A Comparative Study of High Performance CMOS Multipliers, Barrel Shifters and Modeling of NBTI Degradation in Nanometer Scale Digital VLSI Circuits

THESIS

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

ABHIJIT RAMESHWAR ASATI

Under the supervision of

Dr. CHANDRA SHEKHAR

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN) INDIA

2009

A Comparative Study of High Performance CMOS Multipliers, Barrel Shifters and Modeling of NBTI Degradation in Nanometer Scale Digital VLSI Circuits

THESIS

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

ABHIJIT RAMESHWAR ASATI

Under the supervision of

Dr. CHANDRA SHEKHAR

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN) INDIA

2009

ACKNOWLEDGMENTS

I thank Prof. L. K. Maheshwari, Vice-Chancellor BITS, Pilani and Director, BITS, Pilani (Pilani Campus) for providing me the necessary infrastructure, facilities and constant inspiration.

I wish to convey my sincere gratitude to my supervisor Dr. Chandra Shekhar, Director, CEERI, Pilani, for introducing me to the area of data path exploration, providing me able guidance, helping me understand the concepts and continuously encouraging me and providing moral support.

I sincerely thank Prof. Kewal K. Saluja, professor, department of electrical and computer science engineering, university of Wisconsin Madison, for introducing me to the area of NBTI degradation in CMOS digital logic circuits and supporting me with the suitable degradation models.

I thank my Doctoral Advisory Committee (DAC) members, Prof. S. Gurunarayanan and Dr. Anu Gupta, who spared their valuable time to go through my draft thesis and were audience to my pre-submission seminar in order to provide several valuable suggestions that immensely helped in improving the quality of my PhD thesis report.

I thank Prof. K. E. Raman, Director, BITS, Pilani (Goa Campus), Prof. G. Raghurama, Deputy Director (Academics), Prof. R. K. Mittal Deputy Director (Administration), Prof. Ravi Prakash, Dean (Research and Consultancy Division), Prof. A. K. Sarkar, Dean (Instruction Division), Prof. S. Gurunarayanan, Dean (Faculty Division-II), and Prof. R.N. Saha, Dean (Educational Development Division), Prof. B. V. Babu, Dean (Engineering Hardware Division), Prof. Rajiv Gupta, Dean (Engineering Services Division), for providing necessary administrative help.

I specially thank Prof. B. R. Natarajan, Dean (DLPD), for his sincere help and affectionate enquiries about my work to boost my spirit for the research work.

I wish to express my deep gratitude to Prof. Surekha Bhanot, Unit chief (Instrumentation) for her kind and affectionate enquiries about the work and continuous encouragement.

I express my sincere thanks to Prof. V. K. Choubey, Group Leader (Electrical and Electronics Engineering Group), for giving invaluable suggestions from time to time.

I especially wish to express my deep gratitude to Dr. Anu Gupta, BITS, for providing me moral support and help in learning various related software tools.

I wish to express my sincere thanks to Dr. Shikha Tripathi, Group Leader (Instrumentation Group) for providing me help in various forms for the completion of the present work.

I wish to express my sincere thanks to Dr. S. C. Bose, Scientist, CEERI, Pilani, for providing me excellent technical suggestions on high-speed, low-power VLSI design techniques.

I wish to express my sincere thanks to Mr. Raj Singh, Scientist, CEERI, Pilani, for providing me help on cell based design approach and physical design related issues.

I thank all the senior faculty members of Electrical and Electronics Engineering Group and Instrumentation Group along with my colleagues for their kind support in completion of this research work.

I thank Dr. Hemanth Jadav, Mr. Dinesh Kumar, Ms. Monica Sharma, Mr. Sharad Shrivastava, Mr. Gunjan Soni, Mr. Amit Kumar and Ms. Sunita Bansal, nucleus members of Research and Consultancy Division, BITS, Pilani, without their cooperation and guidance it would not have been possible for me to pursue such goal oriented research during each of the past few semesters.

I also express my gratitude to the office staff of Research and Consultancy Division whose secretarial assistance helped me in submitting the various evaluation documents in time and give pre-submission seminar smoothly.

I am thankful to Mr. Pawan Sharma, in-charge, OYESTER LAB, BITS, for his invaluable help.

I am thankful to my parents, my brother, my wife and my son for their constant help and moral support.

Lastly, I express my thanks to all those who directly or indirectly contributed to the completion of this work.

Abhijit Rameshwar Asati

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI, RAJASTHAN

CERTIFICATE

This is to certify that the thesis entitled **"A Comparative Study of High Performance CMOS Multipliers, Barrel Shifters and Modeling of NBTI Degradation in Nanometer Scale Digital VLSI Circuits"** and submitted by **Mr. ABHIJIT RAMESHWAR ASATI, ID. No. 2002PHXF040**, for award of Ph. D. Degree of the Institute, embodies original work done by him under my supervision.

____

Signature in full of the Supervisor

Dr. CHANDRA SHEKHAR
(Name in capital block letters)

Director,
Central Electronics Engineering Research Institute

(Designation)

Date:

ABSTRACT

The objective of this thesis is to explore the design space of two specific data path elements (viz multipliers and barrel shifters) of different bit width at architectural-level, at logic design level, and at transistor size level to select proper architecture, logic design style and physical device sizes; keeping in a view their effects on performance (circuit delay), average power consumption and core area.

The multipliers and barrel shifters are the fundamental data path elements required in high performance 'Standard Digital Signal Processors' and 'ASIC Digital Signal Processors' used for digital signal processing (DSP). Different multiplier and barrel shifter architectures show trade-offs between propagation delay, average power consumption and transistor counts. In deep sub-micron technologies, the simple gate-level analyses are inadequate to validate particular data path architectures. In this thesis we considered the effects of wiring parasitics and MOS parasitics in the assessment of architecture. The selected word widths for different multiplier and barrel shifter architectures are 4-bit, 8-bit, 12-bit and 16-bit; which dominate in DSP applications.

A schematic and physical library consisting of functional cells was defined for static CMOS logic design style, transmission gate (TG) logic design styles, dual rail domino logic design style and true single phase clock (TSPC) logic design style. Versions of the physical libraries were developed using three different sizes of transistors. The layout assemblies for the 4-bit, 8-bit, 12-bit and 16-bit multiplier and barrel shifter circuits were carried out using these cell libraries using automatic place and route tool LEDIT (SPR) from M/s Tanner Research Inc. The circuit delay and average power dissipation then analyzed for each implementation of the multiplier and barrel shifter circuit using the same logic design style but utilizing three different physical libraries differing in their transistor sizes as described above. Maximum instantaneous power, core area, total routing length and number of vias were also analyzed for each implementation for highlighting the very large scale integration (VLSI) implementation characteristics.

Further in nanometer scale digital integrated circuits negative bias temperature instability (NBTI) related circuit performance degradation was studied. The NBTI stress makes P-channel metal oxide semiconductor (PMOS) devices slower over time due to change in their threshold voltages. In deep sub-micron technologies the NBTI degradation decides the lifetime of CMOS circuits; In this thesis we present a novel Verilog HDL based circuit modeling method that incorporates NBTI degradation dynamically. This technique will help the designers to include NBTI degradation effects in their circuit analysis efforts.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALU	Arithmetic logic unit
ASIC	Application specific integrated circuits
CLA	Carry look ahead adder
CMOS	Complimentary metal oxide semiconductor
CPA	Carry propogate adder
CSA	Carry save adder
DSP	Digital signal processing
FIR	Finite impulse response
FSM	Finite state machine
GPS	Global positioning system
IIR	Infinite impulse response
LSB	Least significant bit
MAC	Multiplier-accumulator
MSB	Most significant bit
MUX	Multiplexer
NBTI	Negative bias temperature instability
NMOS	N-channel metal oxide semiconductor
PMOS	P-channel metal oxide semiconductor
TG	Transmission gate
TSPC	True single phase clock
VLSI	Very large scale integration

CHAPTER 1

INTRODUCTION

1.1 Background

Data-paths of different bit-widths are frequently required in very large scale integrated (VLSI) circuits from processors to application specific integrated circuits (ASICs). The performance of microprocessors and computers heavily depends upon the performance of the various data-paths used. Such data paths include data registers that hold operands and results and combinational logic units that manipulate and process data values [1], [2]. Various combinational logic units are adder, multiplier, divider, barrel shifter and arithmetic logic unit (ALU) circuits. The selection of a particular data-path depends upon the state-of-the-art in digital design. The most important and widely accepted metrics for measuring the quality of data-path designs are propagation delay, area and power [2], [3]. Minimizing area and delay has always been important, but reducing power consumption has also gained importance recently because of increasing levels of integration and the desire for portability. Furthermore, the progress in battery technology is slower as compared to the ever-increasing power requirement due to advances in electronic circuits; the battery technology is unable to provide a solution to the power problem, therefore, an accurate estimation of **“**average power dissipation**”** is required to estimate battery life; also, the correct estimate of **“**peak power dissipation**”** is required to study circuit reliability. The three major sources of power dissipation in VLSI circuits are: (i) switching component of power, which is increasing due to increase in on-chip clock rates (ii) component of power due to direct-path short-circuit current in circuits, that depends upon the rise and fall times of signals and (iii) component of power due to leakage current which is increasing at an alarming rate due to thin gate oxide and small geometry effects like tunneling and drain induced barrier lowering which are dominating due to device scaling. The short-circuit dissipation of complimentary metal oxide semiconductor (CMOS) inverter with and without load (for equal rise and fall times of input and output signals) is only a fraction (< 20%) of the total dissipation. The *dominant term* in power dissipation is the switching power component, which is given by equation 1.1.

$$P_{dynamic} = C_L \bullet V_{DD}^2 \bullet f_{CLK} \bullet \eta \qquad (1.1)$$

Low-power designs, thus, aim at minimizing the power consuming transitions (switching activity factor ‘η’), power supply (V_{DD}), and load capacitance (C_L) [3], [4]. Since, the largest component of power dissipation is due to the signal transitions at circuit nodes, an accurate estimation of switching activity at the internal circuit nodes is required. Gate delays have impact on switching activity; a delay model is used for computing the Boolean conditions that cause glitches in the circuit. Glitches occur primarily due to mismatch or imbalance in the path lengths in the logic network. Such mismatches in the path lengths result in mismatches of signal timing with respect to the primary inputs [3], [5]. The probability of each gate switching at any particular time is computed from input switching rates, and then the sum of these probabilities over all the gates gives the switching activity in the entire circuit over all the time points in a clock cycle [1], [3], [6], [7]. Such probabilistic methods can’t be applied reliably to portable high performance applications, where accurate estimation of power is required.

Power dissipation due to leakage currents is gaining utmost importance due to scaling of devices. Many techniques at design and fabrication levels are applied to reduce such leakage. At transistor-level design ‘Variable Threshold CMOS Circuits’ are used and at fabrication level ‘Multiple Threshold CMOS Circuits’ and ‘HIGH-k gate oxide dielectric’ are used to reduce leakage [2], [3], [8], [9].

Portable wireless applications like mobile phones, laptop computers and personal digital assistants (PDA’s) require high-speed circuitry consuming low power. Such design requirements are conflicting and involve design tradeoffs. Furthermore the microprocessor on-chip clock rates have already reached GHz range, leading to substantial increase in dynamic (switching) power consumption. As a result in high performance desktops, sinking large amounts of heat through packages is becoming a difficult problem. Therefore, designing low-power processors is also gaining importance for high performance desktops, as well as for portable applications like laptops and palmtops where big heat sinks cannot be used. A low power processor design without greatly loosing computational speed is a technologically challenging requirement.

There are several degrees of freedom available in the design of low-power high-performance circuits and systems at various abstraction levels. These include process technology level, circuit design level, logic design level, architectural level and algorithmic level. CMOS technology, the vehicle for VLSI, offers a combination of large noise margins, ruggedness of design, low power consumption, scalability of technology and validity of the logic design styles at scaled down technologies [9], [10]. Within the

CMOS technology, designers have the freedom of choosing the architecture, the logic design style, and the transistor sizes for implementing various arithmetic functions. Besides these, technology scaling including threshold voltage scaling, and supply voltage scaling constitute other techniques that can be used in low-power digital design [11].

1.2 Objective of Thesis

It has been reported that the time spent in generating data path designs is typically 60% of the overall chip design due to the fact that it is the only major component, which is still handled manually and is a major bottleneck in the design [12]. Automatic layout generation for the data path circuits is possible [12] but 'design space exploration' is still limited. These automatic layout generation tools exploit regularity and avoid the global routing and inter-module channel routing by optimally performing signal alignments between modules during the module generation. The random selection of architecture or logic design style in VLSI design flow may lead to substantial increase in the design time due to complicated VLSI design flow from net list to layout generation in order to meet the specified design constraints. In the present study two data path elements have been considered namely signed/unsigned multipliers and barrel shifters. These data path elements are implemented as purely combinational logic circuits. The objective of this thesis is to explore the design space of these data path elements for different bit-widths at architectural-level, at logic design level (to select proper logic design style), and at transistor-level (to select proper transistor sizes) keeping in view their contributions to performance indices like average switching energy, circuit delay and area.

Presently, wide exploration has been carried out in literature for adder circuits but exploration for other data path elements is still limited. We have explored the design space of multipliers and barrel shifters (at architectural level, at logic design style level and at transistor size level). Such exploration will help the designer in choosing an optimal implementation strategy in terms of the choices of architecture, logic design style and transistor sizes. Various data path elements considered for exploration are listed in section 1.3.

Further, we have also explored one of the most important circuit reliability issues, namely, negative bias temperature instability (NBTI), which has become the deciding factor for the lifetimes of CMOS devices in deep sub-micron technologies.

1.3 Different Data Path Elements Considered for Exploration

The multiplier and barrel shifters are fundamental building blocks in 'Standard Digital Signal Processors' and 'ASIC Digital Signal Processors' used for digital signal processing (DSP) [13]. The DSP processors are provided with multiplier-accumulators (MACs) in order to perform sum-of-product computations efficiently. The high performance of these processors is achieved by using a high degree of parallelism and faster data-path architectures. Different multiplier and barrel shifter architectures provide trade-offs between gate counts, latency and speed.

1.3.1 Multipliers

Classification of various multiplier architectures is described below: [13]

Multiplier architectures are classified broadly into two categories:

- ❖ Bit serial multipliers: These multipliers are slower but take much less area and power.
- ❖ Bit parallel multiplies: These multiplier are faster but take more area and power.

  Bit parallel multipliers are further classified into following two categories:

  (i) Array type multipliers: These multipliers follow regular array structure, thereby simplifying the wiring and layout design [4], [8], [10], [14], [15], [16], [17] [18], [19]

  (ii) Tree based multipliers: These multipliers show irregular structure and therefore take larger wiring area. These multipliers use different column compression techniques namely Ofman tree, Wallace tree and Dadda tree column compression techniques [20], [21], [22], [23], [24], [25], [26], [27]

In this thesis both regular arrays and Wallace tree multipliers have been considered for exploration. Gate-level analyses suggest that Wallace trees are not only faster than array multiplier but they also consume much less power. However these analyses did not take **wiring into account**, resulting in optimistic timing and power estimates [28]. In sub-micron and deep sub-micron technologies the effect of wiring delays cannot be ignored and therefore wiring parasitics and MOS parasitics must be considered to provide an accurate assessment of a particular architecture. The selected word lengths for multiplier and barrel shifter implementations are 4-bit, 8-bit, 12-bit and 16-bit; which dominate in DSP applications. Different multipliers considered for exploration in this research work are described in section 1.6.

1.3.2 Barrel Shifters

A barrel shifter [29], [30] is a circuit that allows its input to be shifted or rotated any number of positions in either direction. For example, a 4-bit rotating barrel shifter can shift its inputs I_3, I_2, I_1, I_0 by zero, one, two or three bit positions to the right or left by using the shift control inputs S_1, S_0. The direction control bit (DIR) decides the left/right shift direction. Barrel shifter can be implemented as a purely combinational logic circuit, using conventional multiplexers (MUX), decoders, and logic gates. The sequential approach to implement the barrel shifter uses a finite state machine (FSM) and a simpler data-path. Such sequential approaches have not been considered in the present exploration.

Different combinational multiplier and barrel shifter architectures considered for exploration in the present work are described in section 1.6.

Appropriate selection of multiplier and barrel shifter units can be done efficiently by using the data available through such study.

1.4 Research Gaps

Most DSP tasks which are multiplication and shifting intensive must be performed speedily while minimizing cost and power. This requires efficient multipliers and shifters. Different multiplication algorithms differ in the manners of ‘partial product generation’ and ‘partial product addition [21]. The array multipliers have a linear time complexity and therefore their delay increases linearly with operand size n. Also it has poor space complexity O (n^2), as it requires approximately n^2 cells to produce multiplication. Therefore as the operand size grows, the circuit takes larger area and power [14], [15], [16]. A radix-y Booth encoding, where $y=2^x$ reduces the partial product rows by a factor of x. Booth radix-4 ($y=4=2^2$) encoding can reduce the number of partial product rows by a factor of two [22]. Since the number of partial product rows is reduced to half, the hardware required for multiplication is also roughly reduced by a factor of 2 [16]. In Wallace tree multipliers, since ripple effect is reduced they produce products in far less time. The time complexity is reduced to O (log n) but larger routing area is required as compared to regular array multipliers making them less suitable for VLSI implementation [16]. The advantage of reduction in hardware using Booth encoding scheme can be combined with accelerated Wallace tree accumulation of partial products to obtain the reduced time complexity of O (log n), which is well suited for large operand size multipliers [16], [22]. In sub-micron/deep sub-micron technologies for the multipliers of moderate operand sizes, where tree based architectures may degrade their performance due to larger routing lengths, some hybrid architectures [such as array of array multiplier] may show better performance [10]. These multiplier architectures have moderate routing area requirements and time complexity of $O(\sqrt{n})$ [10]. **Even though there is a body of research studies on multipliers in the literature, a systematic research study of promising high performance multipliers across architectures, logic design styles and transistor sizes does not exist.** Such a study can be of great value to multiplier designers.

Similarly different barrel shifter architectures also show tradeoffs between silicon area and speed of operations. Some architectures have a dedicated block for all the operations to be performed by the barrel shifter. They are faster, but consume larger silicon area and power. A significant reduction in area and power required by the barrel shifter circuit is achieved by implementing rightward operations as operations in leftward direction [30]. **Once again, no systematic study of barrel shifter design across architectures, logic design styles and transistor sizes exists in literature.**

In the area of circuit reliability, even though NBTI has been identified as the primary factor limiting the circuit life of CMOS circuits using deep sub-micron technology no published method exists in literature to incorporate and simulate NBTI effects dynamically in digital CMOS circuits. The CAD tools for modeling and simulation of NBTI degradation are not widely available due to this effect's complexity and emerging status [31], [32], [33]. Presently research works on NBTI is actively pursued only within the community of device and reliability physicists and leading industrial companies appear to develop their models and tools only internally to handle this effect [31], [32], [33], [34]. Considering all the above the scope of the research to be carried out under this thesis was determined as follows:

1. To study high performance multipliers (for a range of bit widths from 4-bit to 16-bit) across architectures, logic design styles and transistor sizes to gain an understanding of the optimality of various design approaches for high performance multipliers.
2. To carry out a similar study for barrel shifters.
3. To explore and propose a method of incorporating NBTI effect dynamically in logic/switch level simulation of CMOS circuits.

1.5 Research Methodology/Work-Plan

As proposed, the research work started its journey by collecting the various relevant literature items available on the various multiplier and barrel shifter architectures. This helped to understand their algorithms, architectures and their VLSI implementations. Various techniques to achieve optimum performance (i.e. low-power, high-speed and optimum area) were also studied in detail.

In the second phase, transistor level schematic libraries consisting of a standard sets of functional cells were developed for static logic design style, transmission gate (TG) logic design style, dual-rail domino logic design style and true single phase clock (TSPC) logic design style (used only for the barrel shifters). Corresponding to each schematic library, three different versions of physical library were developed by respectively sizing the W/L ratios of the N-channel metal oxide semiconductor (NMOS) transistor to values of 3, 5 and 7 (W/L values smaller than 3 were also experimented with but not considered further as they resulted in parasitic dominated slower speeds due to weak drives of transistors and were not considered good candidates for high performance). Physical libraries were implemented in 0.5 μm, N-well CMOS process (SCN_SUBM, lambda=0.3) of MOSIS. The layout assemblies for the 4-bit, 8-bit, 12-bit and 16-bit multiplier and barrel shifter circuits were carried out using these cell libraries and automatic place and route tool LEDIT (SPR) from M/s Tanner Research Inc. [35], [36].

In the third phase the generated layouts were then simulated after parasitic extraction using circuit simulator, ELDO spice. Supply voltage V_{DD} was kept at 3.3V. The product of average switching energy and circuit delay was then computed for each implementation of the selected multiplier and barrel shifter circuit using the same logic design style but utilizing three different physical libraries- differing in their transistor sizes as described above. It was noticed that for all the three logic design styles, the physical library utilizing W/L ratio of 3 for NMOS transistors gave the smallest average switching energy-delay product. A detailed comparative study was carried out for different parameters like propagation delay, transistor count, core area and power dissipation at 20MHz input/clock rate (selected for comparison purposes) across all the implementations.

