Mant77] T. A. Manteuffel, "The Thebychev Iteration for Nonsymmetric Linear Systems", *Numer. Math.,* vol. 28, pp. 307-327, (1977).

[Mant78] T. A. Manteuffel, "Adaptive Procedure for Estimating Parameters for the Nonsymmetric Tchebychev Iteration", *Numer. Math.,* vol. 31, pp. 183-208, (1978).

[Mant80] T. A. Manteuffel, "An Incomplete Factorization Technique for Positive Definite Linear Systems", *Math. Comp.,* vol. 34, no. 150, pp. 473-497, 1980.

[MaRV72] H. Martinot, P. Rossel, G. Vassilieff, "Mobility Parameters and MOST Properties", *Electronics Letters*, vol. 8, no. 24, pp. 504-504, (1972).

[MaSS83] G. Masetti, M. Severi, S. Solmi, "Modelling of Carrier Mobility Against Carrier Concentration in Arsenic-, Phosphorous- and Boron Doped Silicon", *IEEE. Trans. Electron Dev.,* vol. ED-30, no. 7 pp. 764-769, (1983).

[MeBC72] G. Merckel, J. Borel, N. Z. Cupcea, "An Accurate Large Signal MOS Transistor Model for Use in Computer Aided Design", *IEEE Trans. Electron Dev.,* vol. ED-19, no. 5, pp. 681-690, (1972).

[MeCo80] C. Mead, L. Conway, "Introduction to VLSI Systems", Addison-Wesley, (1980).

[Meil81] W. C. Meilander, "History of Parallel Processing at Goodyear Aerospace", *Proc. Int. Conf. Parallel Processing*, pp. 5-15, (1981).

[Merc80] G. Merckel, "A Simple Model of the Threshold Voltage of Short and Narrow Channel MOSFETs", *Solid-State Electronics*, vol. 23, pp. 1207-1213, (1980).

[MeVM81] R. P. Mertens, R. J. Van Overstraeten, H. J. De Man, "Heavy Doping Effects in Silicon", in "Advances in electronics and Electron Physics", L. Marton, C. Marton (Editors), Academic Press, New York, (1981).

[MevV77] J. A. Meijerink, H. A. van der Vorst, "An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix", *Math. Comp.,* vol. 31, no. 137, pp. 148-162, 1977.

[MevV81] J. A. Meijerink, H. A. van der Vorst, "Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as They Occur in Practical Problems", *J. Comp. Physics*, no. 44, pp. 134-155, 1981.

[Mock73] M. S. Mock, "A Two Dimensional Mathematical Model of the Insulated Gate Field Effect Transistor", *Solid-State Electronics*, vol. 6, pp. 601-609, (1973).

[Mock83] M. S. Mock, "Analysis of Mathematical Models for Semiconductor Devices", Boole Press, Dublin (1983).

[MoKS82] F. H. de la Moneda, H. N. Kotecha, M. Shatzkes, "Measurement of MOSFET Constants", *IEEE Electron Dev. Letters*, vol. EDL-3, no. 1, pp. 10-12, (1982).

[Mone73] F. H. de la Moneda, "Threshold Voltage from Numerical Solution of the Two-Dimensional MOS Transistors", *IEEE Trans. Circuit Theory*, vol. CT-20, no. 6, pp. 666-673, (1973).

[Moto86] Motorola Semiconductor Technical Data, "56-bit General Purpose Digital Signal Processor", no. BR282, (1986).

[Murr84] N. W. Murray, "Data Memory Control Structure for the TFB", Final Year Undergraduate Project Report, The University of Adelaide, (1984).

[NaBo82] A. A. Naem, A. R. Boothroyd, "Compensation Tendency of Short Channel and Narrow Channel Effects in Small Geometry IGFETs, *Electronics Letters*, vol. 18, no. 3, pp. 135-136, (1982).

[Nash85] S. G. Nash, "Preconditioning of Truncated Newton Methods", *SIAM J. Sci. Stat. Comp.,* vol. 6, no. 3, pp. 599-615, (1985).

[NoBL73] P. Norton, T. Braggins, H. Levinstein, "Impurity and Lattice Scattering Parameters as Determined from Hall and Mobility Analysis in n-Type Silicon", *Phys. Review B*, Vol. 8, no. 12, pp. 5632-5653, (1973).