In fourth phase, one of the most important circuit reliability issues, namely, NBTI [31], [32] was considered. We proposed a new technique to study the NBTI degradation using widely available Verilog HDL, which will help many designers to include NBTI effect in their designs. The NBTI is identified as most critical reliability concern for nanometer scale digital integrated circuits. Degradation occurring in P-channel metal oxide semiconductor (PMOS) devices is most critical as it decides the lifetime of CMOS devices in deep sub-micron technologies. We studied the NBTI degradation for a 1-bit full adder circuit in 90 nm technology using Verilog HDL. The circuit model describes basic static CMOS logic gates using switch-level Verilog description, which also incorporates the model for computing change in the threshold voltage (ΔV_t) and the delay (t_p) of PMOS devices after every NBTI stress. NBTI stress can be computed by knowing the time for which particular PMOS transistor remains under negative bias (i.e $V_{gs}<0$).

1.6 Thesis Structure

In this thesis a total of four multiplier architectures have been chosen for study, out of which two multiplier architectures support signed 2's complement numbers. These are Baugh Wooley multiplier and Booth encoded Wallace tree multiplier. The remaining two multiplier architectures supports unsigned numbers. These are MUX based multiplier and 2×2 cell based multiplier. The barrel shifter architectures chosen for study are Pereira's architecture and MUX-based architecture. The different logic design styles used for VLSI implementation are static CMOS logic, dual rail domino CMOS logic, TG logic and TSPC logic (only for the barrel shifter designs).

The thesis consists of seven chapters. Chapter 2 describe the design philosophy of multiplier and barrel shifter circuits, Chapter 3 explains the different multiplier and barrel shifter architectures, chapter 4 discusses the different CMOS logic design styles and determination of PMOS/NMOS width ratio (β) for high speed design, chapter 5 presents the study of the dynamics of NBTI degradation in digital logic circuits using Verilog HDL. The VLSI implementation and simulation results for different multiplier and barrel shifter circuits are tabulated and compared in chapter 6. The comparison parameters are propagation delay, average power, maximum power and leakage power, transistor count, core layout area, routing length and number of vias. The complete research work is summarized in chapter 7, which concludes the thesis. The list of references, appendix, and brief bibliography of the candidate as well as the supervisor are appended at the end of the thesis.

1.7 Chapter Summary

In this chapter we discussed about the background of the work and the objectives of the thesis. Further we described the different data path architectures considered for exploration and identified the research gaps based on detailed literature survey. The chapter also lays out the research methodology and chapter wise structure of the thesis.

CHAPTER 2

A DESIGN PHILOSOPHY OF MULTIPLIER AND BARREL SHIFTER CIRCUITS

Multiplication is the most widely used operation in many computational systems. Multiplication process is used in many neural computing and DSP applications like instrumentation and measurement, communications, audio and video processing, graphics, image enhancement, 3D rendering, navigation, radar, global positioning system (GPS), and control applications like robotics, machine vision and guidance. It is used mainly to implement algorithms like frequency domain filtering such as finite impulse response (FIR) and infinite impulse response (IIR), frequency-time transformations, correlation etc. Most DSP tasks require real-time processing; it must perform these tasks speedily while minimizing cost and power. The optimizations carried out at different levels of abstraction in the design process are typically at architectural-level, at logic level to select proper logic design style, and at transistor-level to select proper transistor sizes- keeping in view their contributions to performance indices like average power, circuit delay and area.

The multipliers used for various applications can be categorized as unsigned and signed multipliers. The popular 2's complement number representation is considered for signed multipliers. Apart from reducing the length of critical path, the VLSI implementation of multiplier circuit primarily focuses on iterative circuits with uniform interconnection pattern, which also helps in reducing the total interconnect length. The multiplication involves two basic operations i.e. the generation of partial products and their accumulation. Different multiplier algorithms differ in terms of 'partial product generation' and 'partial product accumulation', therefore their time and space complexity also varies. The complexity becomes important when operand size increases (i.e. problem size grows). The speed-up for the multiplication can be achieved using the following two techniques [13], [14], [22], [37]:

1. Reduce the number of partial products.
2. Accelerate the accumulation of partial products.

The smaller number of partial products reduces the time needed to accumulate the partial products. The accumulation process can be accelerated using faster carry save addition (CSA) technique as discussed in section 2.2.

2.1 Array Multiplier [4], [8], [18], [19], [10], [14], [15], [16], [17], [37], [38]

In this scheme, the two dimensional logic can be so organized that the partial product at (i+1)st stage is the sum of the partial products up to ith stage and the left shifted version of (i+1)st partial product. The partial products in the multiplication process may be independently computed in parallel. For a multiplicand $A=A_{n-1}\ A_{n-2}\ \ldots\ldots A_0$, and multiplier $B=B_{n-1}\ B_{n-2}\ \ldots\ldots B_0$ the product P=A.B is given by equation 2.1.

$$P=\sum_{i=0}^{n-1} B_i \cdot 2^i \cdot A=\sum_{i=0}^{n-1} 2^i \sum_{j=0}^{n-1} B_i \cdot A_j \cdot 2^j \qquad (2.1)$$

The ith partial product sum PP_i can be denoted by equation 2.2

$$PP_0=0$$

$$PP_1=PP_0+B_0A$$

$$PP_2=PP_1+B_1 2^1A$$

$$PP_3=PP_2+B_2 2^2A$$

.

.

.

$$PP_{i+1}=PP_i+B_i 2^iA \qquad (2.2)$$

Example 2.1 shows the simple multiplication process. The figure 2.1 shows the simple array multiplier cell. This simple cell is used repetitively and arranged to realize a complete array multiplier circuit as shown in figure 2.2. The critical path in multiplier circuit is also indicated. Reduction in the length of critical path is one of the major objectives in any combinational circuit design.

Example 2.1:
A= Multiplicand
B= Multiplier

```
        A:     1101=13
        B:     1011=11
  ----------------------
               1101
              1101+
             0000++
            1101+++
  ----------------------
           10001111=143
```

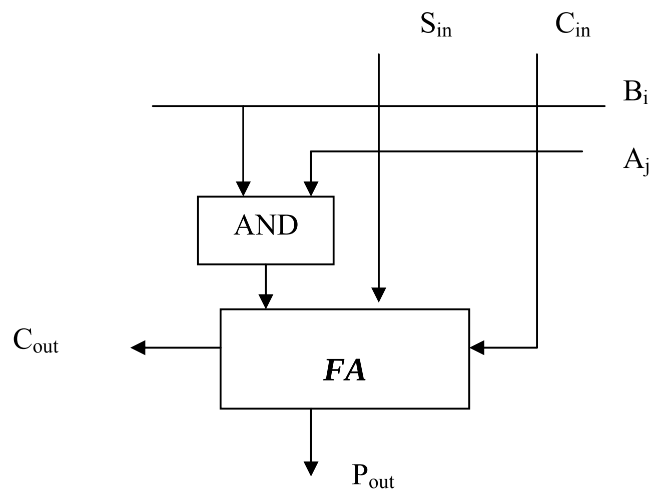


Figure 2.1 Array multiplier cell

Where,

S_{in} is the incoming sum bit and C_{in} is the incoming carry bit being summed along with the partial product bit B_i A_j by the full adder.

Advantages:

- Regular circuit structure; hence suitable for VLSI implementation.
- The length of interconnects is reduced appreciably.
- Execution is comparatively faster than sequential multiplication.
- Such architecture can be faster for smaller operand size multipliers.

Disadvantages:

- As the operand size grows the circuit takes large area and power due to space complexity O (n^2).
- Execution is slow due to ripple effect in partial product addition.
- These multipliers have time complexity of O (n).
- Due to linear time complexity, delays may not be acceptable for larger operand size multipliers.

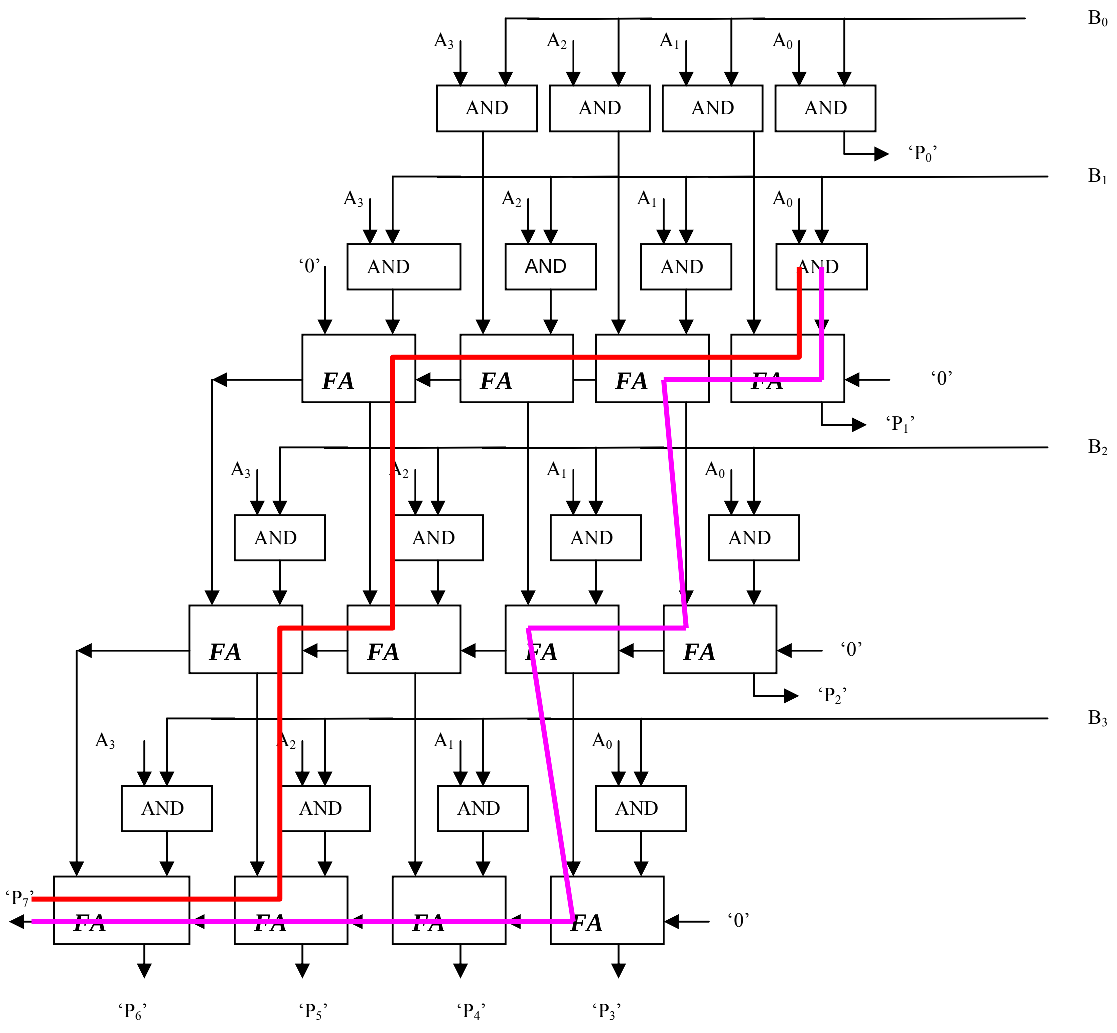


Figure 2 .2 A 4x4 array multiplier showing critical path

2.2 Wallace Tree [22], [37], [39]

A Wallace tree scheme accelerates the accumulation using faster CSA technique. CSA is one of the major speed enhancement techniques used in modern digital multiplier circuits due to its ability to add numbers with minimal carry propagation. Using 3:2 compressors, three numbers can be reduced to two using simple addition while keeping their carries and the sum separate. This means that all of the columns can be added in parallel without waiting for the result of the previous column. The two outputs that the adder generates as sum and carry can be compressed further in next stage. In the last stage, two rows of sum and carry can be added using carry propagate adder (CPA) to obtain final product. Example 2.2 shows accumulation of partial product in a Wallace tree multiplier.

Example 2.2:

```
         10111001
         00101010
         00111001
    Sum: 10101010
  Carry: 00111001X
 Result:100011100
```

Advantages:

- Since ripple effect is reduced, it produces the product in far less time.
- The time complexity is reduced to O (log n).

Disadvantages:

- Requires more hardware to accumulate partial product bits as compared to array multipliers (which are regular).
- Takes larger routing area as compared to regular array multipliers; hence less suitable/adaptable for VLSI implementation.

Example 2.3 shows the addition of partial product rows in a Wallace tree multiplier. The addition can be performed using fast CSA as shown in figure 2.3.

Example 2.3:

A= 111 (7)
B= 110 (6)

00000 (PP_0)
01110 (PP_1)
11100 (PP_2)

101010 (Final Sum Calculated)

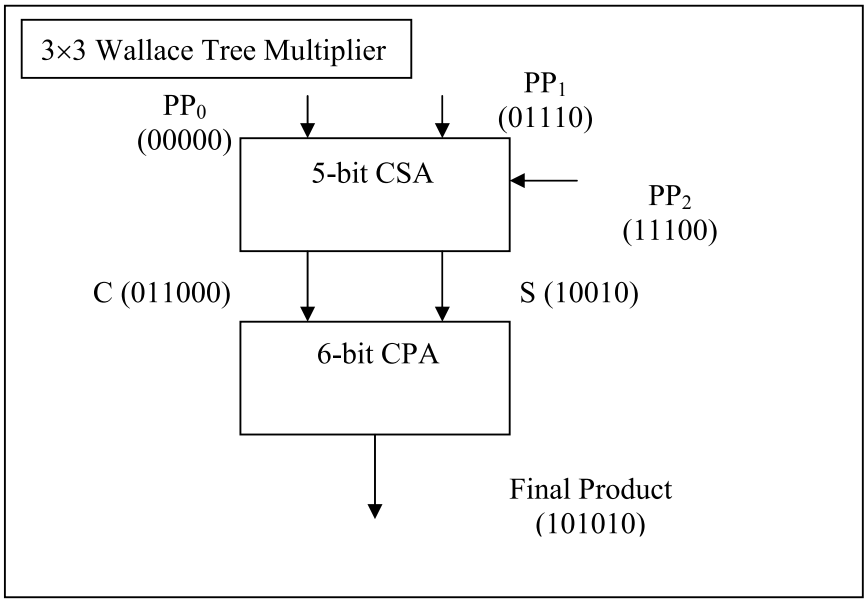


Figure 2.3 Accumulation of partial product in a Wallace tree

2.3 Booth's Algorithm [37], [39]

The motivation of this scheme is to "**speed up**" multiplication process by reducing the number of partial products. It is not essential to execute add and shift for each '1' multiplier bit.

A multiplier (say) B = 001110011110 can be treated as:

0	0	1	1	1	0	0	1	1	1	1	0
0	+1	0	0	-1	0	+1	0	0	0	-1	0

Instead of add and shift operation for each bit that is '1' in the multiplier the '0' and '1' multiplier bits can be replaced by +1, 0, or –1 as noted.

- +A or –A with appropriate shift need to be added to the partial product corresponding to +1 or –1.
- Thus the number of ADD operations is reduced from 7 to only 4 ADD/SUB operations.
- Rules for first order Booth encoding are given in table 2.1.

Table 2.1 Rules for radix-2 Booth encoding

B_i	B_{i-1}	Comments	Y_i
0	0	String of zeros	0
1	1	String of ones	0
1	0	Beginning of a string with ones	-1
0	1	End of a string with ones	1

Such encoding is useful for sequential multiplier design, since it reduces the number of addition operations to be performed. The Booth encoding in combinational circuit design can be explained by example 2.4.

Example 2.4:

- A multiplier of 14 (0,1,1,1,0) can be treated as (+1,0,0, -1,0) i.e. (16-2)
- A multiplier of 15 (0,1,1,1,1) can be treated as (+1,0,0,0, -1) i.e. (16-1)

This is called radix-2 Booth encoding. A radix-y Booth encoding, where $y=2^x$ reduces the number of partial product rows by factor of x. Therefore radix-2 Booth encoding does not reduce the number of partial product rows. Example 2.5 shows that radix-2 Booth multiplier does not show any advantage in combinational multiplier design, since the number of partial product rows remains unreduced and algorithm puts additional burden of negation of a multiplier. Thus radix–2-Booth encoding is costlier than normal multiplier.

Example 2.5:

Multiplicand = A= –5 = 1011 ($\Rightarrow$ -A = +5=0101)

Multiplier = B = –3 =1101

Multiplicand (A)					1	0	1	1
Multiplier (B)					1	1	0	1
Recoded multiplier (Y)					0	-1	1	-1
PP_1		0	0	0	0	1	0	1
PP_2		1	1	1	0	1	1	-
PP_3		0	0	1	0	1	-	-
PP_4		0	0	0	0	-	-	-
Sum		0	0	0	1	1	1	1

To get rid of the above problem higher radix Booth algorithms are used. Booth radix-4 ($y=4=2^2$) encoding can reduce the number of partial product rows by a factor of two. For radix-4 Booth encoding, we are encoding each pair as operation like 0, ±A, ±2A, this will reduce the number of rows of partial products in combinational circuit by a factor of two. In this algorithm B_i and B_{i-1} is recoded as Z_i and Z_{i-1} but taking into account the B_{i-2} bit. The algorithm to convert a sequence is given in the table 2.2.

Table 2.2 Rules for radix-4 Booth encoding

B_i	B_{i-1}	B_{i-2}	Z_i	Z_{i-1}	D_j	Operation
0	0	0	0	0	+0	+0
0	0	1	0	1	+1	+A
0	1	0	0	1	+1	+A
0	1	1	1	0	+2	+2A
1	0	0	-1	0	-2	-2A
1	0	1	0	-1	-1	-A
1	1	0	0	-1	-1	-A
1	1	1	0	0	+0	+0

For the case of radix-4 Booth encoding a signed binary number in two's complement form is partitioned into overlapping groups of three bits (in general for the radix-y Booth encoding the overlapping group takes x+1 bits) and each group is represented by possible value of 0, +A, -A, +2A, -2A using the rules indicated in the table 2.2. The value of overlapping groups of three bits can also be computed easily using the equation 2.3, which computes digits D_j as shown in table 2.2. Use of this algorithm reduces the number of partial product rows to half. Example 2.6 shows the grouping of multiplier bits in a radix-4 Booth encoding.

Example 2.6:
Multiplier (B) = 11010 = $B_4B_3B_2B_1B_0$

Comments	**Multiplier B**								**j=(i-1)/2**	**Operation**
	B_5	**B_4**	**B_3**	**B_2**	**B_1**	**B_0**	**B_{-1}**	**i**		
B is sign extended	**1**	**1**	**1**	**0**	**1**	**0**	**0**			
Group 0					**1**	**0**	**0**	**1**	**0**	**(-2A)**
Group 1			**1**	**0**	**1**			**3**	**1**	**(-A)**
Group 2	**1**	**1**	**1**					**5**	**2**	**(0)**

$$coded\ \ digit\ D_j = -2^{n-1}B_{nj+1}........+2^1 B_{nj+1} + 2^0 B_{nj} + 2^0 B_{nj-1}$$

$$for\ n = 2:$$

$$coded\ \ digit\ D_j = -2^1 B_{2j+1} + 2^0 B_{2j} + 2^0 B_{2j-1}$$

$$\underline{for\ j = 0}$$

$$coded\ \ digit\ D_0 = -2^1 B_1 + 2^0 B_0 + 2^0 B_{-1}$$

$$\underline{for\ \ j = 1}$$

$$coded\ \ digit\ D_1 = -2^1 B_3 + 2^0 B_2 + 2^0 B_1$$

$$\underline{for\ \ j = 2}$$

$$coded\ \ digit\ D_2 = -2^1 B_5 + 2^0 B_4 + 2^0 B_3 \tag{2.3}$$

Example 2.7 shows the multiplication process for radix-4 Booth algorithm. In this method the number of bits should always be even. If the number of bits is odd then sign bit is extended at the most significant bit (MSB) position. The example shows that the number of partial product rows is reduced by a factor of 2.