[NoGi67] C. B. Norris, J. F. Gibbons, "Measurement of High Field Carrier Drift Velocities in Silicon by Time of Flight Technique", *IEEE Trans. Electron Dev.,* vol. ED-14, no. 1, pp. 38-43, (1967).

[Noye82] J. Noye, (Editor), "Numerical Solutions of Partial Differential Equations", North-Holland, (1982).

[OhKD79] N. Ohwada, T. Kimura, M. Doken, "LSI's for Digital Signal Processing", *IEEE Trans. Electron Dev.,* vol. ED-26, no. 4, pp. 292-298, (1979).

[OHYM82] K. Ochii, K, Hashimoto, H. Yasuda, M. Masuda, T. Kondo, H. Npzawa, S. Kohyama, "An Ultralow Power $256K \times 8$ bit Full CMOS RAM with a Six Transistor Cell", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 5, pp. 798-803, (1982).

[OhWD80] S. Y. Oh, D. E. Ward, R. W. Dutton, "Transient Analysis of MOS Transistors", *IEEE Trans. Electron Dev.,* vol. ED-27, no. 9, pp. 1571-1578, (1980).

[OmOh79] Y. Omura, K. Ohwada, "Threshold Voltage Theory for a Short Channel MOSFET using a Surface Potential Distribution Model", *Solid-State Electronics*, vol. 22, pp. 1045-1051, (1979).

[ONNI80] H. Oka, K. Nishiuchi, T. Nakamura, H. Ishikawa, "Computer Analysis of a Short Channel BC MOSFET, *IEEE Trans. Electron Dev.,* vol. ED-27, no. 8, pp. 1514-1520, (1980).

[Orbi87] Orbit Semiconductor, Inc., "2-micrometer Single Poly Double Metal CMOS Process Design Rules", Technical Data, (1987).

[Park87] D. Parkinson, "Organisational Aspects of Using Parallel Computers", *Parallel Computing*, vol. 5, pp. 75-83, (1987).

[PaSa66a] H. C. Pao, C. T. Sah, "Effects of Diffusion Current on Characteristics of Metal Oxide (Insulator) Semiconductor Transistors, *Solid-State Electronics*, vol. 9, pp. 927-937, (1966).

[PaSa66b] H. C. Pao, C. T. Sah, "The Effects of Fixed Bulk Charge on the Characteristics of Metal Oxide Semiconductor Transistors, *IEEE Trans. Electron Dev.,* vol. ED-13, no. 4, pp. 393-409, (1966).

[PaSe80] D. A. Patterson, C. H. Séquin, "Design Considerations for Single Chip Computers of the Future", *IEEE Trans. Computers*, vol. C-29, no. 2, pp. 108-115, (1980).

[Penu86] B. R. Penumali, "Physical Models and Numerical Methods for VLSI Process Simulation", in "Advance in CAD for VLSI", W. L. Engl (Editor), North-Holland, (1986).

[Peza71] S. D. Pezaris, "A 40-ns 17-Bit by 17-Bit Array Multiplier", *IEEE Trans. Computers*, vol. C-20, no. 4, pp. 442-447, (1971).

[PfSM85] J. R. Pfister, J. D. Shott, J. D. Meindl, "Performance Limits of CMOS ULSI", *IEEE Trans. Electron Dev.,* vol. ED-32, no. 2, pp. 333-343, (1985).

[Pier83] R. F. Pierret, "Field Effect Devices", Addison-Wesley, (1983).

[PiSa68] R. F. Pierret, C. T. Sah, "An MOS-Oriented Investigation of Effective Mobility Theory", *Solid-Stage Electronics*, vol. 11, pp. 279-290, (1968).

[PiSh83] R. F. Pierret, J. A. Shields, "Simplified Long Channel MOSFET Theory", *Solid-State Electronics*, vol. 26, pp. 143-147, (1983).

[PlSt77] J. C. Plunkett, J. L. Stone, "A Computer Algorithm for Accurate Profile Analysis Using Anodizing and Stripping of Silicon, *Solid-State Electronics*, vol. 20, pp. 447-453, (1977).

[Popa72] A. Popa, "An Injection Level Dependent Theory of the MOS Transistor in Saturation", *IEEE Trans. Electron Dev.,* vol. ED-119, no. 6, pp. 774-781, (1972).

[PYJB73] H. C. Poon, L. D. Yau, R. L. Jonston, D. Beecham, "DC Model for Short Channel IGFETs", *Technical Digest*, International Electron Devices Meeting, Washington, D.C., pp. 156-159, (1973).

[RaGo75] L. R. Rabiner, B. Gold, ,"Theory and Application of Digital Signal Processing", Prentice-Hall, Englewood Cliffs. N.J., 1975.

[RaMa81] C. V. Ramamoorthy, Y. W. Ma, "Impact of VLSI on Computer Architectures", in "VLSI Electronics: Microstructure Science", N. G. Einspruch (Editor), Academic Press, (1981).

[RaMe82] K. N. Ratnakumar, J. D. Meindl, "Short Channel MOST Threshold Voltage Model", *J. Solid-State Circuits*, vol. SC-17, no. 5, pp. 937-947, (1982).

[RaSa84] P. Ratman, C. A. T. Salama, "A New Approach to the Modelling of Nonuniformly Doped Short Channel MOSFETs", *IEEE Trans. Electron Dev.,* vol. ED-31, no. 9, pp. 1289-1298, (1984).

[ReKM81] P. Reusens, W. H. Ku, Y. H. Mao, "Fixed Point High Speed Parallel Multipliers in VLSI", in "VLSI Systems and Computations", H. T. Kung, B. Sproull, G. Steele (Editors) Springer-Verlag, (1981).

[ReRe79] R. Resta, R. Resca, "Ionized Impurity Scattering in Semiconductors", *Phys. Review B*, vol. 20, no. 8, pp. 3254-3257, (1979).

[ReSa65] V. G. Reddi, C. T. Sah, "Source to Drain Resistance Beyond Pinch-Off in Metal Oxide Semiconductor Transistors (MOST), *IEEE Trans. Electron Dev.,* vol. ED-14, no. 3, pp. 139-141, (1965).

[Risc83] L. Risch, "Electron Mobility in Short Channel MOSFETs with Series Resistances", *IEEE Trans. Electron Dev.,* vol. ED-30, no. 8, pp. 959-961, (1983).

[RoAC82] D. J. Roulston, N. D. Arora, S. G. Chamberlain, "Modeling and Measurement of Minority Carrier Lifetime versus Doping in Diffused Layers of np Silicon Diodes", *IEEE Trans. Electron Dev.,*vol. ED-29, no. 2, pp. 284-289, (1982).

[Rock86] J. E. Rockliff, "The Implementation of Testability Strategies in a VLSI Circuit", M. Eng. Sc. Thesis, The University of Adelaide, Adelaide, Australia, (1986).

[RoMV76] P. Rossel, H. Martinot, G. Vassilieff, "Accurate Two Sections Model for MOS Transistor in Saturation", *Solid-State Electronics*, vol. 19, pp. 5-56, (1976).

[Roos50] W. V. Van Roosbroeck, "Theory of Flow of Electrons and Holes in Germanium and Other Semiconductors", *Bell. Sys. Tech. J.,* vol. 29, pp. 560-607, 1950.

[RoRN67] V. Rodriguez, H. Ruegg, M. A. Nickolet, "Measurement of the Drift Velocity of Holes in Silicon at High Field Strengths", *IEEE Trans. Electron Dev.,* vol. ED-14, no. 1, pp. 44-46, (1967).

[Rubi75] L. P. Rubinfield, "A Proof of the Modified Booth's Algorithm for Multiplication", *IEEE Trans. Computers*, vol. C-24, pp. 1014-1015, (1975).

[Ryde53] E. J. Ryder, "Mobility of Holes and Electrons in High Electric Fields", *Physical Review*, vol. 90, no. 5, pp. 766-769, (1953).

[RySh51] E. J. Ryder, W. Shockley", *Physical Review*, vol. 81, pp. 139-145, (1951).

[Saad84] Y. Saad, "Practical use of some Krylov Subspace Methods for Solving Indefinite and Nonsymmetric Linear Systems", *SIAM J. Sci. StatComput.,* vol. 5, n. , pp. 203-227, (1984).

[SAIW85] H. Shinohara, K. Anami, K. Ichinose, T. Wada, Y. Kohno, Y. Akasaka, S. Kayano, "A 45 ns CMOS Static RAM with a Tri Level Word Line", *IEEE J. Solid-State Circuits*, vol. SC-20, no. 5, pp. 929-934, (1985).

[SaNT72] C. T. Sah, T. H. Ning, L. L. Tschopp, "The Scattering of Electrons by Surface Oxide Charges and by Lattice Vibrations at the Silicon Silicon Dioxide Interface", *Surface Science*, vol. 52, pp. 561-575, (1972).