Example 2.7:
Multiplicand A = +13 = 01101 (⇒ -A = 10011)
Multiplier B= -6 = 11010 = 111010 (sign extended to 6-bit)

Modified multiplier bits:
i = 5 ⇒ Z_5Z_4 = (0 0), i =3 ⇒ Z_3Z_2 = (0 –1) and i = 1 ⇒ Z_1Z_0= (–1 0)

OR {since, i=2j+1 ⇒ j=(i-1)/2}

j = 2 ⇒ D_2=0, j = 1 ⇒D_1 = -1, j = 0 ⇒ D_0 = -2

Operation to be performed = 0, –A, –2A (From Table 2.2)

```
              0   0   1   1   0   1        (+13)
              0   0   0  –1  –1   0        (-6)
     ------------------------------------------------
  1   1   1   1   0   0   1   1   0        (-26)
  1   1   1   0   0   1   1   +   +        (-52)
  0   0   0   0   0   +   +   +   +         (0)
  ---------------------------------------------------
  1   1   0   1   1   0   0   1   0        (-78)
```

Note: Number of partial product rows have been reduced by a factor of 2

Advantages:

- Number of partial product rows is reduced to half. This also implies that the hardware required to generate partial products is reduced to $n^2/2$ cells.
- The advantage of reduction in hardware using Booth encoding scheme can be combined with accelerated Wallace tree accumulation of partial products to obtain the reduced time complexity of O (log n).
- Such Booth encoded tree multipliers are highly suitable for large operand size multipliers.

Disadvantages:

- The Booth encoding requires extra hardware and generation of partial products becomes complex due to increased number of operations on multiplicand A.
- The Booth encoder circuit adds an extra delay to critical path hence for smaller operand-size multipliers the performance may degrade.

2.4 Array of Array Multiplier [10]

As discussed earlier, array multipliers are preferred for smaller operand sizes due to their simpler VLSI implementation, in-spite of their linear time complexity. On the other hand the tree-based multipliers have better time complexity as compared to array based multipliers but are less suitable for VLSI implementation; since being less regular they require larger total routing length, which may degrade their performance. Some hybrid architectures have area and latency characteristics in between the two extremes. These are called array of array based schemes, which have routing area requirements close to an array multiplier and time complexity of $O(\sqrt{n})$.

2.5 Different Multiplier Architectures Considered for Exploration

The different multiplier architectures to be studied based on the above points of view are listed below.

(1) Baugh Wooley Multiplier [8]

(2) Booth Encoded Wallace Tree Multiplier [22], [40], [41]

(3) MUX Based Multiplier [16]

(4) 2×2 Cell Based Multiplier [10], [42]

The multipliers (1) and (2) are signed multipliers while multipliers (3) and (4) are unsigned multipliers. The detailed design implementation for these multiplier architectures are discussed in chapter 3. These architectures have been considered for exploration in the present study. The performance and characteristic parameters for comparison purposes are propagation delay, average power, maximum power and leakage power, transistor count, core layout area, routing length and number of vias.

2.6 Barrel Shifter Design Philosophy and Architectures Considered for Exploration

Data shifting is a requirement of many key computer operations- from address generation to arithmetic functions. Shifting a single data bit one field at a time can be a slow process; this is where a barrel shifter comes in. A barrel shifter is a combinational logic device/circuit that can shift or rotate a data word by desired number of bits in a single operation. It is another most important block in DSP processor circuits. Intel 80386 and Motorola 68030 chips have also utilized the barrel shifter circuit in their design. It is used for floating-point normalization, word pack/unpack, and field extraction from a bit stream, editing, data modification, and arithmetic manipulation.
Different barrel shifter architectures show tradeoffs between silicon area and speed of operations. Some architectures have a dedicated block for all the operations to be performed by the barrel shifter. They are faster, but consume larger silicon area and power. A significant reduction in area and power required by the barrel shifter circuit is achieved by implementing rightward operations as operations in leftward direction [30]. The barrel shifter architectures to be studied from the above points of view and included in our studies are listed below.

(1) MUX Based Barrel Shifter [43]

(2) Pereira’s Barrel Shifter [29]

The detailed design implementation for these barrel shifter architectures are discussed in chapter 3.

2.7 Chapter Summary

In this chapter we present the design philosophy of the multiplier and barrel shifter circuits for achieving high performance. The chapter presents the schematic structure and highlights the associated advantages and disadvantages of array multipliers, Wallace tree multipliers, Booth’s algorithm and array of array multipliers. Furthermore, the chapter proposes four high-speed multiplier architectures for further investigation. Similarly, it proposes two high-speed barrel shifter architectures for further investigation.

CHAPTER 3
DIFFERENT MULTIPLIER AND BARREL SHIFTER ARCHITECTURES

Basics of multiplier design have been already discussed in chapter 2. In this chapter we present details of architecture and logic implementations in respect of four different multipliers and two different barrel shifters selected for exploration.

The multipliers selected are:

(1) Baugh Wooley Multiplier [8]

(2) Booth Encoded Wallace Tree Multiplier [22], [40], [41]

(3) MUX Based Multiplier [16]

(4) 2×2 Cell Based Multiplier [10], [42]

Multiplier (1) and (2) are signed multipliers and multiplier (3) and (4) are unsigned multipliers. The signed multipliers follow the 2's complement number representation. The barrel shifter architectures considered for explorations are:

(1) MUX Based Barrel Shifter [43]

(2) Pereira's Barrel Shifter [29]

All the multipliers as well as barrel-shifter architectures are implemented for 4-bit, 8-bit, 12-bit and 16-bit sizes using four different logic design styles.

3.1 Baugh-Wooley Multiplier

The Baugh-Wooley multiplier is a s**igned array multiplier,** which utilizes 2's complement number system in the implementation of multiplication algorithm. Partial products are adjusted to maximize regularity of multiplication array. Algorithm moves partial product terms with negative signs to the last steps, where it adds (-2^{2N-1}) to other partial product terms to get sum of partial products as shown in equation 3.4, which is derived using equation 3.1, equation 3.2 and equation 3.3.

Design implementation for a 4×4 multiplier: As explained earlier in a 4×4 multiplier the sign-bit of product carries a weight of -2^7. The design procedure for a 4×4 multiplier utilizes equation 3.1, equation 3.2 and equation 3.3 and derives the terms T_1, T_2, T_3, T_4, T_5 and T_6 (where, the first term $T_1=-2^7$) such that their summation generates the final product in 2's complement notation as shown by equation 3.4

$$A = -A_3 2^3 + (A_2 2^2 + A_1 2^1 + A_0 2^0) \tag{3.1}$$

$$B = -B_3 2^3 + (B_2 2^2 + B_1 2^1 + B_0 2^0) \tag{3.2}$$

$$\begin{aligned} So,\ P = AB = A_3 B_3 2^6 - A_3 (B_2 2^2 + B_1 2^1 + B_0 2^0)\cdot 2^3 - B_3 (A_2 2^2 + A_1 2^1 + A_0 2^0) 2^3 \\ + (A_2 2^2 + A_1 2^1 + A_0 2^0)(B_2 2^2 + B_1 2^1 + B_0 2^0) \end{aligned} \tag{3.3}$$

$$\begin{aligned} &Let, \\ &-(B_2 2^2 + B_1 2^1 + B_0 2^0) = -2^3 + B_2^{'} 2^2 + B_1^{'} 2^1 + B_0^{'} 2^0 + 2^0 \\ &[e.g.,\ -111 = -1000 + 000 + 1;\ -000 = -1000 + 111 + 1] \\ &So,\ P = AB = A_3 B_3 2^6 - A_3 2^6 + A_3 (B_2^{'} 2^2 + B_1^{'} 2^1 + B_0^{'} 2^0)\cdot 2^3 + A_3 2^3 \\ &-B_3 2^6 + B_3 (A_2^{'} 2^2 + A_1^{'} 2^1 + A_0^{'} 2^0)\cdot 2^3 + B_3 2^3 \\ &\qquad\qquad + (A_2 2^2 + A_1 2^1 + A_0 2^0)(B_2 2^2 + B_1 2^1 + B_0 2^0) \end{aligned}$$

$$\begin{aligned} &again, \\ &-A_3 2^6 = -2^7 + A_3^{'} 2^6 + 0^{'} 2^5 + 0^{'} 2^4 + 0^{'} 2^3 + 0^{'} 2^2 + 0^{'} 2^1 + 0^{'} 2^0 + 1 \\ &\qquad\quad = -2^7 + A_3^{'} 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + 1 = -2^6 + A_3^{'} 2^6 \\ &-B_3 2^6 = -2^7 + B_3^{'} 2^6 + +2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 + 1 = -2^6 + B_3^{'} 2^6 \\ &\Rightarrow -A_3 2^6 - B_3 2^6 = -2^6 (1+1) + (A_3^{'} + B_3^{'})\cdot 2^6 \\ &\Rightarrow -A_3 2^6 - B_3 2^6 = -2^7 + (A_3^{'} + B_3^{'})\cdot 2^6 \end{aligned}$$

$$
\begin{aligned}
P = AB = & -2^7 && T_1 \\
& +(A_3B_3 + A_3' + B_3')\cdot 2^6 && T_2 \\
& +\mathrm{A}_3(B_2'2^2 + B_1'2^1 + B_0'2^0)\cdot 2^3 && T_3 \\
& +\mathrm{B}_3(A_2'2^2 + A_1'2^1 + A_0'2^0)\cdot 2^3 && T_4 \\
& +(A_3 + B_3)\cdot 2^3 && T_5 \\
& +(A_2 2^2 + A_1 2^1 + A_0 2^0)(B_2 2^2 + B_1 2^1 + B_0 2^0) && T_6
\end{aligned}
\tag{3.4}
$$

The final product obtained by adding T_1, T_2, T_3, T_4, T_5 and T_6 is then rearranged to produce product bits P_0 to P_7 as shown in equation 3.5, which helps the designer to implement the combinational logic circuit for 4×4 multiplier as shown in figure 3.1. For this multiplier, A_i & B_i are two 4-bit input vectors and P_k & C_{out} are outputs of 8-bit and 1-bit length respectively. The final carry bit C_{out} is discarded.

Similar technique is followed in design of 8×8, 12×12 and 16×16 multipliers.

$$
\begin{aligned}
P_0 &= +2^0(A_0B_0) \\
P_1 &= +2^1(A_1B_0 + A_0B_1) \\
P_2 &= +2^2(A_2B_0 + A_1B_1 + A_0B_2) \\
P_3 &= +2^3(A_2B_1 + A_1B_2 + A_3 + B_3 + A_3B_0' + A_0'B_3) \\
P_4 &= +2^4(A_2B_2 + A_3B_1' + B_3A_1') \\
P_5 &= +2^5(A_3B_2' + A_2'B_3) \\
P_6 &= +2^6(A_3B_3 + A_3' + B_3') \\
P_7 &= -2^7
\end{aligned}
\tag{3.5}
$$

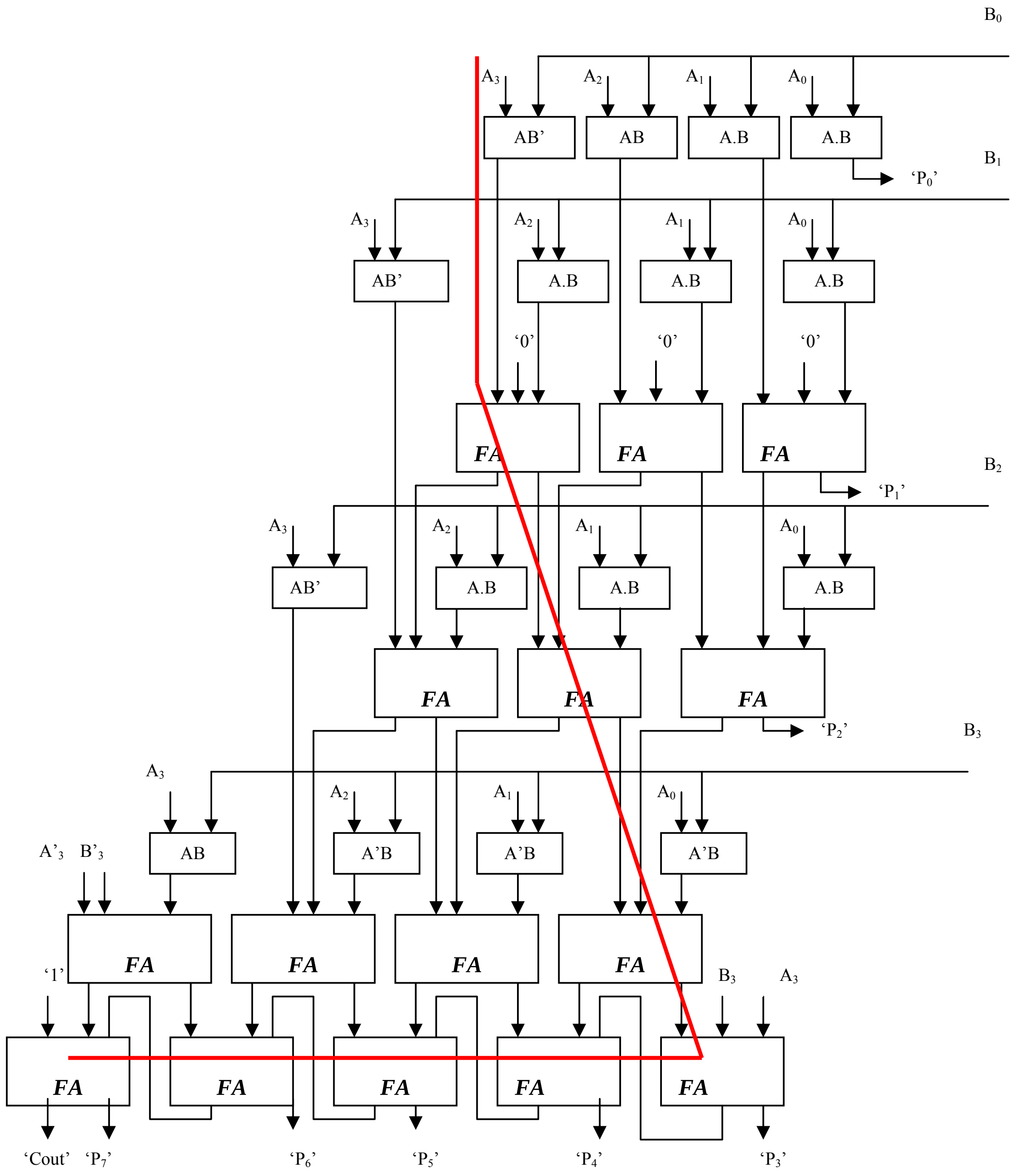


Figure 3.1 A 4 ×4 Baugh Wooley multiplier

3.2 Booth Encoded Wallace Tree Multiplier

The basic operation of radix-2 and radix-4 Booth encoding has already been discussed in chapter 2. This section presents design implementation for a M×N radix-4 Booth encoded optimized Wallace tree multiplier. In this multiplier design, the partial product at each bit position is compressed into sum and carry signals, which are then added to give the final output as explained later in section 3.2.2.

3.2.1 Multiplication Logic

The multiplier's main blocks are multiplier/multiplicand selector block, the modified Booth encoder block, partial product generator block, Wallace tree section (which adds all the partial products simultaneously to produce final two rows of sum and carry). The final two rows of sum and carry are then added using CPA.

Mao-Sorley proposed modified Booth 3-bit recoding. Application of this method of recoding to an M-bit two's complement binary number B is shown in equation 3.6. An equivalent base 4 redundant sign digit representation is obtained as B' is shown in equation 3.7.

$$B = -2^{M-1} B_{M-1} + \sum_{i=0}^{M-2} B_i 2^i \tag{3.6}$$

$$B' = \sum_{i=0}^{\frac{M}{2}-1} D_i 4^i \tag{3.7}$$

The digits D_i are chosen from the set +2,+1,0,-2,-1 according to the rules in table 3.1 At each step 3-bits of B i.e.$B_{2i+1}B_{2i}B_{2i-1}$, are examined and table 3.1 is referred to for selecting D_i. The multiplier B is always appended on the right by a zero (i.e. $B_{-1}=0$) and M is always even.

Table 3.1 - Booth encoder truth table

$B_{(2i+1)}$	$B_{(2i)}$	$B_{(2i-1)}$	D_i
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	-2
1	0	1	-1
1	1	0	-1
1	1	1	0

In the M-bit by N-bit multiplication, the M-bit multiplier number B is Booth encoded. The product AB is then obtained by adding M/2 partial product rows as shown in equation 3.8. These partial products rows are calculated easily by shifting and/or complementing the multiplicand A.

$$AB = \sum_{i=0}^{\frac{M}{2}-1} D_i 4^i A \qquad (3.8)$$

$$AB = \sum_{i=0}^{\frac{M}{2}-1} P_i . 4^i \qquad (3.9)$$

$$Where, \quad P_i = D_i A \quad \left(i = 0,\ 1,\ 2......\left(\frac{M}{2} - 1 \right) \right)$$

When adding the M/2 partial product rows P_i, the P_{i+1}st partial product row is placed two bits to the left of partial product row P_i as indicated by equation 3.9. However, all P_i rows must be sign extended to the (M+N)th binary position, such a sign extension becomes very costly in terms of hardware. In order to avoid excessive hardware required for sign extension, we use sign extension prevention technique, which assumes that all of the partial products are negative**.** In this case, the sum of all sign extensions can be pre-calculated as shown in equation 3.10.

$$Signs = \sum_{i=0}^{\frac{M}{2}-1}(-1)2^N 4^i$$

$$Signs = 2^N(-1)\frac{(2^M-1)}{3} \qquad (3.10)$$

Equation 3.10 suggests that a fixed number, (-1) (2^M – 1)/3, should be added to the (un-extended) partial products, starting from the N^{th} binary position leftwards as shown in table 3.2, which depicts the logical operation of a 4×4 multiplier. If it turns out that a partial product $P_i = D_i A$ is indeed positive, we simply replace its sign bit with a one to undo the effect of our earlier assumption that the partial product P_i is negative, that is we will take 2's complement of sign extended portion (i.e. add jS'='1' at sign bit to invert all 1's and add jS'='1' at the beginning of partial product row as shown in example 3.1). Equation 3.12 explains the generation of jS' bit.

Table 3.2 Logical operation of a Booth encoded 4×4 multiplier assuming that all of the partial products are negative

				A_3	A_2	A_1	A_0
					B_2		B_0
1	**1**	**1**	**1**	$P_{3,0}$	$P_{2,0}$	$P_{1,0}$	$P_{0,0}$
1	**1**	$P_{3,2}$	$P_{2,2}$	$P_{1,2}$	$P_{0,2}$		
Sum of sign bits							
1	**0**	**1**	**1**				
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

A fixed number = (-1) (2^M – 1)/3 = (-5) = 1011 is added at the position indicated in table 3.3.

The original algorithm does not generate correct result if particular partial product row is '0', since it is neither positive nor negative. Hence for 0×0 it gives wrong output. The algorithm is slightly modified as shown in table 3.3.

Example3.1:

Let A=(-3) $_{10}$=(1101) $_2$ and B=(22) $_{10}$=(010110) $_2$

Booth encoding of B gives:- (010), (011), (100) =A, 2A, -2A

Note: Multiplier to be Booth encoded has size of M-bit and multiplicand has size of N-bit.

M = 6, N = 4 ⇒ signs = $(336)_{10}$= $(1010110000)_2$

Or fixed number = (-1) (2^6 – 1)/3 = (-21) = 101011 is added at position indicated.

					(0S'=1)	0	1	0	1
									(0S'=1)
			(2S'=0)	1	0	1	0	+	+
							(2S'=0)		
	(4S'=0)	1	1	0	1	+	+	+	+
					(4S'=0)				
1	0	1	0	1	1				
1	1	1	0	1	1	1	1	1	0

Table shows that if partial product is non zero then jS' is selected for addition at both MSB and least significant bit (LSB) positions and if partial product row is zero then jM_{VDD}='1' is added at MSB position and jM_{GND}='0' is added at LSB position. The prevention of sign extension plays an important role in hardware reduction of a multiplier architecture design based on Booth encoding. Equation 3.13 explains the generation of jM_{VDD} and jM_{GND} bits.

Table 3.3 Modified logical operation of Booth encoded 4×4 multiplier

				A3	A2	A1	A0
					B2		B0
			0S'/0M_{VDD}	$P_{3,0}$	$P_{2,0}$	$P_{1,0}$	$P_{0,0}$
	2S'/2M_{VDD}	$P_{3,2}$	$P_{2,2}$	$P_{1,2}$	$P_{0,2}$		0S'/0M_{GND}
1	0	1	1		2S'/2M_{GND}		
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

3.2.2 Implementation Technique

This section presents the design procedure for implementing the M×N multiplier as shown in table 3.3. The multiplier has three design levels:

The first level in design is the part where the partial products are generated using the Booth encoded multiplier. The second level in deign is the accumulation of the partial products using 3:2 and 4:2 compressors (as explained later in section 3.2.2.5), which generate intermediate sum and carry. The intermediate sum and carry of the second level are further accumulated using 3: 2 compressors that are essentially full adders to generate final row of sum and carry. The third level in design accumulates sum and carry obtained by the second level, using a (M+N) bit CPA , which gives the final output of (M+N) bits. Figure 3.2 explains all three levels in the design of Booth encoded multiplier.