[Savi84] E. Savio, "Input and Output Modules for TFB", Final Year Undergraduate Project Report, The University of Adelaide, (1984).

[ScGu69] D. L. Scharfetter, H. K. Gummel, "Large Signal Analysis of a Silicon Read Diode Oscillator", *IEEE. Trans. Electron Dev.,* vol. ED-16, no. 1, pp. 64-77, (1969).

[Scho75] H. Schomberg, "A Special Purpose Computer for Partial Differential Equations", *Advances in Computer Methods for Partial Differential Equations*, pp. 187-191, R. Vichnevetsky (Editor), (1975).

[ScSP82] A. Schütz, S. Selberherr, H. W. Pötzl, "A Two Dimensional Model of the Avalanche Effect in MOS Transistors", *Solid State Electron.,* vol. 25, pp. 177-183, 1982.

[Seag86] M. K. Seager, "Parallelizing Conjugate Gradient for the CRAY X-MP", *Parallel Computing*, vol. 3, pp. 35-47, (1986).

[Selb84] S. Selberherr, "Analysis and Simulation of Semiconductor Devices", Springer-Verlag, Wien, New York, 1984.

[Séqu83] C. H. Séquin, "Managing VLSI Complexity: An Outlook", *IEEE Proceedings*, vol. 71, no. 1, pp. 149-166, (1983).

[SeRi84] S. Selberherr, C. A. Ringhofer, "Implications of Analytical Investigations About the Semiconductor Equations on Device Modeling Programs", *IEEE Trans. Computer Aided Design*, vol. CAD-3, no. 1, pp. 52-63, 1984.

[SeSP80] S. Selberherr, A. Schütz, H. W. Pötzl, "MINIMOS-A Two-Dimensional MOS Transistor Analyzer", *Trans. Electron Dev.,* vol. ED-27, no. 8, pp. 1540-1550, 1980.

[ShDa85] N. Shigyo, R. Dang, "Analysis of an Anomalous Subthreshold Current in a Fully Recessed Oxide MOSFET using a Three-Dimensional Device Simulator", *IEEE Trans. Electron Dev.,* vol. ED-32, no. 2, pp. 441-445, (1985).

[ShDa86] N. Shigyo, R. Dang, "Three-Dimensional Device Simulation Using a Mixed Process/Device Simulator", pp. 301-327, in "Process and Device Modeling", W. L. Engl (Editor), Elsevier Science Publishers B. V. (North-Holland), (1986).

[Shic83] H. Shichijo, "A Re-Examination of Practical Performance Limits of Scaled n-Channel and p-Channel MOS Devices for VLSI, *Solid-State Electronics*, pp. 969-986, (1983).

[Shim86] T. Shima, "Device and Circuit Simulator Integration Technique", in "Process and Device Modeling", W. L. Engl, (Editor), North-Holland, (1986).

[Shoc51] W. Shockley, "Hot Electrons in Germanium and Ohm's Law", *Bell System Technical Journal* vol. 30, pp. 990-1034, (1951).

[ShYD83] T. Shima, H. Yamada, R. L. M. Dang, "Table Look-up MOSFET Modeling System Using a 2-D Device Simulator and Monotonic Piecewise Cubic Interpolation", *IEEE Trans. Comp. Design*, vol. CAD-2, no. 2, pp. 121-126, (1983).

[SiFP84] A. L. Silburt, R. C. Foss, W. F. Petrie, "An Efficient MOS Transistor Model for Computer Aided Design", *IEEE Trans. Computer Aided Design*, vol. CAD-3, no. 1, pp. 104-111, (1984).

[SiHa87] R. Simar, M. Hames, "CMOS DSP Chip Packs Punch of a Supercomputer", *Electronic Design*, pp. 103-10, March 19, (1987).

[SiOC86] Y. W. Sing, J. Oxaal, G. Chiu, "32-Bit Data Path Chip Heralds Next Generation of Floating Point Tasks", *Electronic Design*, pp. 187-193, May 29, (1986).

[Slot73] J. W. Slotboom, "Computer-Aided Two-Dimensional Analysis of Bipolar Transistors", *IEEE Trans. Electro Dev.,* vol. ED-20, no. 8, (1973).

[Smit78] R. A. Smith, "Semiconductors, Cambridge University Press, (1978).