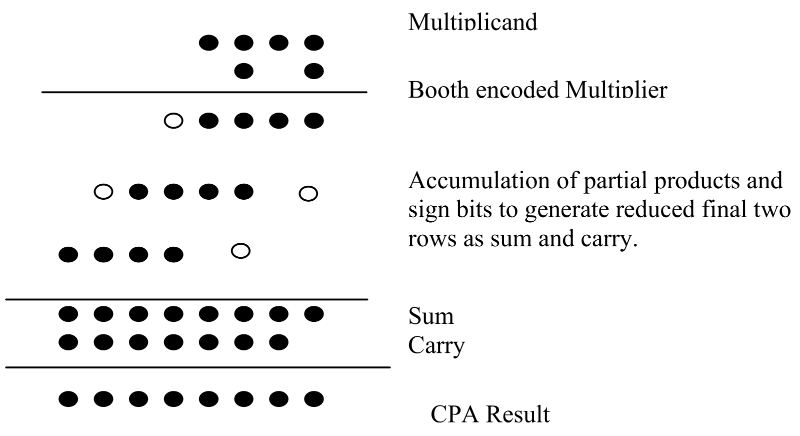

Multiplicand
Booth encoded Multiplier
Accumulation of partial products and sign bits to generate reduced final two rows as sum and carry.
Sum
Carry
CPA Result

Figure 3.2 Basic Steps in Booth multiplication

The functions of different design blocks used in Booth encoded Wallace tree multiplier are explained in sections from 3.2.2.1 to 3.2.2.6.

3.2.2.1 Modified Booth Encoder

The multiplier is Booth encoded using the Booth encoder to generate the Booth encoded bits (PL_j, M_j, X_j, $2X_j$). The PL_j, M_j, X_j and $2X_j$ are derived from the basic Booth algorithm. Table 3.4 shows the Booth encoding for all the input combinations.

Where, PL_j =1, $\Rightarrow$ positive partial product

M_j =1, $\Rightarrow$ negative partial product

X_j =1, $\Rightarrow$ partial product neither doubled and nor zero

$2X_j$ =1, $\Rightarrow$ partial product doubled

Table 3.4 Modified Booth encoding

Inputs			Function	Sign Select			
B_{j+1}	B_j	B_{j-1}		X_j	$2X_j$	PL_j	M_j
0	0	0	0	0	1	0	0
0	0	1	+A	1	0	1	0
0	1	0	+A	1	0	1	0
0	1	1	+2A	0	1	1	0
1	0	0	-2A	0	1	0	1
1	0	1	-A	1	0	0	1
1	1	0	-A	1	0	0	1
1	1	1	0	0	1	0	0

After logic optimization using Karnaugh maps equations obtained are shown as equation 3.11. Using these equations the combinational circuit can be designed to generate Booth-encoded bits (PL_j, M_j, $2X_j$). *(Note that X_j bit is not evaluated, as it can be eliminated in generation of partial products as explained in next section.)*

$$\begin{aligned}
&2X_j = (B_j \oplus B_{j-1})' \\
&M_j = (B_j \bullet B_{j-1})' \bullet B_{j+1} \\
&PL_j = B'_{j+1} \bullet B'_j \bullet B'_{j-1} \\
&Where,\ \ j = 0, 2..........M-2
\end{aligned} \tag{3.11}$$

3.2.2.2 Sign Bit Generator

This generates the sign bit as shown in table 3.3, which removes the need for sign extension. This takes PL_j, M_j and MSB of A input as input and gives jS' as output. M_j+PL_j='1' means partial product is non-zero. MSB of A='1' means multiplicand is negative. If M_j='1' then jS will be zero meaning partial product is positive and therefore jS'=not (jS)='1' is added at the appropriate positions as shown in table 3.3. The logic equation used for implementing sign bit generator is shown in equation 3.12. Thus this sign bit is valid for non-zero partial products and when multiplicand is negative.

$$
\begin{aligned}
& jS = [(M_j + PL_j) \bullet (MSB\ of\ A)] \oplus M_j \\
& jS' = not(jS) \\
& Where, \\
& j = 0, 2 \ldots\ldots\ldots\ldots M-2 \qquad (3.12)
\end{aligned}
$$

Table 3.3 shows that if partial product is non zero then jS' is selected for addition at both MSB and LSB positions and if partial product row is zero then jM_{VDD}='1' is added at MSB position and jM_{GND}='0' is added at LSB position. M_j+PL_j= '0' means partial product is zero, for such cases '1' is placed at jM_{VDD} location and '0' at jM_{GND} locations as shown in table 3.3, which are taken care by jM_{VDD} and jM_{GND} bits. The generation of jM_{VDD} and jM_{GND} bits can be obtained by modifying equation 3.12. The modified equation 3.13 takes care of the case of zero partial products for which original algorithm gives wrong results.

$$
\begin{aligned}
& jM_{VDD} = (M_j + PL_j) jS' + (M_j + PL_j)' V_{DD} \\
& jM_{GND} = (M_j + PL_j) jS' + (M_j + PL_j)' G_{ND} \\
& Where, \\
& j = 0, 2 \ldots\ldots\ldots\ldots M-2 \qquad (3.13)
\end{aligned}
$$

3.2.2.3 Partial Product Generator

The partial products are generated using the partial product generator. The bits of partial products generated at different bit positions are accumulated. When there is a single partial product then the simple partial product generator is enough. For generating and accumulating 2 or 3 or 4 partial products, 2D, 3D and 4D units are used respectively, which generate the respective number of partial products and then compress them into sum and carry signals. Partial product bit at position (i,j) as given by reference [41] are shown in equation 3.14.

$$P_{i,j} = X_j(A_i PL_j + A_i^{'} M_j) + 2X_j(A_{i-1} PL_j + A_{i-1}^{'} M_j)$$

$$\begin{aligned}&Where,\\ &i = 0,1,2......(N-1)\\ &j = 0,2,4....(M-4),(M-2)\end{aligned} \qquad (3.14)$$

Equation 3.14 can also be implemented using two 2:1 MUX and two XOR gates as shown by equation 3.15 (this eliminates the X_j input, which is complement of input $2X_j$).

$$P_{i,j} = M_j \oplus [(M_j \oplus PL_j) \bullet ((2X_j)^{'} A_i + (2X_j) A_{i-1})]$$

$$\begin{aligned}&Where\ ,\\ &i = 0,1,2......(N-1)\\ &j = 0,2,4....(M-4),\ (M-2)\end{aligned} \qquad (3.15)$$

Equivalace between 3.14 and 3.15 has been shown in table 3.5

Table 3.5 Equivalace between two different patial product generation logics

X_j	$2X_j$	PL_j	M_j	$P_{i,j}$ using eq. (3.14)	$P_{i,j}$ using eq. (3.15)
0	1	0	0	0	0
1	0	1	0	A_i	$A_i \oplus M_j = A_i$
1	0	1	0	A_i	$A_i \oplus M_j = A_i$
0	1	1	0	A_{i-1}	$A_{i-1} \oplus M_j = A_{i-1}$
0	1	0	1	$A^{'}_{i-1}$	$A_{i-1} \oplus M_j = A^{'}_{i-1}$
1	0	0	1	$A^{'}_{i}$	$A_i \oplus M_j = A^{'}_{i}$
1	0	0	1	$A^{'}_{i}$	$A_i \oplus M_j = A^{'}_{i}$
0	1	0	0	0	0

3.2.2.4 Addition of Fixed Number

The fixed number (-1) $(2^{M} - 1)/3$, should be added to the un-extended partial products, starting from the Nth binary position leftwards.

3.2.2.5 Accumulation of Partial Product Rows

Partial product rows are accumulated using fast CSA/ 3:2 compressor/ 4:2 compressor/ 2D/ 3D/ 4D to generate final rows of sum and carry [40], [41]. These final rows of sum and carry are added using CPA to generate final product for an M×N multiplier.

(A) 3:2 Compressor: This is essentially a 1-bit full adder. It accepts 3-bit input and generates two bit output as sum and carry.

(B) 4:2 Compressor: The first stage sum and carries are given to the 4:2 compressor as 4-bit input, which generates two bit output as sum and carry.

(C) 2D Block: This block generates two partial products using partial product generator and then these bits are compressed into sum and carry using 3:2 compressor.

(D) 3D Block: This block generates three partial products using partial product generator and then these bits are compressed into sum and carry using 3:2 compressor.

(E) 4D Block: this block generates four partial products using partial product generator and then these bits are compressed into sum and carry using 4:2 compressor.

3.2.2.6 Eliminating the Other Limitations of an Existing Algorithm

Another limitation of the basic algorithm is that it does not work for all the combinations of inputs. When both the inputs are zero it produces wrong results. Equation (3.12) suggests that MSB of multiplicand input A has to be '1' meaning it is always negative and multiplier input B is Booth encoded. Adding a 2's complement and swapping unit before Booth encoding removes these limitations, but it increases gate count and delay. The logic for pre-processing of inputs through 2's complement and/or swapping unit is presented below in table 3.6. The external inputs AI and BI are assigned to algorithm inputs A and B using swap select logic. Note that algorithm input A is always negative and B input is Booth encoded. This can be seen from equation 3.12 where MSB of A has to be '1'. If out of AI and BI one number is negative and the other number is positive then A is taken – ve & B is taken as positive. In such cases always the positive B operand is Booth encoded. If both numbers AI and BI are negative then A=AI and B=BI and B operand is Booth encoded. AI='0' is considered as +ve number if other number is negative. AI='0' is considered as -ve number if other number is positive. BI='0' is always considered as +ve irrespective of the other number. The swapping and 2's complement operations are explained in table 3.6.

Table 3.6 Logic for preprocessing of inputs

External Inputs	Swapping	2'S Complement	Algorithm Inputs

AI (+ve)
BI (+ve)
	NO	YES	A = -AI
			B = -BI

AI (+ve)
BI (-ve)
	YES	NO	A = BI
			B = AI

AI (-ve)
BI (+ve)
	NO	NO	A = AI
			B = BI

AI (-ve)
BI (-ve)
	NO	NO	A = AI
			B = BI

AI (zero) / AI (+ve)
BI (zero)
	NO	YES	A = -AI
			B = -BI

AI (-ve)
BI (zero)
	NO	NO	A = AI
			B = BI

AI (zero)
BI (-ve)
	YES	NO	A =BI
			B = AI

AI (zero)
BI (+ve)
	NO	NO	A = AI
			B = BI

Figure 3.3 shows the Block diagram of Booth encoded Wallace tree multiplier, which eliminates all limitations of existing algorithm.

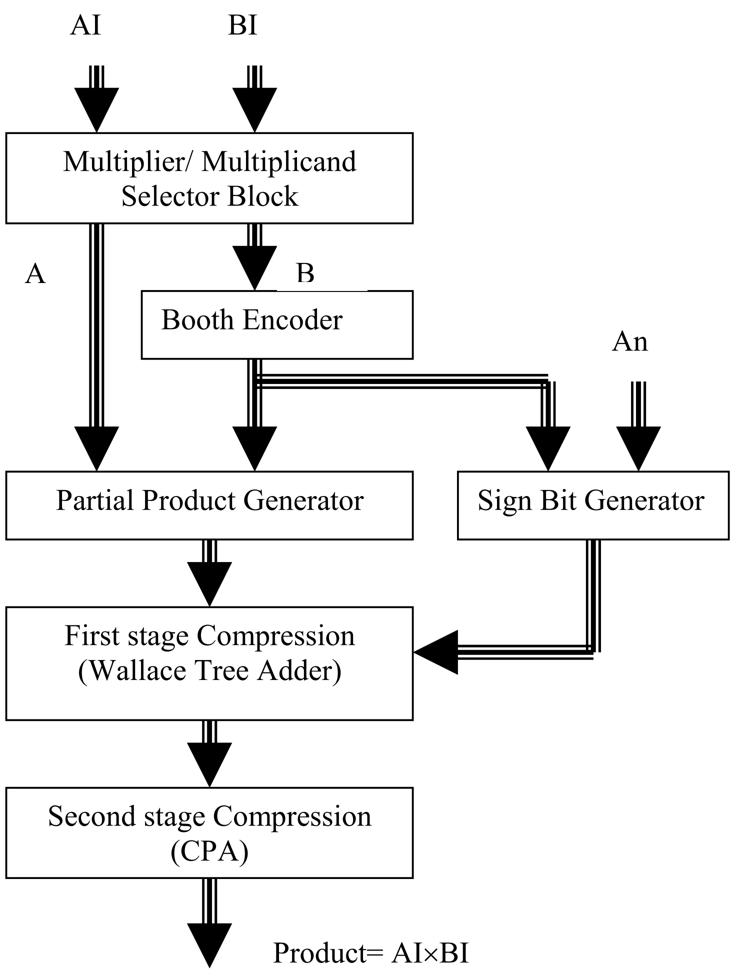

AI
BI
Multiplier/ Multiplicand Selector Block
A
B
Booth Encoder
An
Partial Product Generator
Sign Bit Generator
First stage Compression (Wallace Tree Adder)
Second stage Compression (CPA)
Product= AI×BI

Figure 3.3 Block diagram of Booth encoded Wallace tree multiplier eliminating limitations

Table 3.3 represents the 4×4 multiplier. The partial product bits in a particular column are generated and accumulated by either simple partial product generator/2D/3D/4D to give sum and carry bits as per equations 3.16-a. The selection of particular block depends on number of partial products to be generated and accumulated in a particular column (e.g. for generating and accumulating a single bit of partial product in a column requires simple partial product generator, while, if in a particular column 2 partial product bits are required to be generated and accumulated then 2D unit will be used). The sum and carry bits obtained by partial product generator/2D/3D/4D units are then added with sign bits and fixed number derived by using equation 3.10 to generate the final rows of sum and carry as per equations 3.16-b. These final rows of sum and carry are then added using a CPA to generate the product bits of the multiplier as per equations 3.16-b.

$$
\begin{aligned}
&S_0 = P_{0,0} \\
&S_1 = P_{1,0} \\
&C_2, S_2 = P_{2,0} + P_{0,2} \\
&C_3, S_3 = P_{3,0} + P_{1,2} \\
&S_4 = P_{2,2} \\
&S_5 = P_{3,2}
\end{aligned}
\qquad (3.16\text{-}a)
$$

Where, S_i and C_i are the bits after the 1st stage compression of the partial product bits produced by partial product generator/ 2D/3D/4D units. $1S_i$ & $1C_i$ are intermediate sum and carry bits generated after 2nd stage compression, which accumulate S_i & C_i sign bits and fixed number. $2C_i$ are carry bits and P_i are final product bits generated by the last stage of the CPA as shown by equation 3.16-b.

$$
\begin{aligned}
&1C_0, P_0 = S_0 + 0M_{GND} \\
&1C_1, 1S_1 = S_2 + 2M_{GND} \\
&1C_2, 1S_2 = S_3 + C_2 \\
&1C_3, 1S_3 = S_4 + 0M_{VDD} + C_3 + '1' \\
&1C_4, 1S_4 = S_5 + '1' \\
&1S_5 = 2M_{VDD} \\
&2C_0, P_1 = S_1 + 1C_0 \\
&2C_1, P_2 = 1S_1 + 2C_0 \\
&2C_2, P_3 = 1C_1 + 1S_2 + 2C_1 \\
&2C_3, P_4 = 1C_2 + 1S_3 + 2C_2 \\
&2C_4, P_5 = 1C_3 + 1S_4 + 2C_3 \\
&2C_5, P_6 = 1S_5 + 2C_4 \\
&P_7 = '1' + 2C_5
\end{aligned}
\qquad (3.16\text{-}b)
$$

Figure 3.4 shows a 4×4 Booth encoded Wallace tree multiplier designed using above described technique. Similar technique is followed in the design of 8×8, 12×12 and 16×16 multipliers.

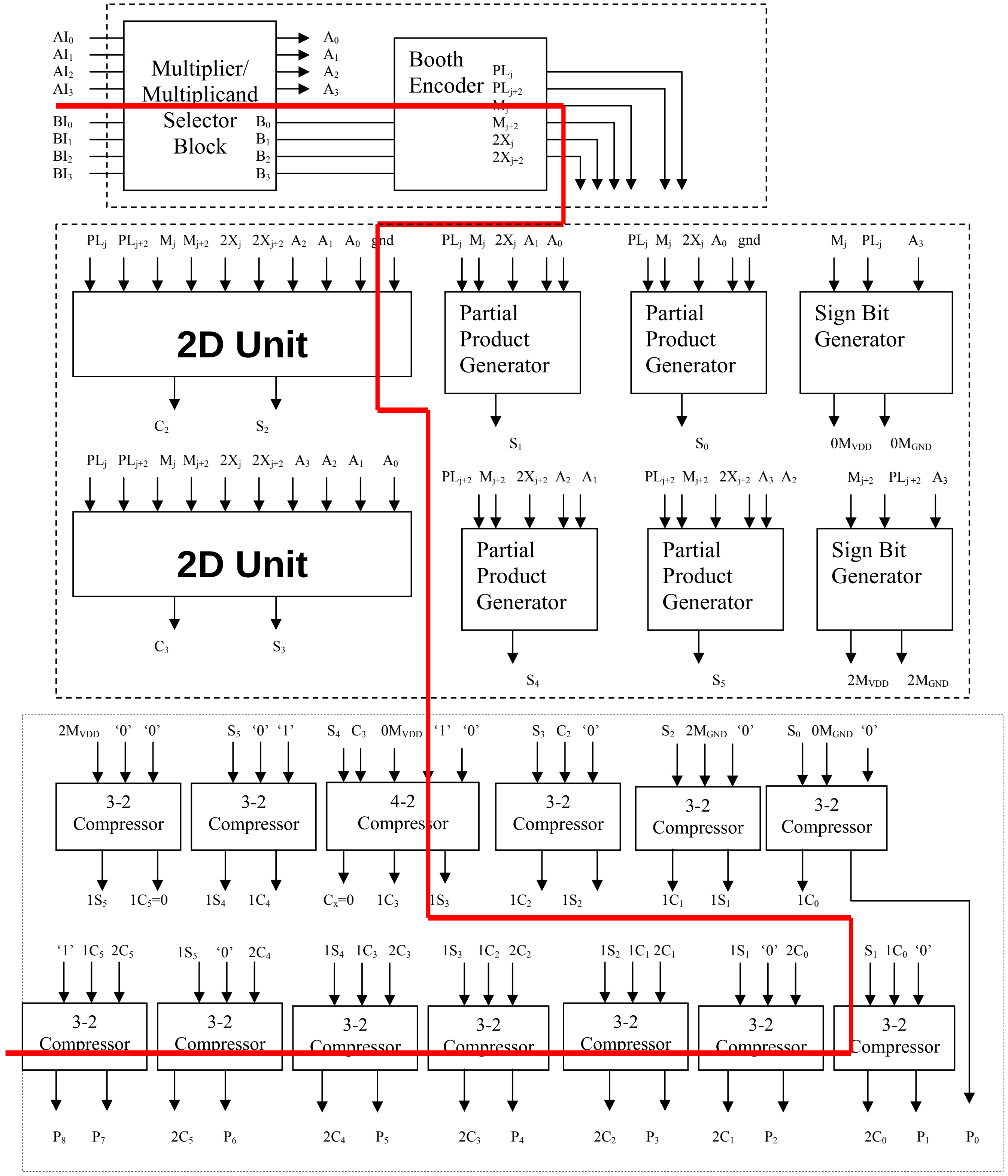

Multiplier/ Multiplicand Selector Block
Booth Encoder
2D Unit
2D Unit
Partial Product Generator
Partial Product Generator
Sign Bit Generator
Partial Product Generator
Partial Product Generator
Sign Bit Generator
3-2 Compressor
4-2 Compressor

Figure 3.4 A 4×4 Booth encoded Wallace tree multiplier-eliminating limitations (j='0')

3.3 MUX Based Multiplier

It is based on an **unsigned multiplier** algorithm in which one bit of the multiplier and one bit of the multiplicand are processed in parallel. The algorithm is symmetric, i.e. the multiplier and multiplicand can be interchanged. According to this algorithm, sum of the two operands, progressively computed, is a useful quantity that is used in the computation of certain partial products. The different quantities are computed one bit at each step of the algorithm and the appropriate quantity is then selected in the next step, if so required. The parallel implementation of this algorithm yields an iterative type array. Compared to implementation based on the modified Booth's algorithm, this algorithm requires similar amount of circuitry but yields faster multiplication [16]. This MUX based architecture performs parallel computation of the partial sums of the two operands together, which simplifies the tasks such as compression and accumulation. It also performs favorably well with regards to gate area, compared to other regular array architectures [16]. This architecture can also be extended to accept input in 2's complement form, with a little modification, but is not considered for present exploration.