[Snyd84] L. Snyder, "Supercomputers and VLSI: The Effect of Large-Scale Integration on Computer Architecture", in "Supercomputers and VLSI", pp. 2-33, (1984).

[SSMY82] T. Shima, T. Sugawara, S. Moriyama, H. Yamada, "Three Dimensional Table Look-up MOSFET Model for Precise Circuit Simulation", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 3, pp. 449-453, (1982).

[Sten83] L. J. Stenberg, "Modified Strong Inversion Potential for Accurate Modelling of Long Channel MOS Transistors", *IEEE Proceedings,* vol. 130, no. 2, pp. 57-60, (1983).

[Ston71] H. S. Stone, "Parallel Processing with the Perfect Shuffle", *IEEE Trans. Computers* vol. C-20, pp. 153-161, (1971).

[Ston75] H. S. Stone, "Parallel Tridiagonal Equation Solvers", *ACM Trans. on Math. Software*, vol. 1, no. 4, pp. 298-307, (1975).

[SuJo80] P. I. Suciu, R. L. Johnston, "Experimental Derivation of the Source and Drain Resistance in MOS Transistors", *IEEE Trans. Electron Dev.,* vol. ED-27, no. 9, pp. 1846-1849, (1980).

[SuPl80] S. C. Sun, J. D. Plummer, "Electron Mobility in Inversion and Accumulation Layers on Thermally Oxidized Silicon Surfaces *IEEE Trans. Electron Dev.,* vol. ED-27, no. 8, pp. 1497-1508, (1980).

[Suth80] A. D. Sutherland, "On the Use of Overrelaxation in Conjunction with Gummel's Algorithm to Speed the Convergence in a Two Dimensional Computer Model for MOSFET's", *IEEE Trans. Electron. Dev.,* vol. ED-27, pp. 1297-1289, (1980).

[Sze81] S. M. Sze, "Physics of Semiconductor Devices, New York: Wiley (1981).

[TaJA85] E. Takida, G. A. C. Jones, H. Ahmed, "Constraints on the Application of 0.5-micron MOSFETs to ULSI Systems", *IEEE Trans. Electron Dev.,* vol. ED-32, no. 2, pp. 322-372, (1985).

[Tayl78] G. W. Taylor, "Subthreshold Conduction in MOSFETs, *IEEE Trans. Electron Dev.,* vol. ED-25, no. 3, pp. 337-350, (1978).

[Tayl79] G. W. Taylor, "The Effects of Two Dimensional Charge Sharing on the Above Threshold Characteristics of Short Channel IGFETs, *Solid-State Electronics*, vol. 22, pp. 701-717, (1979).

[TaYY85] N. Takagi, H. Yasuura, S. Yajima, "High Speed VLSI Multiplication Algorithm with a Redundant Binary Tree", *IEEE Trans. Computers*, vol. C-34, no. 9, pp. 789-796, (1985).

[Thor80] K. K. Thornber, "Relation of Drift Velocity to Low Field Mobility and High Field Saturation Velocity", *J. Appl. Phys.*, vol. 51, no. 4, pp. 2127-2138, (1980)

[TIIK85] K. Takeda, F. Ishino, Y. Ito, R. Kasai, T. Nakashima, "A Single Chip 80-bit Floating Point Processor", *IEEE J. Solid-State Circuits*, vol. SC-20, no. 5, pp. 986-991, 1985.

[TMAS85] T. Toyabe, H. Masuda, Y. Aoki, H. Shukuri, T. Hagiwara, "Three Dimensional Device Simulator CADDETH with Highly Convergent Matrix Solution algorithms", *IEEE Trans. Computer Aided Design*, vol. CAD-4, no. 4, pp. 482-487, (1985).

[ToAs79] T. Toyabe, S. Asai, "Analytical Models of Threshold Voltage and Breakdown Voltage of Short Channel MOSFETs Derived from Two Dimensional Analysis", *IEEE J. Solid-State Circuits*, vol. SC-14, no. 2, pp. 375-382, (1979).

[TrFo77] R. R. Troutman, A. G. Fortino, "Simple Model for Threshold Voltage in a Short Channel IGFET", *IEEE Trans. Electron Dev.*, vol. ED-24, pp. 1266-1268, (1977).

[TRLG81] S. Trimberger, J. A. Rowson, C. R. Lang, J. P. Gray, "A Structured Design Methodology and Associated Software Tools, *IEEE Trans. Circuits Systems*, vol. CAS-28, no. 7, pp. 618-633, (1981).