3.3.1 Multiplication Logic

Equation 3.17, equation 3.18, equation 3.19, equation 3.20 and equation 3.21 explain the multiplication logic.

$$\begin{aligned} &A = a_{n-1}a_{n-2}\dots a_0 \\ &B = b_{n-1}b_{n-2}\dots b_0 \qquad (3.17) \\ &Let\ , \quad P = AB \end{aligned}$$

A_j & B_j are binary numbers after truncation, up-to the (j-1)th bit in A, B respectively as per equation 3.18.

$$\begin{aligned} &A_j = a_{j-1}a_{j-2}\dots a_0 = \sum_{n=0}^{j-1} a_n 2^n \\ &B_j = b_{j-1}b_{j-2}\dots b_0 = \sum_{n=0}^{j-1} b_n 2^n \qquad (3.18) \end{aligned}$$

$$A_j = a_{j-1} a_{j-2} \dots a_0$$
$$B_j = b_{j-1} b_{j-2} \dots b_0$$
$$P_j = A_j B_j \qquad for \qquad 0 < j < n \qquad (3.19)$$

$$A_0 = B_0 = 0 = P_0$$

$$\begin{aligned}
A &= A_n = A_{n-1} + 2^{n-1} a_{n-1} \\
\& \; B &= B_n = B_{n-1} + 2^{n-1} b_{n-1} \\
P_n &= A_n B_n = \left\{A_{n-1} + 2^{n-1} a_{n-1}\right\}\left\{B_{n-1} + 2^{n-1} b_{n-1}\right\} \\
&= 2^{2n-2} a_{n-1} b_{n-1} + 2^{n-1}(a_{n-1} B_{n-1} + A_{n-1} b_{n-1}) + A_{n-1} B_{n-1} \\
&= \sum_0^{n-1} a_j b_j 2^{2j} + \sum_0^{n-1} (a_j B_j + A_j b_j) 2^j + P_0 \\
&= \sum_0^{n-1} a_j b_j 2^{2j} + \sum_0^{n-1} (a_j B_j + A_j b_j) 2^j \\
&= \sum_0^{n-1} a_j b_j 2^{2j} + \sum_0^{n-1} Z_j 2^j \qquad (3.20)
\end{aligned}$$

Where ,

$$\begin{aligned}
& Z_j = a_j B_j + A_j b_j \\
& \Rightarrow Z_j = A_j \qquad if \quad a_j = 0, b_j = 1 \\
& \quad Z_j = B_j \qquad if \quad a_j = 1, b_j = 0 \\
& \quad Z_j = 0 \qquad if \quad a_j = 0, b_j = 0 \\
& \quad Z_j = A_j + B_j \qquad if \quad a_j = 1, b_j = 1 \qquad (3.21)
\end{aligned}$$

3.3.2 An Illustration of the Multiplication Logic

Example 3.2 shows the multiplication process for two 4-bit binary numbers using MUX based approach. The multiplication process shows that the number of rows remains the same, but number of partial product bits to be compressed in a particular column are now restricted to 3 bits only. This makes compression much faster and easier. If carry bits C_1,

C_2, C_3 … as shown by example 3.2 are taken care then the number of bits to be added in particular column will be 2 bits only. The two columns can be added simultaneously using 2-bit carry look ahead adder (CLA), which also accepts carry input C_1, C_2, C_3 of particular column (this is possible because, these carries are occurring in alternate columns). Thus the first step in algorithm is generation of partial product rows and second step performs the addition of these partial products together with compression.

Example 3.2(a): A_0B_0, A_1B_1, A_2B_2 & A_3B_3 at the positions shown below has be added with appropriate term selected by 4:1 MUX based on select lines shown in first column.

Let A= $A_3A_2A_1A_0$ = 0111 = $(+7)_{10}$ and B= $B_3B_2B_1B_0$ = 0011 = $(+3)_{10}$
The uncolored portion explains the operation to be performed by algorithm and colored portion show the application of algorithm on selected inputs A and B.
Working of MUX: Select lines '00'/'01'/'10'/'11' corresponds to $I_1/I_2/I_3/I_4$.

								Select line for 4:1 MUX
	A_3B_3		A_2B_2		A_1B_1		A_0B_0	
	0		0		1		1	
					$0/0/0/C_1$ =0/0/0/1	$0/A_0/B_0/S_0$ =0/1/1/0		A_1B_1 ='11'
					1	0		
			$0/0/0/C_2$ =0/0/0/1	$0/A_1/B_1/S_1$ =0/1/1/1	$0/A_0/B_0/S_0$ =0/1/1/0			A_2B_2 ='10'
			0	1	1			
	$0/0/0/C_3$ =0/0/0/1	$0/A_2/B_2/S_2$ =0/1/0/0	$0/A_1/B_1/S_1$ =0/1/1/1	$0/A_0/B_0/S_0$ =0/1/1/0				A_3B_3 ='00'
	0	0	0	0				
0	0	0	1	0	1	0	1	$=(21)_{10}$
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	

Example 3.2 (b): A_0B_0, A_1B_1, A_2B_2 & A_3B_3 at the positions shown below has be added with appropriate term selected by 4:1 MUX based on select lines shown in first column.

Let A= $A_3\,A_2\,A_1\,A_0$ = 1111= $(+15)_{10}$ and B= $B_3\,B_2\,B_1\,B_0$ = 1111 = $(+15)_{10}$
The uncolored portion explains the operation to be performed by algorithm and colored portion show the application of algorithm on selected inputs A and B.
Working of MUX: Select lines '00'/'01'/'10'/'11' corresponds to $I_1/I_2/I_3/I_4$.

								Select line for 4:1 MUX
	A_3B_3		A_2B_2		A_1B_1		A_0B_0	
	1		1		1		1	
					$0/0/0/C_1$ =0/0/0/1	$0/A_0/B_0/S_0$ =0/1/1/0		A_1B_1 ='11'
					1	0		
			$0/0/0/C_2$ =0/0/0/1	$0/A_1/B_1/S_1$ =0/1/1/1	$0/A_0/B_0/S_0$ =0/1/1/0			A_2B_2 ='11'
			1	1	0			
	$0/0/0/C_3$ =0/0/0/1	$0/A_2/B_2/S_2$ =0/1/1/1	$0/A_1/B_1/S_1$ =0/1/1/1	$0/A_0/B_0/S_0$ =0/1/1/0				A_3B_3 ='11'
	1	1	1	0				
1	1	1	0	0	0	0	1	$=(225)_{10}$
P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	

It produces output in time T = (n+1) τ_{FA_2CLA} where τ_{FA_2CLA} is the delay of a 2-bit CLA with a timing overhead of one 4:1 MUX delay, while regular array multiplier takes approximate delay of T = (2n) τ_{FA} as seen in figure 3.1. The large area overhead will be due to routing needed between these MUXs.

3.3.3 Implementation

The logic explained in example 3.2 can be shown through a schematic, which uses 4:1 MUXs & AND gates as shown in figure 3.5(a). The MUXs are used to choose the Z_j for the $\mathbf{Z_j\ 2^j}$ terms (refer to equation 3.21) while AND gates are used to produce the $\mathbf{a_j b_j 2^{2j}}$ terms. Complete implementation of the multiplier is shown in figure 3.5 (b). Cell I and cell II used in MUX based multiplier implementation are shown in figure 3.5 (c) and figure 3.5 (d) respectively. The number of cell I required in a n×n multiplier are n(n-1)/2 while number of cell II required are n.

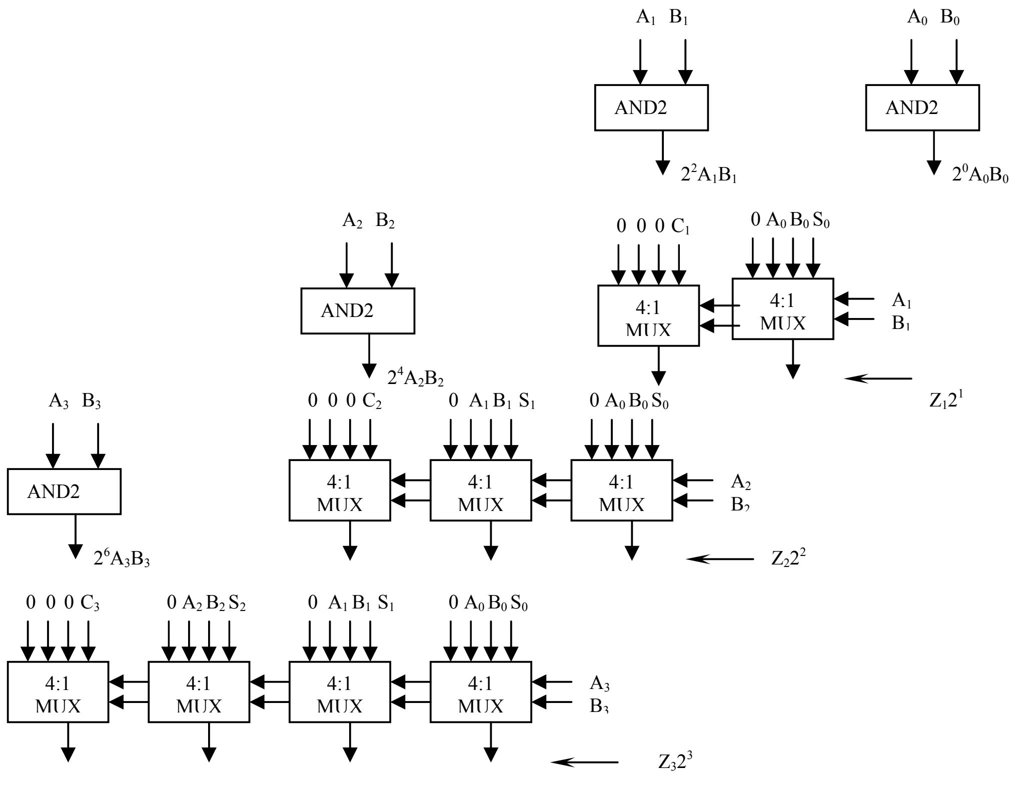


Figure 3.5 (a) MUX based multiplier implementation logic

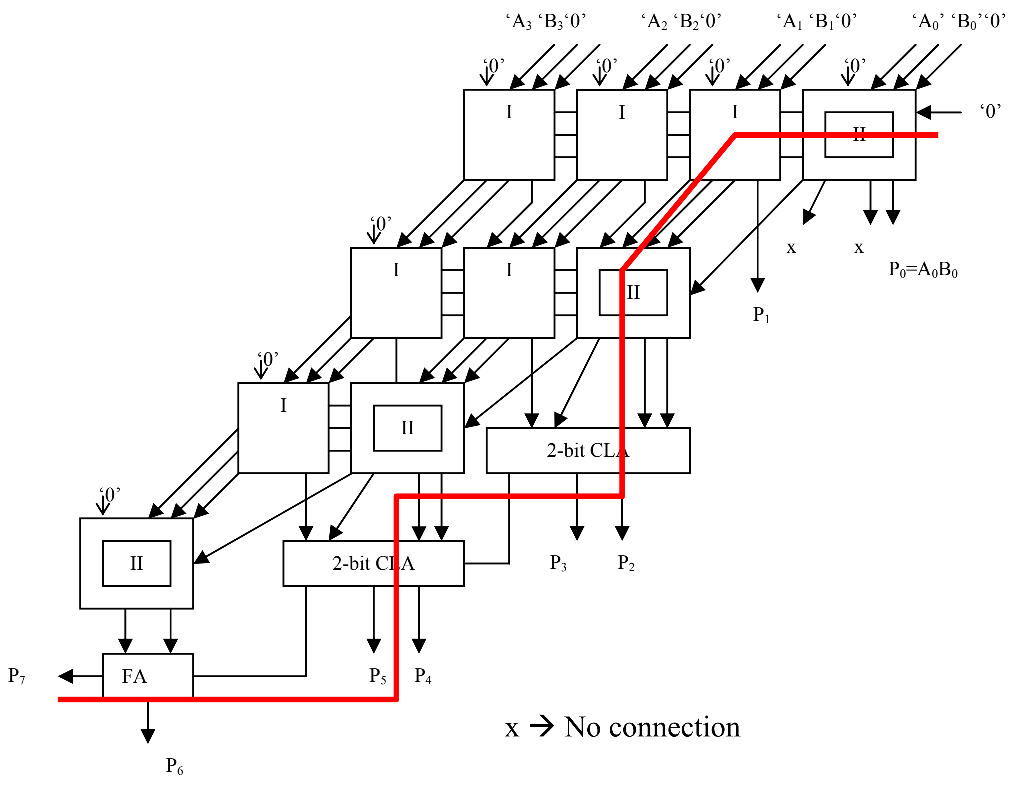


Figure 3.5 (b) MUX based multiplier implementation

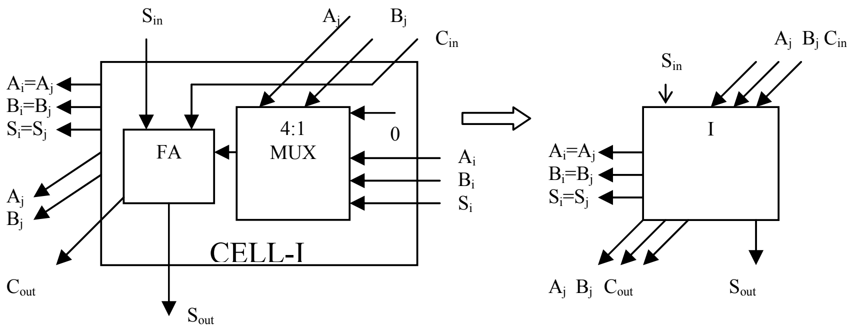


Figure 3.5 (c) Cell I used in MUX based multiplier implementation

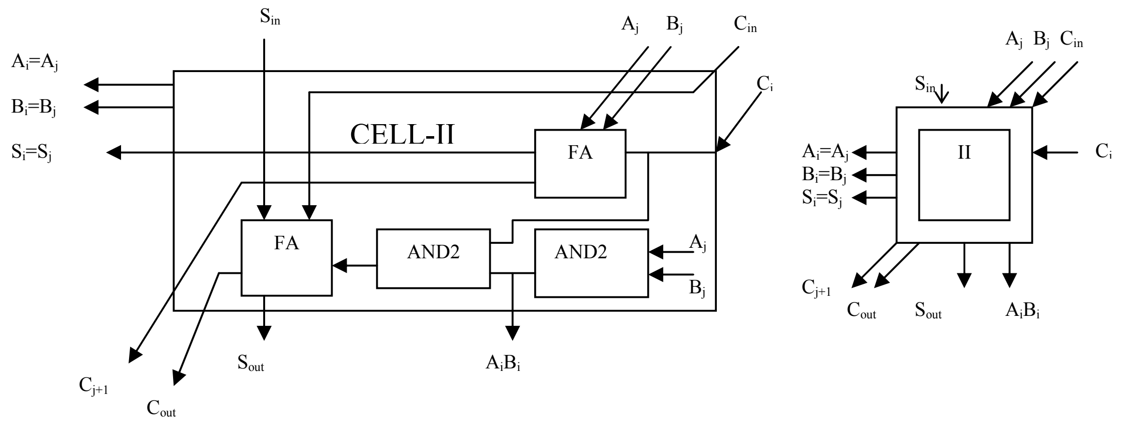


Figure 3.5 (d) Cell II used in MUX based multiplier implementation

3.4 2×2 Cell Based Multiplier

In this architecture the 2×2 multiplier is used as a basic building block in the hierarchical design of a larger bit-size multiplier. This multiplier uses a hybrid scheme called the array of array multiplier, which has moderate routing area requirements and time complexity of $O(\sqrt{n})$ for an n×n multiplier [10]. The truth table for a 2×2 combinational multiplier is shown in table 3.7. The truth table can be solved using Karnaugh maps, which generates the equation (3.22) as shown below. A combinational circuit can be realized using these equations. Figure 3.6 shows the schematic of a 2×2 combinational multiplier.

First step in the design of a 4-bit multiplier will be to find the different combinations of input bit pairs that are to be processed by 2×2 multipliers. Each input bit pair is handled by a separate 2×2 combinational multiplier to produce a partial product row. These partial products rows are then added optimally to generate the final product bits. The design procedure for 4×4 combinational multiplier is shown in table 3.8. Figure 3.7 shows the schematic of a 4×4 combinational multiplier designed using 2×2 combinational multiplier.

Table 3.7 Truth table for a 2×2 combinational multiplier

A_1	A_0	B_1	B_0	P_3	P_2	P_1	P_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

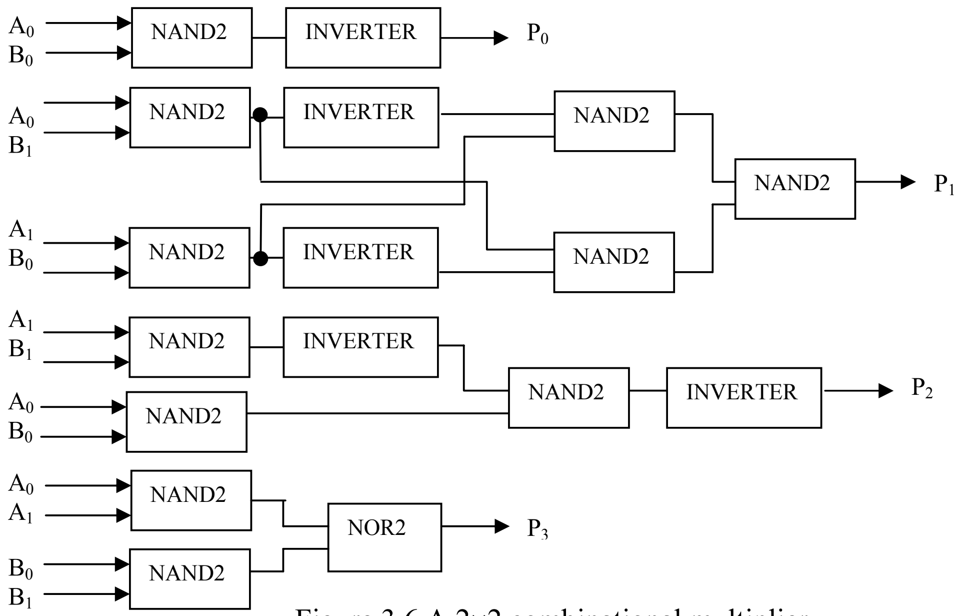


Figure 3.6 A 2×2 combinational multiplier

$$P_0 = A_0 \cdot B_0$$

$$\begin{aligned}
P_1 &= A_0 B_1(B'_0 + A'_1) + A_1 B_0(B'_1 + A'_0) \\
&= \{[A_0 B_1(B'_0 + A'_1) + A_1 B_0(B'_1 + A'_0)]'\}' \\
&= \{[A_0 B_1(B'_0 + A'_1)]' \cdot [A_1 B_0(B'_1 + A'_0)]'\}' \\
&= \{[(A_0B_1)' + (B_0 A_1)] \cdot [(A_1 B_0)' + (A_0B_1)]\}' \\
&= [(A_0B_1)' + (B_0 A_1)]' + [(A_1B_0)' + (A_0B_1)]' \\
&= [A_0B_1 (B_0A_1)'] + [(A_1 B_0)(A_0B_1)'] \\
&= \{[A_0B_1 (B_0A_1)']' \cdot [(A_1 B_0)(A_0B_1)']'\}'
\end{aligned}$$

$$\begin{aligned}
P_2 &= A_1B_1 (A'_0 + B'_0) \\
&= \{[A_1B_1 (A'_0 + B'_0)]'\}' \\
&= [(A_1 B_1)' + (A_0 B_0)]' \\
&= (A_1 B_1) \cdot (A_0 B_0)'
\end{aligned}$$

$$\begin{aligned}
P_3 &= A_1 A_0 B_1 B_0 \\
&= \{[A_1A_0 B_1 B_0]'\}' \\
&= [(A_1A_0)' + (B_1B_0)']' \qquad (3.22)
\end{aligned}$$

Table 3.8 Design of a 4-bit multiplier using 2×2 combinational multiplier

Pair					$\underline{A_3}$	$\underline{A_2}$	$\underline{A_1}$	$\underline{A_0}$
					Group II		Group I	
					$\underline{B_3}$	$\underline{B_2}$	$\underline{B_1}$	$\underline{B_0}$
					Group IV		Group III	
I × III					PP_3	PP_2	PP_1	PP_0
II ×III			PP_7	PP_6	PP_5	PP_4		
I × IV			PP_{11}	PP_{10}	PP_9	PP_8		
II × IV	PP_{15}	PP_{14}	PP_{13}	PP_{12}				
Sum	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0