[Trof65] F. N. Troffimenkoff, "Field Dependent Mobility Analysis of the FET", *IEEE Proceedings*, vol. 53, pp. 176-1766, (1965).

[Trou77] R. R. Troutman, "Ion-Implanted Threshold Tailoring for IGFETs", *IEEE Trans. Electron Dev.*, vol. ED-24, no. 3, pp. 182-192, (1977).

[Trou79] R. R. Troutman, "VLSI Limitations from Drain-Induced Barrier Lowering", *IEEE J. Solid-State Circuits*, vol. SC-14, no. 2, pp. 383-390, (1979).

[TuMa84] C. Turchetti, G. Masetti, "A CAD-Oriented Analytical MOSFET Model for High Accuracy Applications", *IEEE Trans. Computer Design*, vol. CAD-3, no. 2, pp. 117-122, (1984).

[Uhr84] L. Uhr, "Algorithm Structured Computer Arrays and Networks", Academic Press, (1984).

[UIMI82] Y. Uchida, T. Iizuka, J. Matsunaga, M. Isobe, S. Konishi, M. Sekine, T. Ohtani, S. Kokyama, "A Low Power Resistive Load 64 kbit CMOS RAM", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 5, pp. 804-809, (1982).

[Unge58] S. H. Unger, "A Computer Oriented Toward Spacial Problems", *IRE Proceedings*, pp. 1744-1750, October, (1953).

[Varg62] R. S. Varga, "Matrix Iterative Analysis", Prentice-Hall, (1962).

[Vars73] R. C. Varshney, "Simple Theory for Threshold Voltage Modulation in Short Channel MOS Transistors",*Electronics Letters*, vol. 9, no. 25, pp. 600-602, (1973).

[VaXu71] D. Vandorpe, N. H. Xuong, "Mathematical 2 Dimensional Model of Semiconductor Devices", *Electronics Letters*, vol. 7, pp. 47-50, (1971).

[VIVI85] Microelectronics Center of North Carolina, "Vertically Integrated VLSI System", Designer's Documentation, (1985).

[Vors81] H. A. van der Vorst, "Iterative Solution Methods for Certain Sparse Linear Systems with a Nonsymmetric Matrix Arising from PDE Problems", *J. Comp. Physics*, vol. 44, pp. 1-19, (1981).

[Vors82] H. A. van der Vorst, "A Vectorizable Variant of Some ICCG Methods", *SIAM J. Sci. Stat. Comp.*, vol. 3, no. 3, pp. 350-356, (1982).

[Vuil83] J. Vuillemin, "A Very Fast Multiplication Algorithm for VLSI Implementation", *Integration*, vol. 1, pp. 39-52, (1983).

[WaFl82] S. Waser, M. J. Flynn, "Introduction to Arithmetic for Digital Systems Designers", Holt, Rinehart and Wilson, (1982).

[Wall63] C. S. Wallace, "A Suggestion for a Fast Multiplier", *IEEE Trans. Electronic Computers*, vol. EC-13, pp. 14-17, (1963).

[WaMc82] F. Ware, W. McAllister, "CMOS Chip Set Streamlines Floating Point Processing", *Electronics*, pp. 149-152, Feb. 10, (1982).

[Wang77] P. P. Wang, "Double Boron Implant Short-Channel MOSFET", *IEEE Trans. Electron Dev.*, vol. ED-24, no. 3, pp. 196-204, (1977).

[Wang78] P. P. Wang, "Device Characteristics of Short Channel and Narrow Width MOSFETs", *IEEE Trans. Electron Dev.*, vol. ED-25, pp. 779-786, (1978).

[Wase78] S. Waser, "High Speed Monolithic Multipliers for Real Time Digital Signal Processing", *IEEE Computer*, pp. 19-29, Oct. (1978).

[WeEs85] N. Weste, K. Eshraghian, "Principles of CMOS VLSI Design, A Systems Perspective", Addison-Wesley, (1985).

[Wein67] A. Weinberger, "Large Scale Integration of MOS Complex Logic: A Layout Method", *IEEE J. Solid-State Circuits*, vol. SC-2, no. 4, pp. 182-189, (1967).

[West80] N. H. E. Weste, "MULGA-An Interactive Symbolic Layout System for the Design of Integrated Circuits", *Bell System Technical Journal,* vol. 60, no. 6, pp. 823-857, (1980).

[WhWL80] M. H. White, F. van de Wiele, J. P. Lambot, "High-Accuracy MOS Models for Computer-Aided Design", *IEEE Trans. Electron Dev.*, vol. ED-27, no. 5, pp. 899-906, (1980).

[WiFa86] B. Windsor, D. Fair, "Fast 64-Bit IEEE-754 Floating Point Multiplier and ALU", Analogue Dialog vol. 20, no. 1, pp. 7-11, (1986).

[WiPa83] T. W. Williams, K. P. Parker, "Design for Testability- Survey", *IEEE Proc.,* vol. 71, no. 1, pp. 89-112, (1983).

[WiWi84] B. Windsor, J. Wilson, "Arithmetic Duo Excels in Computing Floating Points Products", *Electronic Design*, pp. 144-151, May 17, (1984).

[WLWW84] F. Ware, R. Wong, B. Woo, C. Hansen, "Fast 64 Bit Chip Set Gangs Up for Double Precision Floating Point Work", *Electronics*, pp. 99-103, June 12, (1984).

[WMCS82] F. A Ware, w. H. McAllister, J. R. Carlson, D. K. Sun, R. J. Vlach, "64-Bit Monolithic Floating Point Processors", *IEEE J. Solid-State Circuits*, vol. SC-17, no. 5, pp. 898-907, (1982).

[WoLO83] B Woo, L, Lin, R. E. Owen, "ALU, Multiplier, Chips Zip Through IEEE Floating Point operations", *Electronics*, pp. 121-126, May 19, (1983).

[Wrig70] G. T. Wright, "Current/Voltage Characteristics, Channel Pinchoff and Field Dependence of Carrier Velocity in Silicon IGFETs", *Electronics Letters*, vol. 6, no. 4, pp. 107-109, (1970).

[Yama79] K. Yamaguchi, "Field-Dependent Mobility Model for Two-Dimensional Numerical Analysis of MOSFETs", *IEEE Trans. Electron Dev.,* vol. ED-26, no. 7, pp. 1068-1074, (1979).

[Yama83] K. Yamaguchi, "A Mobility Model for Carriers in the MOS Inversion Layer", *IEEE Trans. Electron Dev.,* vol. ED-30, no. 6, pp. 658-663, (1983).

[YKTH82] A. Yoshii, H. Kitasawa, M. Tomizawa, S. Horiguchi, T. Sudo, "A Three Dimensional Analysis of Semiconductor Devices", *IEEE Trans. Electron Dev.,* vol. ED-29, no. 2, pp. 184-189, (1982).

[YaMo83] T. Yamaguchi, S. Morimoto, "Analytical Model and Characterization of Small Geometry MOSFETs", *IEEE Trans. Electron Dev.,* vol. ED-30, no. 6, pp. 559-566, (1983).

[Yau74] L. D. Yau, "A simple theory to predict the threshold voltage of short channel IGFETs",*Solid-State Electronics*, vol. 17, pp. 1059-1063, (1974).

[YTNM85] S. Yamamoto, N. Tanimura, K. Nagasawa, S. Meguro, T. Yasui, O. Minato, T. Masuhara, "A 256K CMOS SRAM with Variable Impedance Data Line Loads", *IEEE J. Solid-State Circuits*, vol. SC-20, no. 5, pp. 924-928, 1985.

[YuTa81] S. P. Yu, W. Tantraporn, "Computer Modelling of MOSFETs", in "VLSI Electronics: Microstructure Science", N. G. Einspruch, (Editor), Academic Press, (1981).

[Zakh84] V. Zakharov, "Parallelism and Array Processing", *IEEE Trans. Computers*, vol. C-33, no. 1, pp. 45-76, (1984).

[ZiNu74] A. van der Ziel, A. Nussbaum, "Normalized Characteristics of Saturated High Field MOSFETs", *Solid-State Electronics*, vol. 17, pp. 413-414, (1974).

[ZyCo87] G. B. Zyner, P. H. Cole, "A small geometry MOSFET model for circuit simulation," *Conference Proceedings, IREECON International Convention*, Sydney Australia, September, (1987).

[ZyEs85] G. B. Zyner, K. Eshraghian, "Multiplier/Divider Designs for VLSI Signal Processing System", *Conference Proceedings, IREECON International Convention*, Melbourne Australia, October, (1985).