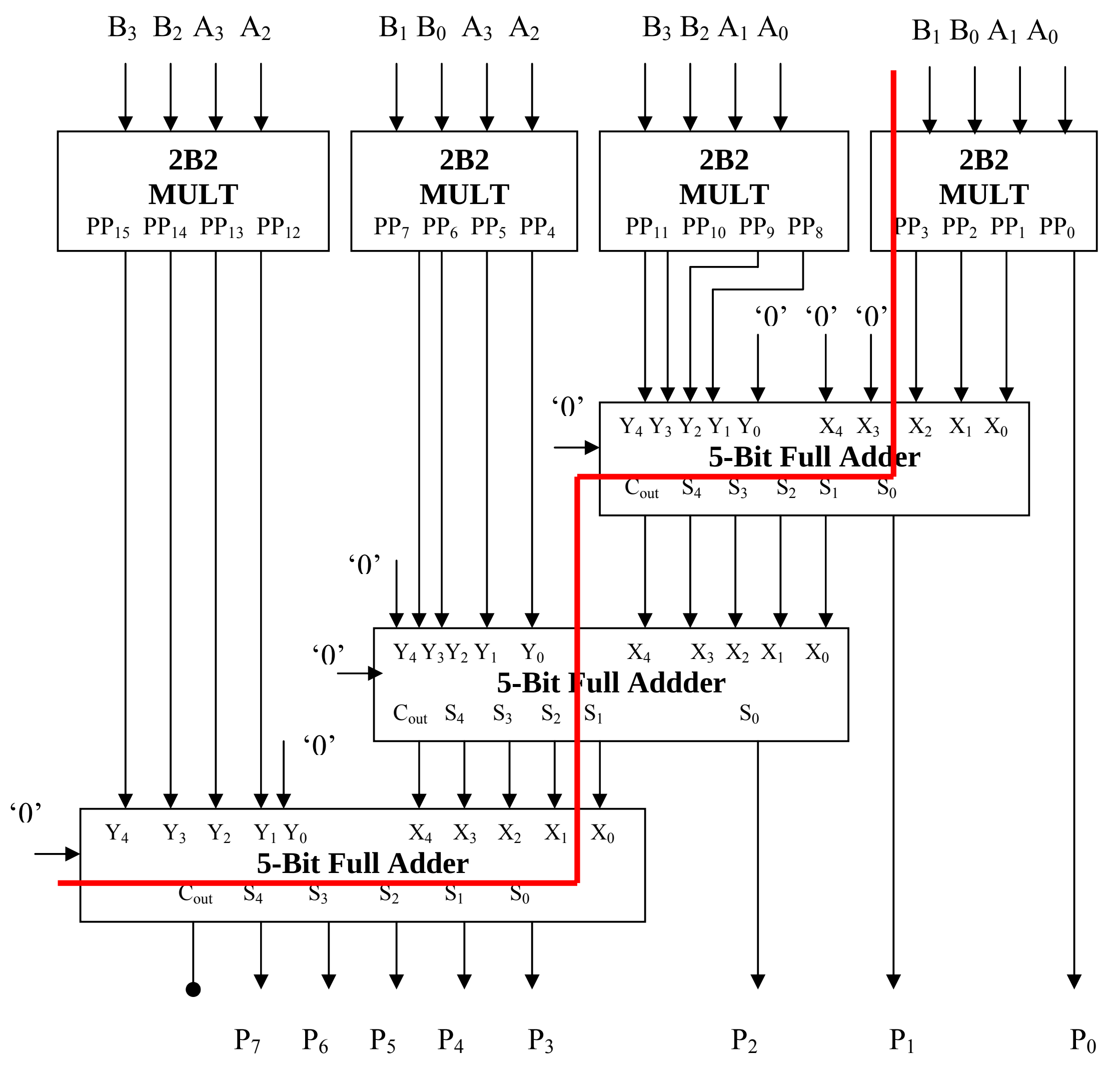


Figure 3.7 A 4×4 combinational multiplier

Similar technique is used in the design of 8×8, 12×12 and 16×16 multipliers. In such design at each level of hierarchy in schematic design only four partial product rows are required to be generated. Hence the accumulation of these partial product rows is much simplified as compared to other architectures

3.5 MUX Based Barrel Shifter

The MUX based barrel shifter architecture has been designed using 4:1, 8:1, 16:1, 32:1 and 64:1 MUXs. The design follows a hierarchy, which can be described as follows. The 2:1 MUX is first designed using CMOS logic. It is then used to design the 4:1 MUX. The 8:1 MUX is designed using the 4:1 MUXs as the basic building block. Similar hierarchy is followed in the design of 16:1, 32:1 and 64:1 MUXs. A 2^n:1 MUX will have 2^n input lines, n select lines and one output line. MUXs with any n, greater than one can be implemented using 2:1 MUX. One 2^n: 1 MUX requires (2^n-1) 2:1 MUX s and has a delay of n 2:1 MUX. In this architecture n =1, 2, 3, 4, 5, 6 are used in the design 2:1, 4:1, 8:1, 16:1, 32:1 and 64:1 MUXs.

3.5.1 Design of 4-bit MUX Based Barrel Shifter

Table 3.9 shows the behavior of 4-bit MUX-based barrel shifter, which utilizes control inputs D for direction, S/R for operation (shift/rotate) and S_1, S_0 for number of bits to be shifted or rotated. O_0, O_1, O_2, O_3 represent the output bits and I_0, I_1, I_2, I_3 are input bits. D='0' means the direction of shift/rotate operation is towards left and D='1' means the direction of shift/rotate operation is towards right. F represents the fill bit, fill bit is '0' for left shift and fill bit is MSB bit for right shift operation. The line S/R='1' for rotate operation and S/R= '0' for shift operation. Bits S_1 and S_0 are length selection bits. S_1S_0='00' means length is zero bit, S_1S_0= '01' means length is one bit, S_1S_0= '10' means length is two bits and S_1S_0= '11' means length is three bits. Table 3.9 explains the various operations performed by 4-bit barrel shifter.

As there are four control inputs, we need a 16:1 MUX for each output bit. Thus for 4 output bits, we need four 16:1 MUXs in the design of a 4-bit barrel shifter. In addition we need a 2:1 MUX for fill bit as shown in figure 3.9. Thus the total number of 2:1 MUXs required in the design are 61.

Table 3.9 Truth Table for 4-bit barrel shifter operation

Operation	D	S/R	S_1	S_0	O_3	O_2	O_1	O_0
Arithmetic shift left	0	0	0	0	I_3	I_2	I_1	I_0
	0	0	0	1	I_2	I_1	I_0	F
	0	0	1	0	I_1	I_0	F	F
	0	0	1	1	I_0	F	F	F
Rotate left	0	1	0	0	I_3	I_2	I_1	I_0
	0	1	0	1	I_2	I_1	I_0	I_3
	0	1	1	0	I_1	I_0	I_3	I_2
	0	1	1	1	I_0	I_3	I_2	I_1
Arithmetic shift right	1	0	0	0	I_3	I_2	I_1	I_0
	1	0	0	1	F	I_3	I_2	I_1
	1	0	1	0	F	F	I_3	I_2
	1	0	1	1	F	F	F	I_3
Rotate right	1	1	0	0	I_3	I_2	I_1	I_0
	1	1	0	1	I_0	I_3	I_2	I_1
	1	1	1	0	I_1	I_0	I_3	I_2
	1	1	1	1	I_2	I_1	I_0	I_3

Each row of the truth table 3.9 can be implemented with a dedicated 16:1 multiplexer circuit, which is designed using 2:1 MUX cells, to obtain the final output. Similar technique is followed in the design of 8-bit, 12-bit and 16-bit barrel shifters.

3.5.2 Fill Bit Logic

Fill bits are required only in the case of shift operations. In left shift operation, the lower significant bits are filled with 0's. While in the right shift operation, in order to preserve the sign of the input number, MSB is sign extended. This is accomplished using a 2:1 MUX. The D='0' represents the left operation for which fill bit is '0' and D='1' means the right operation for which fill bit is MSB bit of input. The Fill bit is utilized for shift operations, while it is discarded for rotate operations. Figure 3.8 shows the schematic of a 4-bit MUX-based barrel shifter, which also includes fill bit logic.

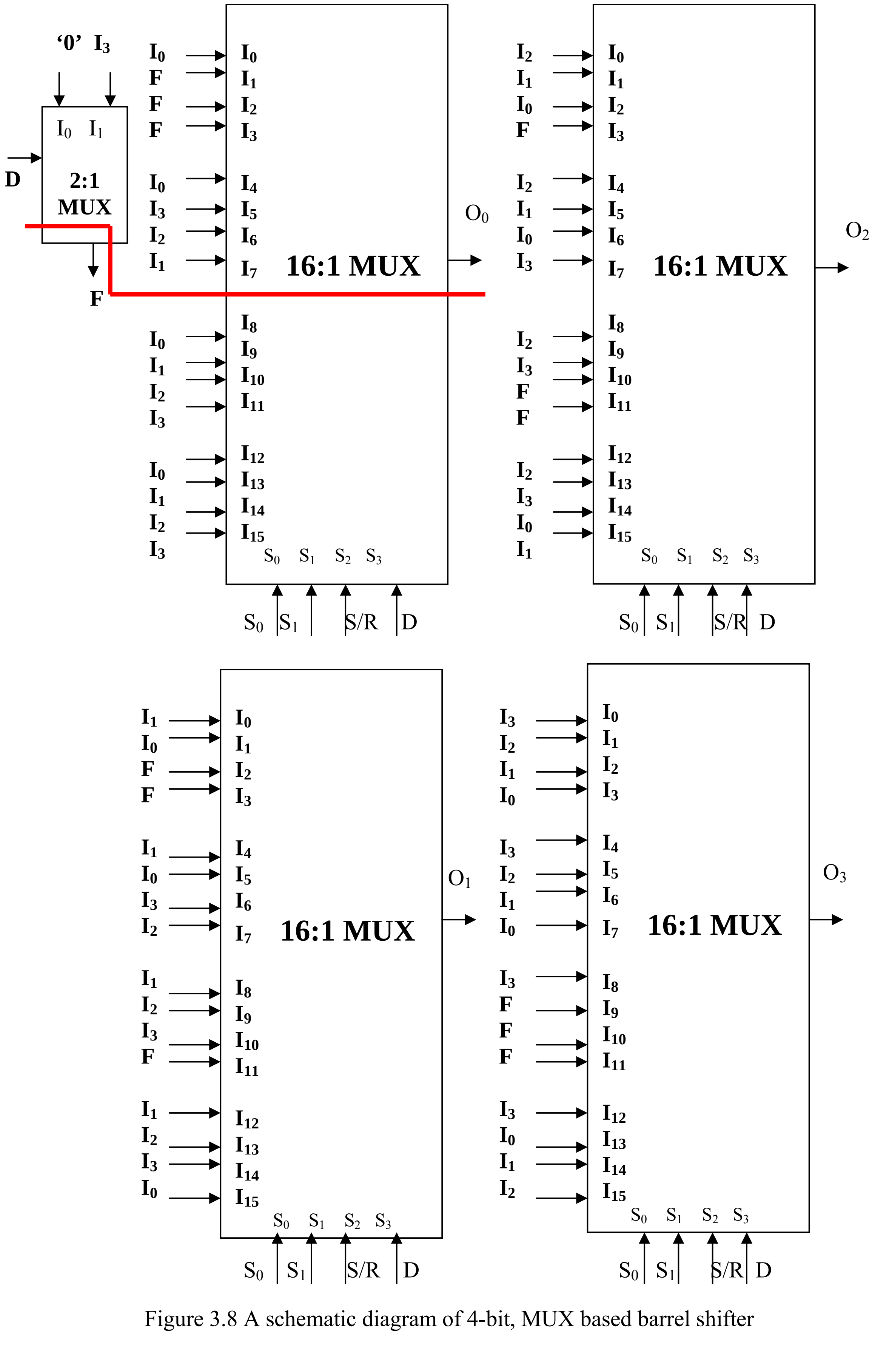


Figure 3.8 A schematic diagram of 4-bit, MUX based barrel shifter

3.6 Pereira's Barrel Shifter

In this architecture significant reduction in area occupied by the barrel shifter circuit is achieved by implementing rightward operations as operations in leftward direction and significant reduction in delay is possible by reducing the length of the critical path. The 16-bit barrel shifter is chosen for explanation. The 16-bit barrel shifter design has six external control signals out of which the first four bits represent the length of operation, while the fifth and sixth bits represent the 'direction' and the 'type' of the operation respectively. Length bits $LGTH_x$ [x=0, 1, 2, 3], allow shifting or rotation of data from 0 [i.e $LGTH=LGTH_3\ LGTH_2\ LGTH_1\ LGTH_0$='0000'] to 15 positions [i.e $LGTH=LGTH_3\ LGTH_2\ LGTH_1\ LGTH_0$='1111']. DIR bit decides direction of shift/rotation (DIR='0' means leftward and DIR='1' means rightward direction) and TYP bit decides shifting or rotation (TYP='0' means shifting and TYP='1' means rotation). This barrel shifter circuit consists of shift rotate array, programming unit and mask generator unit. Section 3.6.1 explains the shift-rotate array, section 3.6.2 describes the programming unit and section 3.6.3 describes the mask generator unit. Schematic and operation are described in section 3.6.4

3.6.1 Shift-Rotate Array

This unit consists of six stages. The first stage will allow leftward rotation of data by 0 or 1 position. This stage is used for creating an *additional* left rotation by 1-bit, required for rightward operations. Next four stages provide shifting or rotation of data from 0 to 15 positions to the left, using combination of second stage (0 or 1 position), third stage (0 or 2 positions), fourth stage (0 or 4 positions) and fifth stage (0 or 8 positions). The sixth stage performs the masking for right shift operations only.

Rotating 16-bit ($N=2^4$) data p positions to the right where $0 \leq p \leq N-1$ is equivalent to rotating data (N-p) positions to the left. Similarly, shifting p position to the right where $0 \leq p \leq N-1$ is equivalent to rotating data (N-p) positions to the left and masking p most significant bits with the sign bit of the input data for arithmetic shifting. Attaining the (N-p) positions leftward operation is achieved by first rotating by 1-bit towards left and then rotating leftward again by 1's complement of length LGTH. Figure 3.9 shows the schematic representation of a shift rotate array. First stage of the circuit performs the additional left rotation of data by 0 or 1 position and is controlled by R and R' signals, which are derived from DIR. Stage 2 to stage 5 perform the task of left rotation/left shift and are controlled by LS_x, LS'_x and LR_x which are generated by programming unit using equation (3.27), equation (3.28) and equation (3.29). Note that x=0 (for stage 2), x=1 (for stage 3), x=2 (for stage 4) and x=3 (for stage 5). For left shift operation fill bits are '0' which are appropriately created by stage 2 to stage 5. Since for right shift operation fill bits are sign extension bits, we require an extra masking stage labeled as stage 6, which provides sign extensions of MSB bit for appropriate bit length and is controlled by signal M_i (i=1…..15) and M'_i (i=1…..15) which are generated by the masking unit (as described in section 3.6.3).

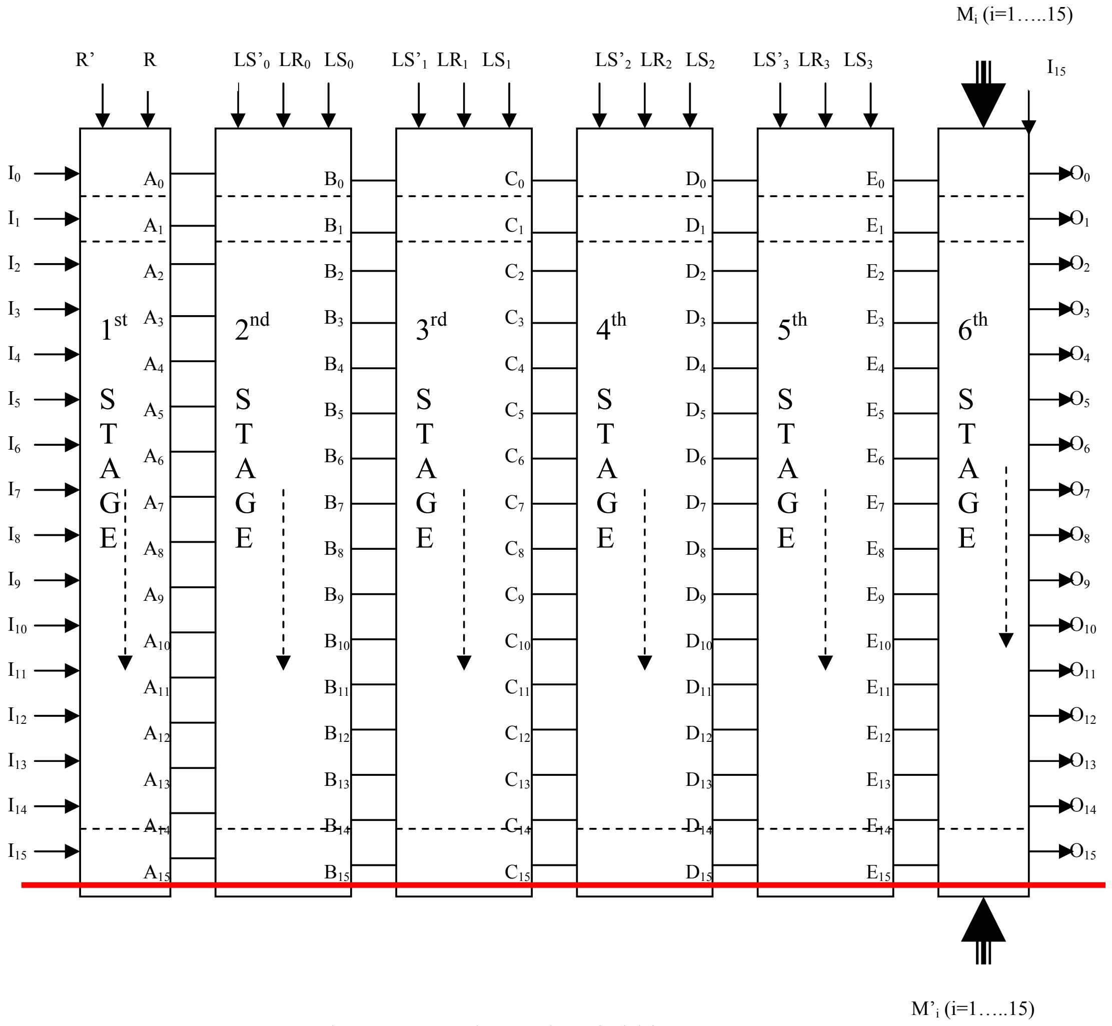


Figure 3.9 Schematic of shift-rotate array

Each stage of the shift rotate array circuit consists of one instance of AND-OR function per bit (i.e. for 16 bits we need 16 interconnected AND-OR functions). The AND-OR function implements the logic function given in equation (3.23). Operations of AND-OR function are explained in table 3.10.

$$Y_i=C_1.\ I_i+C_2.\ I_j. \tag{3.23}$$

Where C_1, and, C_2 are internal control bits applied to individual instances of AND-OR function that are derived from top level control signals. Also 'I_i' is input bit, and, 'I_j' is left rotated bit which are appropriately derived from input data bits to each stage (as described below).

Table 3.10 Operations performed by AND-OR function

C_1	C_2	Output (Y_i)	Comments
0	0	Fill bit '0' is passed	Required for left shift operations for stage 2 to stage 5
0	1	Left rotated bit (I_j) is passed	Required for passing left rotated data for stage 1 to stage 5
1	0	Input bit (I_i) is passed	Required for passing data without any change for stage 1 to stage 6
1	1	I_i+I_j	Never comes

The 1st stage of shift rotate array has C_1=R', and, C_2=R for all 16 instances of AND-OR. Note that R'='0', and, R='1' for performing 1-bit additional left rotate operation required for rightward operations. R'='1', and, R='0' for passing data without any operation (as required in leftward operations). The operation of 1st stage is explained in equation (3.24).

i.e.

For R'='1', and, R='0':

$$A_i=I_i \qquad \text{(For } i=0.....15)$$

For R'='0', and, R='1':

$$A_i=I_{15} \qquad \text{(For } i=0)$$

$$A_i=I_{i-1} \qquad \text{(For } i=1.....15) \tag{3.24}$$

The 2nd stage of the circuit has C_1= LS'_0 and C_2= LR_0 for one instance of AND-OR corresponding to one LSB fill bits. The remaining 15 instances of AND-OR have C_1= LS'_0 and C_2=LS_0. For passing the data without any modification LS'_0='1', LR_0='0', and, LS_0='0'. For left rotation LS'_0='0', LR_0='1', and, LS_0='1', but for left shift operation LS'_0='0', LR_0='0', and, LS_0='1' [since LS'_0='0', and, LR_0='0' the fill bit instance of AND-OR will generate output '0' according to equation (3.23)]. The operation of 2nd stage is explained in equation (3.25).

i.e.

For LS'_0='1', LR_0='0', and, LS_0='0':

$B_i=A_i$ (For i=0…..15)

For LS'_0='0', LR_0='1', and, LS_0='1':

$B_i=A_{15}$ (For i=0)

$B_i=A_{i-1}$ (For i=1…..15)

For LS'_0='0', LR_0='0', and, LS_0='1':

$B_i=0$ (For i=0)

$B_i=A_{i-1}$ (For i=1…..15) (3.25)

Operations performed by 3rd, 4th & 5th stages can be explained in similar manner as shown in table 3.11.

Table 3.11 Operations performed by 3rd, 4th and 5th stages

Stage #3	Stage #4	Stage #5
For $LS'_1=1, LR_1=0$, and $LS_1=0$:	For $LS'_2=1, LR_2=0$, and $LS_2=0$:	For $LS'_3=1, LR_3=0$, and $LS_3=0$:
$C_i=B_i$ (For i=0…..15)	$D_i=C_i$ (For i=0…..15)	$E_i=D_i$ (For i=0…..15)
For $LS'_1=0, LR_1=1$, and $LS_1=1$:	For $LS'_2=0, LR_2=1$, and $LS_2=1$:	For $LS'_3=0, LR_3=1$, and $LS_3=1$:
$C_i=B_{14+i}$ (For i=0,1)	$D_i=C_{12+i}$ (For i=0….3)	$E_i=D_{8+i}$ (For i=0…7)
$C_i=B_{i-2}$ (For i=2…..15)	$D_i=C_{i-4}$ (For i=4…..15)	$E_i=D_{i-8}$ (For i=8…..15)
For $LS'_1=0, LR_1=0$, and $LS_1=1$:	For $LS'_2=0, LR_2=0$, and $LS_2=1$:	For $LS'_3=0, LR_3=0$, and $LS_3=1$:
$C_i=0$ (For i=0,1)	$D_i=0$ (For i=0,….3)	$E_i=0$ (For i=0…7)
$C_i=B_{i-2}$ (For i=2…..15)	$D_i=C_{i-4}$ (For i=4…..15)	$E_i=D_{i-8}$ (For i=8…..15)

The 6th stage of the shift rotate array has $C_1=M'_i$ and $C_2= M_i$ for 15 instances of AND-OR corresponding to 15 MSB bits. Since maximum possible length for right shift is 15 bits, no masking is required for 0th bit. Thus 0th bit can be passed directly. Generation of these masking bits is explained in section 3.6.3. Operation of the 6th stage is explained in equation (3.26).

For M_i ='0' and M'_i='1':

$O_0=E_0$

$O_i=E_i$ (For i=1…..15)

For M_i ='1' and M'_i='0':

$$O_0=E_0$$

$$O_i=MSB=I_{15} \qquad \text{(For i=1.....15)} \qquad (3.26)$$

3.6.2 Programming Unit

Programming unit controls the operation of the five stages of the shift-rotate array. A schematic circuit diagram of the programming unit is shown in figure 3.10. Input signals to programming units are TYP, DIR and $LGTH_x$ [x=0, 1, 2, 3]. The unit generates outputs as LS_x, LS'_x and LR_x [x=0,……3] using equations (3.27), (3.28) and (3.29):

$$LS_x = [LGTH_x \oplus DIR] \qquad \text{where x= 0,1,2,3} \qquad (3.27)$$

$$LS'_x = \text{not } [LGTH_x \oplus DIR] \qquad \text{where x=0,1,2,3} \qquad (3.28)$$

$$LR_x = (DIR + TYP)\,(LGTH_x \oplus DIR) \qquad \text{where x= 0,1,2,3} \qquad (3.29)$$

3.6.3 Mask Generator Unit

Mask generator unit controls masking in the last stage of the shift-rotate array. This module has input signals as TYP, DIR, $LGTH_x$ [x=0, 1, 2, 3] and generates outputs M_i [i =0,……15] and M'_i [i =0,……15] using equation (3.30) & (3.31) respectively. For right shift operation, masking bits are sign extension bits and for left shift they are '0'. Sixth stage of shift-rotate array along with masking unit provides sign extensions of MSB bit for desired length.

$$IM_x = [DIR.TYP'].\,LGTH_x, \qquad \text{where x= 0,1,2,3}$$

$$M_1 = (DIR\ TYP'\ LGTH_3).\,(DIR\ TYP'\ LGTH_2).\,(DIR\ TYP'\ LGTH_1).$$

$$(DIR\ TYP'\ LGTH_0)$$

$$= IM_3.IM_2.IM_1.IM_0 \qquad \text{For } LGTH=(1111)_2=(15)_{10}$$

M_2= $IM_3.IM_2. IM_1.IM'_0$+ M_1 For LGTH=$(1110)_2$=$(14)_{10}$

M_3= $IM_3.IM_2. IM'_1.IM_0$+ M_2 For LGTH=(1101)2=$(13)_{10}$

M_{15}= $IM'_3.IM'_2. IM'_1.IM_0$+ M_{14} For LGTH=(0001)= $(01)_{10}$

(3.30)

& M'_i = not (M_i) (For i=1…..15) (3.31)

3.6.4 Schematic and Operation

Shift-Rotate array, programming unit and mask generator unit are combined to form a 16-bit barrel shifter. Figure 3.10 shows the schematic diagram of the complete 16-bit barrel shifter unit. Control inputs for performing different shift and rotate operations are listed in table 3.12.

Table 3.12 Operation performed by 16-bit barrel shifter

Operation	DIR	TYP	LGTH			
			$LGTH_3$	$LGTH_2$	$LGTH_1$	$LGTH_0$
Arithmetic shift left	0	0	p_3	p_2	p_1	p_0
Rotate left	0	1	p_3	p_2	p_1	p_0
Arithmetic shift right	1	0	p_3	p_2	p_1	p_0
Rotate right	1	1	p_3	p_2	p_1	p_0

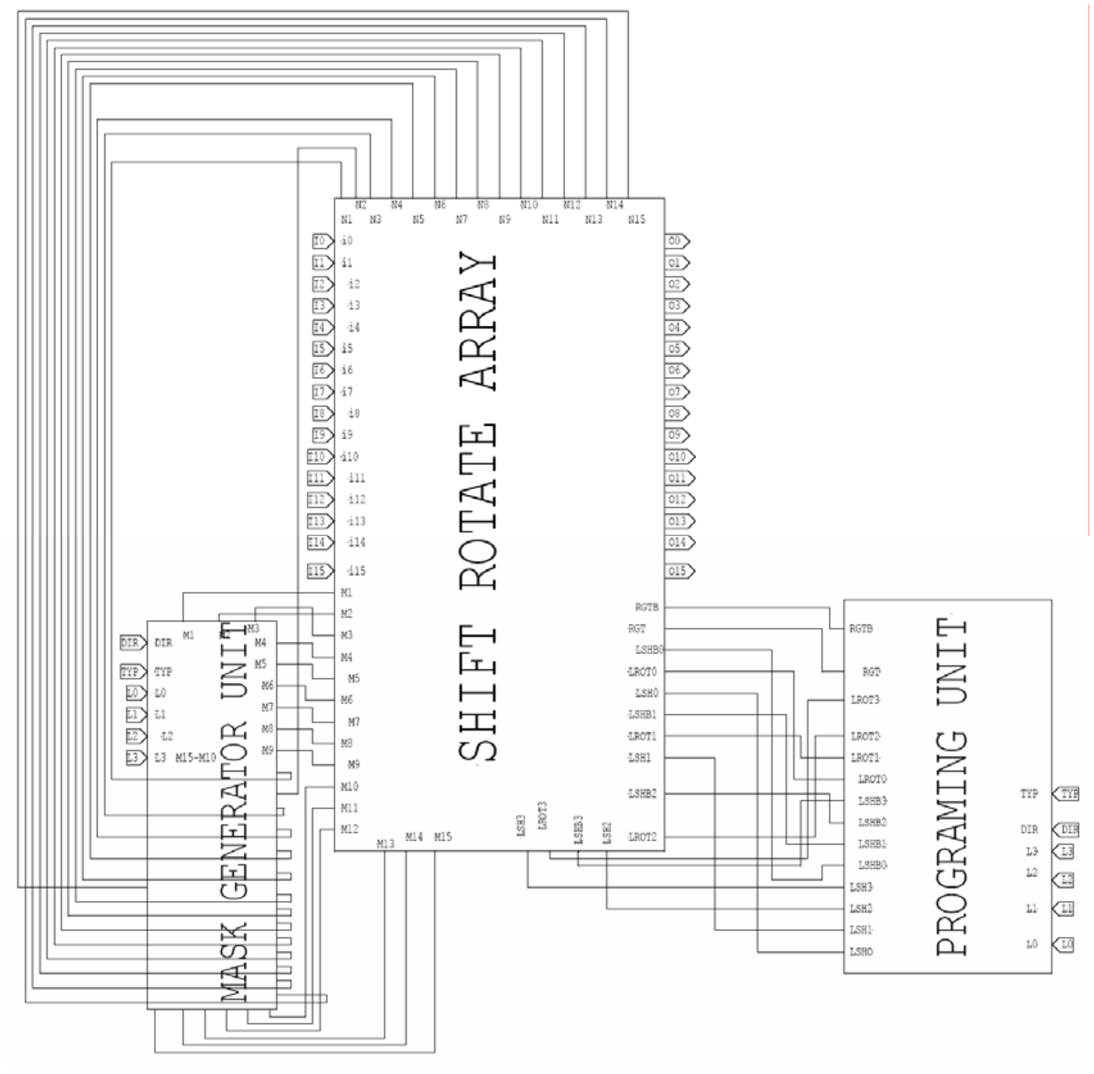


Figure 3.10 Complete barrel shifter unit

3.7 Chapter Summary

In this chapter we present the detailed schematic level design and associated Boolean logic for the selected multiplier and barrel shifter architectures. The selected signed multipliers are Baugh Wooley multiplier and Booth encoded Wallace tree multiplier, while unsigned multipliers are MUX based multiplier and 2×2 cell based multiplier. The signed multipliers follow the 2's complement number representation. The selected barrel shifter architectures are MUX based barrel shifter and Pereira's barrel shifter. All the multipliers as well as barrel-shifter architectures are implemented for 4-bit, 8-bit, 12-bit and 16-bit sizes using four different logic design styles. The details of logic design styles used for VLSI implementation of these architectures are presented in the next chapter.

CHAPTER 4

DIFFERENT CMOS LOGIC DESIGN STYLES AND DETRMINATION OF PMOS/NMOS WIDTH RATIO FOR HIGH SPEED DESIGN

The scalable CMOS technology offers many advantages like more transistors per chip and improvement in speed besides low power consumption, and large noise margins [1], [2], [3]. VLSI implementations of a circuit using different CMOS logic design styles show trade-offs in terms of parameters like transistor count, core layout area, average power and peak power consumption. The suitability of a particular logic design style for design is decided by performance, power and cost targets to be achieved. The multiplier and barrel shifter architectures discussed in chapter 3 are designed for 4-bit, 8-bit, 12-bit and 16-bit sizes using four different CMOS logic design styles in our explorations. In this chapter we discuss the fundamentals of the logic design styles considered for exploration, these logic design styles are categorized as described below.

- ❖ Static CMOS logic circuits
  - Static logic design style
  - TG logic design style
- ❖ Dynamic CMOS logic circuits
  - Dual rail domino logic design style
  - TSPC logic design style

4.1 Static CMOS Logic Circuits [1], [2], [3]

Static logic circuits allow versatile implementations of logic functions based on static or steady state behavior of simple CMOS structures. Typically a static logic gate generates its output corresponding to the applied input voltages after a certain time delay, and it can preserve its output level (or state) as long as the power supply is provided. This approach, however may require a large number of transistors to implement a function, and may cause a considerable time delay. In steady state each gate output is connected to either V_{DD} or G_{ND} through a low-resistive path and therefore for a static input, the output levels are preserved. On the other hand dynamic logic circuits for their operation rely on temporary storage of logic signal values on the capacitances of circuit nodes.

4.1.1 Static Logic Design Style

The 2-input NOR gate implemented using static logic design style is shown in figure 4.1. The features of static logic style are listed below.

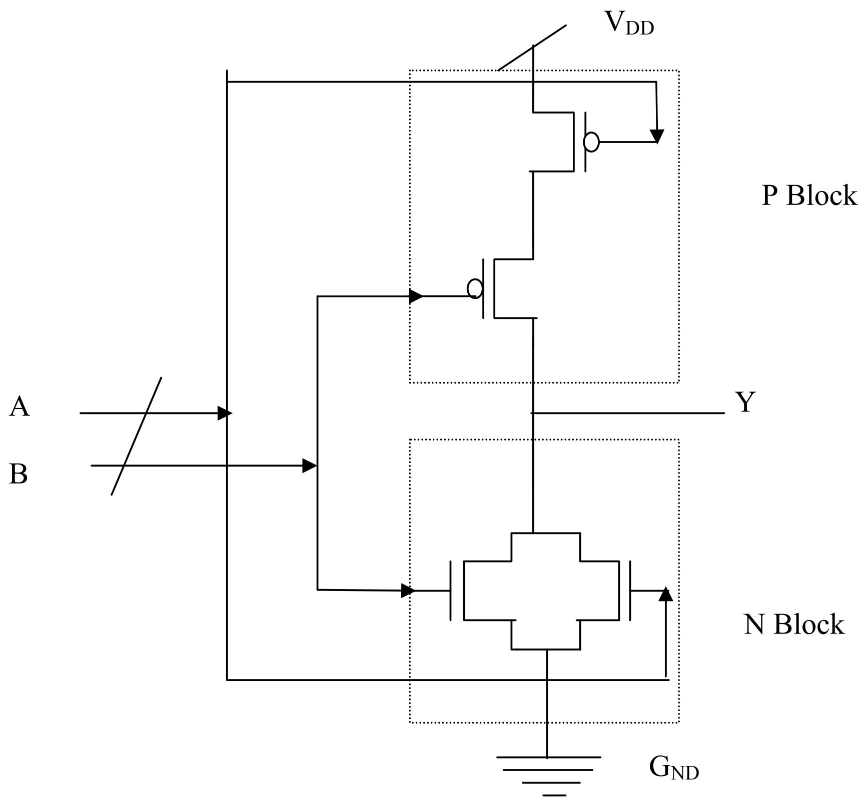


Figure 4.1 Static logic gate

- ❖ Static logic design style is most suitable and widely accepted for many VLSI circuit implementations due to its important properties like high speed, low power, large noise margins, no logic degradation and validity of logic design style at scaled down technologies. Circuits implemented using static logic design style give negligible static power dissipation, as there is no direct path between power supply and ground for any of the logic input combinations under steady state condition.
- ❖ A logic gate with fan-in of *n* requires 2*n* (i.e n number of N-type and n number of P-type) devices. Two logic blocks, the N-block and the P-block, form a CMOS gate. The topology of N-block is the dual of that of the P-block. Since both the blocks have equal number of transistors, transistor count of gate is large.
- ❖ The channel widths of series connected NMOS transistors or PMOS transistors have to be increased to obtain a reasonable conducting current to drive capacitive

loads. The increase in size of transistors results in a significant area overhead, and also an increased gate input capacitance, which may lead to high dynamic power dissipation. The higher gate input capacitance loads the previous stage thereby increasing the delay. The ratio of PMOS/NMOS transistor widths (β) should be chosen optimally for achieving higher speed and lower power consumption as described in section 4.3.

- ❖ Static logic gates also exhibit short-circuit currents. However, by sizing transistors for equal rise and fall times, the short-circuit power component can be minimized.
- ❖ Due to unequal path delays of logic block ⁄ sub-block, the circuit nodes can spuriously switch before their correct logical value stabilizes. Such transitions increase the dynamic power consumption of the circuit. Buffers can be inserted at appropriate locations to equalize path delays.

4.1.2 TG Logic Design Style

TG logic design style presents a class of logic circuits, which use the TG as their basic building blocks. The CMOS TG switch consists of one NMOS and one PMOS transistor, connected as shown in figure 4.2. The gate voltages applied to these two transistors are complementary signals. The CMOS TG switch operates as a bi-directional switch between the nodes A and B which is controlled by signal C. If the control signal C is high, both the transistors are turned on and provide a low resistance current path between nodes A and B. If the control signal C is low, both the transistors are off and the path between nodes A and B will be an open circuit. Figure 4.2 shows the schematic diagram of a CMOS TG switch.

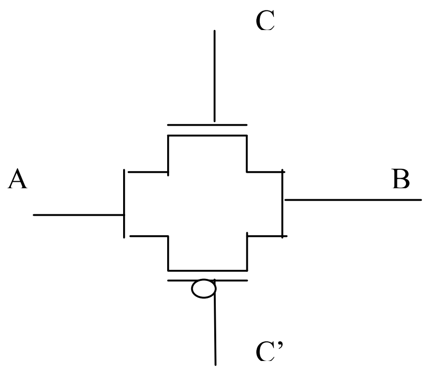


Figure 4.2 CMOS TG switch

The TG logic shows an advantage over the pass transistor logic in that the output levels are not degraded [21]. The TG logic design style passes both logic ‘0’ and logic ‘1’ without any degradation. Figure 4.3 shows the different representations of the CMOS TG switch and its operation.

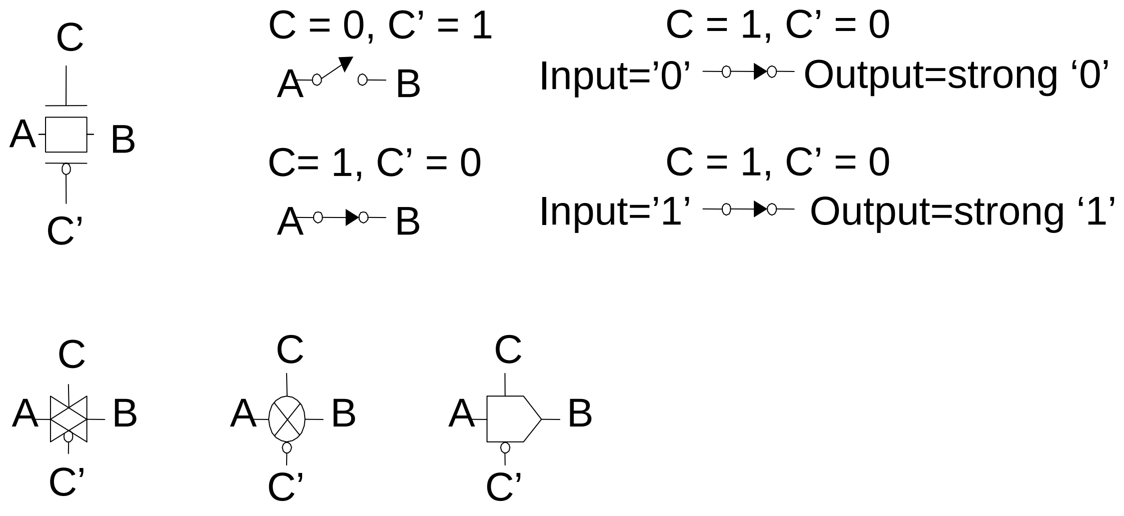


Figure 4.3 Different representations of CMOS TG switch

The speed of the TG logic design style may degrade drastically compared to static logic design style when length of the critical path in a circuit becomes too long and no repeaters are present in the path.

4.2 Dynamic CMOS Logic Circuits [1], [2], [3]

In high density, high performance digital implementations where reduction of circuit delay and silicon area are major objectives; dynamic logic circuits offer several advantages. The operation of all dynamic logic gates depends upon the temporary (transient) storage of charge on circuit node capacitances. This operational property necessitates the periodic updating of internal node voltage levels, since stored charge on capacitors cannot be retained indefinitely. Therefore dynamic logic circuits require periodic clock signals to control charge refreshing.

4.2.1 Domino and Dual Rail Domino Logic Design Style

A domino logic implementation is shown in figure 4.4. It consists of a **dynamic CMOS circuit followed by static CMOS inverter**. The dynamic circuit consists of a PMOS precharge transistor, a NMOS evaluation transistor, and a N-logic block, which, in general, is a series-parallel combination of NMOS transistors activated by the inputs and implementing the required logic. This circuit style uses a single-phase clock (clk). When clk='0', *the dynamic node* is pre-charged to V_{DD} and during the evaluation phase, the dynamic node either remains at V_{DD} or discharges to logic '0'. The numbers of cascaded domino logic gates are limited by the duration of the evaluation clock phase. The features of domino logic style are listed below.

- ❖ In comparison to the static logic, domino logic has a smaller gate input capacitance (C_{in}) and smaller output load capacitance (C_L) due to the absence of large sized PMOS transistors.
- ❖ Domino gate features faster switching speeds due to reduced load capacitances owing to lower input gate capacitance (C_{in}) and smaller output load capacitance (C_L).
- ❖ A simple domino gate with a fan-in of n requires $n + 2$ (i.e. n+1 number of N-type and one P-type) transistors. However, other transistors may be required to remove unwanted effects like charge sharing and for gate cascading.
- ❖ The disadvantage of simple domino logic is that the gate suffers from the charge-sharing problem in which the parasitic capacitances at the internal nodes get coupled to the load capacitance. A weak pull up PMOS transistor connected as shown in figure 4.4, solves this problem [9]. This requires a static inverter and a PMOS transistor; the use of a static inverter makes cascading possible. A simple domino gate without pull up transistor but utilizing a static inverter to avoid cascading problem may show degradation of logic '1' voltage at the input of inverter, which may increase its static power consumption.
- ❖ The advantage of domino gates as compared to static logic gates is that they do not show short-circuit power dissipation and glitching problems, since any node can undergo at most only one transition per clock cycle. The logic also shows the full voltage swing (i.e. $V_{OL} = G_{ND}$ and $V_{OH} = V_{DD}$) as for the case of static logic.

- ❖ However in the circuits designed using domino logic design style, the power consumption may increase due to higher switching activity as compared to equivalent static logic circuits, because all the domino nodes are pre-charged to V_{DD} during each clock cycle. A further increase in power consumption may result from the highly loaded clock distribution network, which consumes a significant amount of dynamic power [9].
- ❖ In the domino logic gate as shown in figure 4.4, the PMOS transistors do not appear in series. Also, C_{in} and C_L are smaller as compared to static logic. Therefore fall times of the domino logic gates will be improved. However, their rise times depend on combined effect of C_L and an additional series transistor (evaluate transistor) in pull-down path. The decreased C_L will tend to decrease the rise time, while the series transistor will tend to increase the rise time.
- ❖ The circuits designed using simple domino logic gates implement only non-inverting logic functions and are incapable of implementing inverting logic functions like NAND, NOT, NOR and XOR etc. mainly due to cascading problem [9].

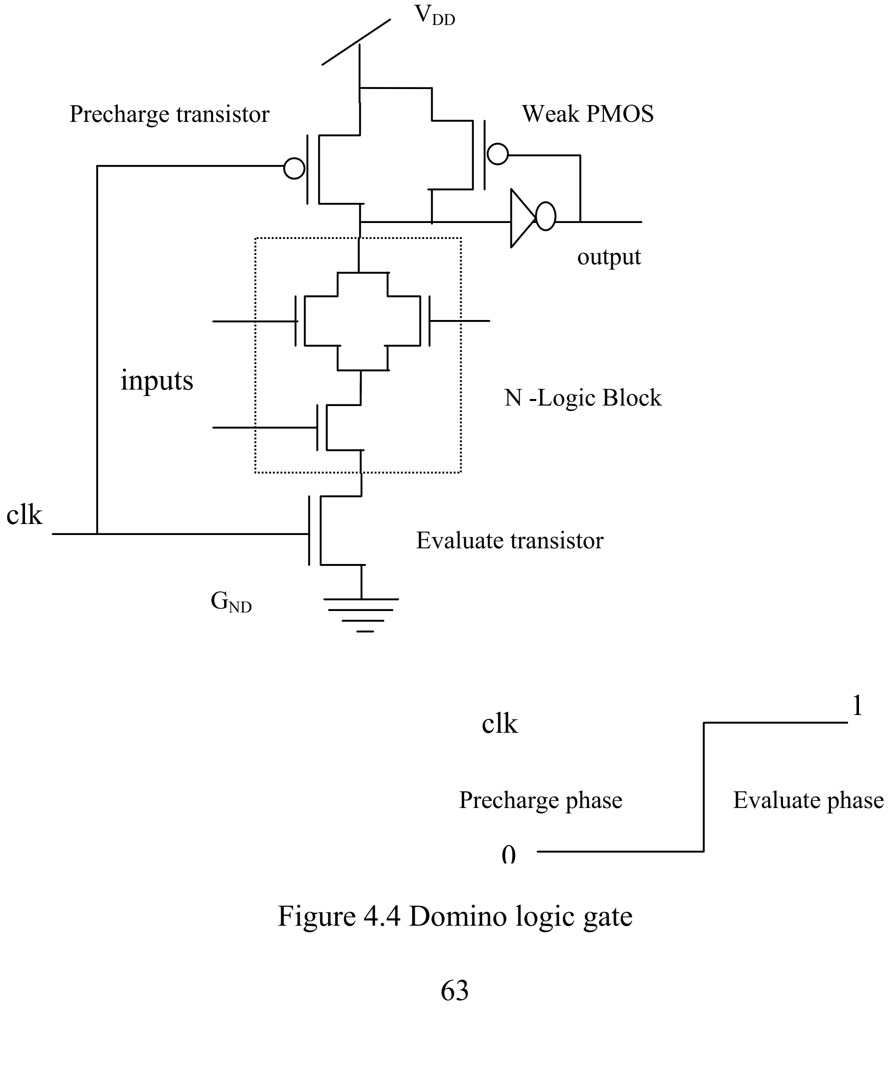


Figure 4.4 Domino logic gate

- The "**dual rail domino logic**" overcomes the limitations of simple domino logic design style, since it can also implement inverting logic functions. In almost all circuit designs inverting logic functions are unavoidable and hence the dual rail domino logic design style is used instead of the simple domino logic design style. The dual rail domino schematic cell for 2-input XOR function is shown in figure 4.5.
- The disadvantage of using dual rail domino logic is that the number of transistors required to implement any logic function will be twice of that required for the simple domino logic. Thus transistor count is increased, which increases the power consumption due to increased switching activity. Another disadvantage is that the circuits designed using dual rail domino logic may increase total length of interconnects, which may also increase the length of critical path, and therefore, the propagation delay of the circuit and the layout core area may increase.

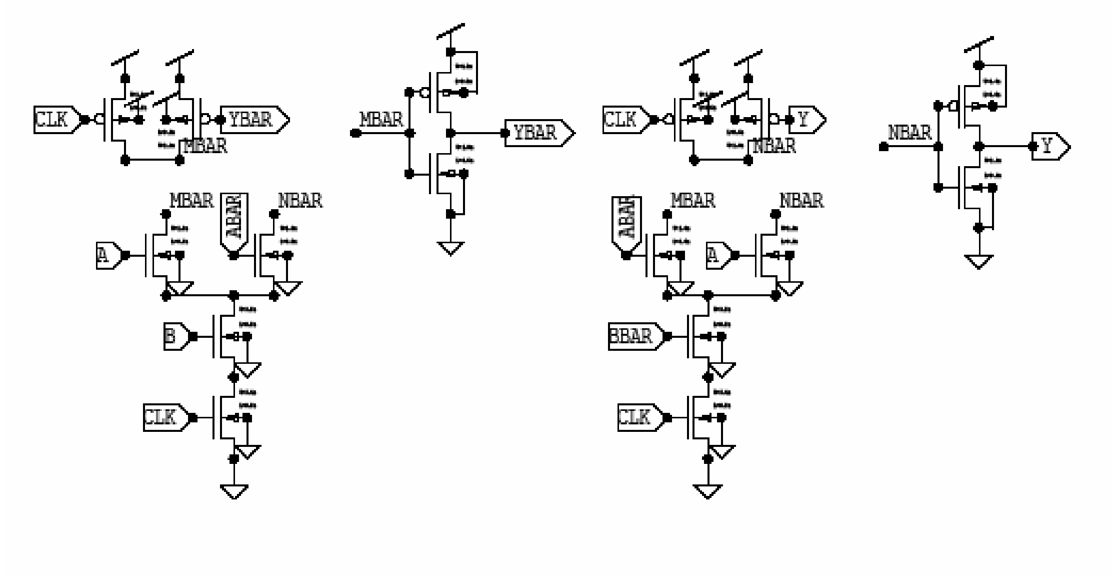


Figure 4.5 Schematic of a 2input XOR cell designed using dual rail domino cell

4.2.2 TSPC Logic Design Style

TSPC logic is one of the high performance dynamic logic circuit design styles which is distinctly different from the NORA logic design style in that it uses only one clock signal, which is never inverted. Since the inverted clock signal is not used anywhere in the system, no clock skew problem exists. Consequently higher clock frequencies can be achieved for dynamic pipelined operations. The features of TSPC logic style are listed below.

- ❖ TSPC logic supports the high-speed pipelined circuit operation, which increases the circuit throughput [26], [29]. A circuit designed using the TSPC logic accepts the inputs every clock cycle and produces the outputs after a fixed number of clock cycles equal to the latency of the circuit. Since the waiting time to apply new inputs is reduced as compared to a purely combinational circuit, hence it shows improved performance. Thus TSPC logic forms an acyclic sequential circuit.
- ❖ Apart from supporting pipelining of operations, an important feature of the TSPC logic cells is their compactness: normally they occupy much less area as compared to dual rail domino logic design style.
- ❖ All the logic cells in all the stages of the pipelined logic circuit perform the pre-charge and evaluation logic operations every clock cycle simultaneously and the clock rate is high (few MHz to GHz) therefore average power consumption and peak power consumption of the circuit is much higher as compared to other logic design styles.

The TSPC logic Cell implementation for the logic function Y=A . B is shown in figure 4.6. Table 4.1 explains the summary of operation for TSPC logic circuits.

Table 4.1 Summary of operation

Cell Type	CLK='1'	CLK='0'
First stage	Dynamic evaluation	Pre-charge + Dynamic latch
Second stage	Dynamic latch	Dynamic evaluation

The operation of first stage is explained in table 4.2 and the operation of second stage is explained in table 4.3.

Table 4.2 Operation of first stage

A	B	CLK	Z	Comments
0	0	1	0	Dynamic evaluation
0	1	1	0	Dynamic evaluation
1	0	1	0	Dynamic evaluation
1	1	1	1	Dynamic evaluation
X	X	0	Latch	Pre-charge + Dynamic latch

Table 4.3 Operation of second stage

Z	CLK	Y	Comments
0	0	0	Dynamic evaluation
1	0	1	Dynamic evaluation
X	1	Latch	Dynamic latch

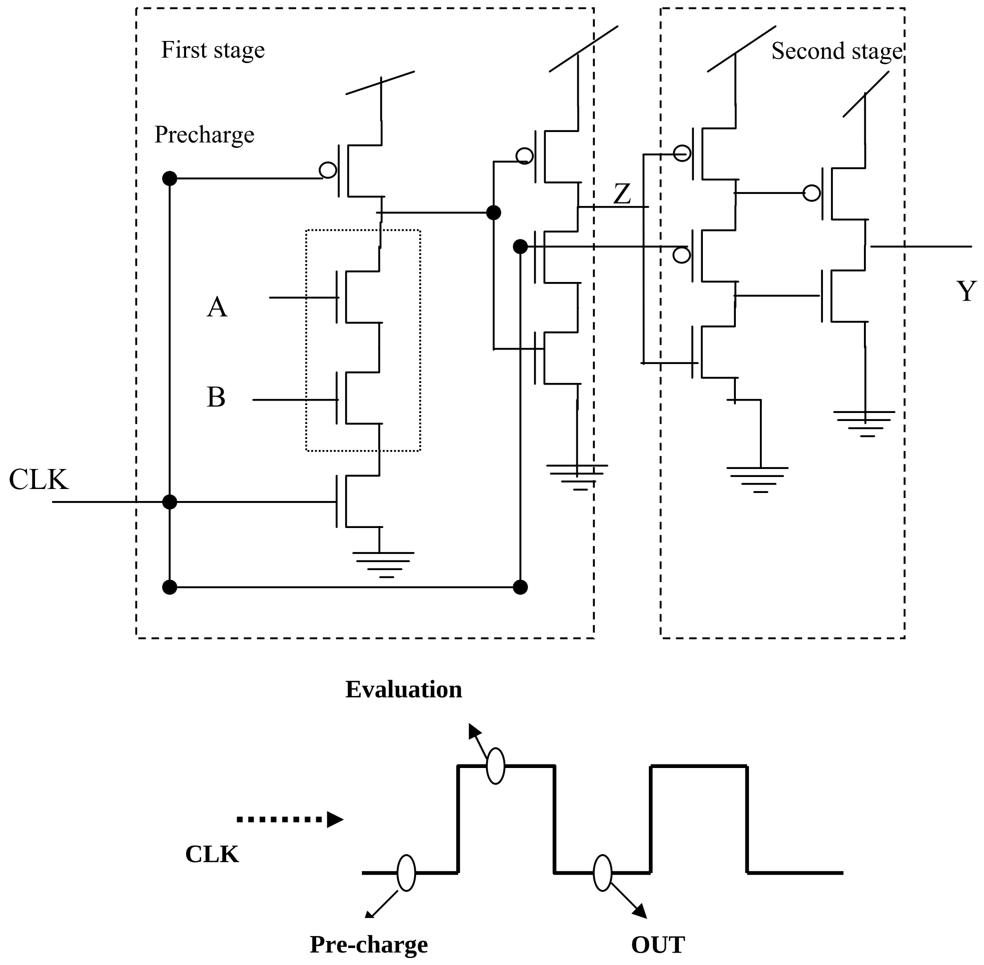


Figure 4.6 TSPC logic circuit

4.3 Determination of β Ratio for High-Speed Digital Designs

In CMOS circuit designs, the low mobility (μ_p) PMOS devices are sized up to attain the same conduction performance as the high mobility (μ_n) NMOS devices. The β is an important ratio in the design of digital circuits using static CMOS logic. The conventional method of estimating β excludes the effect of several technology parameters in estimation of β ratio. In this section we discuss a more accurate estimation of β ratio using relevant technology parameters like thickness of gate oxide (t_{ox}), threshold voltage of NMOS device (V_{tn}), threshold voltage of PMOS device (V_{tp}), zero-bias junction capacitance per unit area of NMOS device (C_{j0n}), zero-bias junction capacitance per unit area of PMOS device (C_{j0p}), side wall zero-bias junction capacitance per unit length of NMOS device (C_{j0swn}), side wall zero-bias junction capacitance per unit length of PMOS device (C_{j0swp}) and built-in potential of PN junction (PB). β ratios are computed for 0.5 μm technology and are compared to their values computed using the conventional method. The β ratio thus computed taking into consideration the above mentioned other technology parameters improves the inverter threshold or switching threshold voltage (V_{th}) by 5% and inverter average propagation delay ($\tau_{P)}$ by 0.6%.

The static CMOS logic has robustness against voltage and transistor scaling, and it provides reliable operation at low voltages together with low switching activity as compared to other logic styles. Static CMOS logic has high noise margins owing to the presence of a static path from output to the appropriate supply that restores the correct logic state in the presence of noise. Logic gates in static CMOS are constructed from a N-block and a P-block. The N-block evaluates the '0' state while the P-block evaluates the '1' state, where only one of the blocks is conducting at steady state. The main drawback of static CMOS circuits is the existence of the P-block because of its low mobility (μ_p) devices as compared to the NMOS devices (μ_n). Therefore, PMOS devices need to be sized up to attain the gate's performance. The highest noise margin for static CMOS is conventionally obtained by using a β ratio of μ_n / μ_p [3], [9], which is also conventionally taken to provide identical current driving capability for the N and P networks. If symmetry and noise margins are not of prime concern, then it is possible to speed up the inverter by reducing the width of PMOS device. Conventionally the best gate performance is supposed to be achieved with a β ratio of $\sqrt{\mu_n / \mu_p}$ [3], [9], because, widening the PMOS improves the t_{PLH} of the inverter but it also degrades the t_{PHL} due to increased capacitance of next stage. The above result is based on the assumption that ratio β depends only on the mobilities whereas in reality it depends upon many other technology parameters. In this analysis we have included the effect of relevant technology parameters in determining the ratio β. 0.5 μm technology has been considered for the present study. Section 4.3.1 explains the calculation of β ratio and V_{th}, section 4.3.2 describes dependence of β ratios on various technology parameter; section 4.3.3 provides comparison of different β ratios for 0.5 μm technology. Section 4.3.4 provides a comparison of V_{th} for different β values, section 4.3.5 provides a comparison of τ_P for different β ratios.

4.3.1 Calculation of β ratio and V_{th}

We assume that a static CMOS inverter is driving another identical static CMOS inverter of same W/L ratio as shown in figure 4.7.

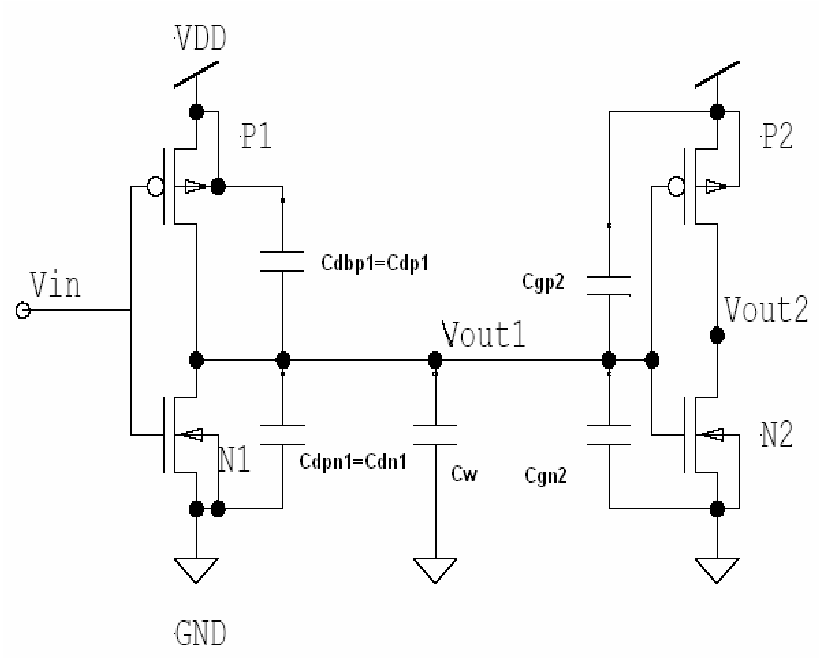


Figure 4.7 A static CMOS inverter driving an identical static CMOS inverter

The load capacitance seen by the first inverter is given by equation (4.1) & (4.2).

$$C_L = C_{dp1} + C_{dn1} + C_{gp2} + C_{gn2} + C_W \tag{4.1}$$

Where, C_{dp1} and C_{dn1} are equivalent drain diffusion capacitance of PMOS and NMOS transistors of the first inverter and C_{gp2} and C_{gn2} are the gate capacitances of the second inverter. C_w represents the wiring capacitance.

If PMOS devices are β times larger than NMOS devices then:

$$C_{dp1} = \beta C_{dn1} \quad \& C_{gp2} = \beta C_{gn2}$$
$$\Rightarrow C_L = (1+\beta)(C_{dn1} + C_{gn2}) + C_W$$
$$where:$$
$$C_{gn2} = (CGD0_n + CGS0_n)W_{n2} + C_{ox}W_{n2}L_{n2}$$
$$C_{dn1} = K_{eqn}A_{Dn1}CJ + K_{eqswn}P_{Dn1}CJSW$$
$$where:$$
$$CGS0_n \ and \ CGD0_n = gate-source \ and \ gate-drain$$
$$overlap \ capaci\tan ce \ per \ unit \ length \ of \ NMOS$$
$$C_{ox} = gate \ oxide \ capaci\tan ce \ per \ unit \ area$$
$$K_{eqn} = \frac{-2\sqrt{\phi_o}}{V_2 - V_1}(\sqrt{\phi_o - V_2} - \sqrt{\phi_o - V_1}), \quad (For \ MJ = MJSW = 0.5)$$
$$K_{eqn} = K_{eqswn} = voltage \ equivalent \ facor, 0 < K_{eqn} < 1$$
$$\phi_o = PB = Built-in \ potential \ of \ PN \ junction$$
$$CJ = C_{j0n} = C_{j0p} \ and \ CJSW = C_{j0swn} = C_{j0swp} \tag{4.2}$$

The value of β to achieve highest speed of the inverter if symmetry and noise margin are not of prime concern is given by equation (4.3) [3]

$$\beta_{optimum} = \sqrt{r(1 + \frac{C_W}{C_{dn1} + C_{gn2}})} \tag{4.3}$$

Conventionally, it is assumed that $C_W << C_{dn1} + C_{gn2}$ and therefore, $\beta_{optimum} = \sqrt{r}$.

$$where: r = \frac{R_{eqp}}{R_{eqn}}$$

$$If \ |V_{tp}| = V_{tn}, then$$
$$\beta_{optimum} = \sqrt{\frac{\mu_n C_{ox}}{\mu_p C_{ox}}} = \sqrt{\frac{\mu_n}{\mu_p}} \tag{4.4}$$

The calculation of β given by equation (4.4) is under the assumption that no significant wiring delay is present and $V_{tn}=|V_{tp}|$. If significant wiring delay is present and $V_{tn}\neq|V_{tp}|$ then,

$$\beta_{optimum} = \sqrt{r(1+\frac{C_W}{C_{dn1}+C_{gn2}})} = \sqrt{r(1+x)}$$

$$where: \qquad (4.5)$$

$$x = \frac{C_W}{C_{dn1}+C_{gn2}}$$

$$r = \frac{\mu_n C_{ox}(V_{gs}-V_{tn})^2}{\mu_p C_{ox}(V_{gs}-V_{tp})^2}$$

The V_{th} is an important parameter characterizing the DC performance of the inverter. Equation (4.6) describes the calculation of V_{th} [1].

$$V_{th} = \frac{V_{tn} + \sqrt{\frac{1}{K_R}}(V_{DD}+V_{tp})}{1+\sqrt{\frac{1}{K_R}}} \qquad (4.6)$$

$$where \quad K_R = \frac{K_n}{K_p} = \frac{\mu_n C_{ox}(W/L)_n}{\mu_p C_{ox}(W/L)_p} = \frac{\mu_n C_{ox}}{\mu_p C_{ox}\times\beta}$$

4.3.2 Dependence of β Ratio on Other Technological Parameters

Equation (4.4) shows that β ratio can be defined in terms of technology dependent process trans-conductance ratio [44], [45], [46]. While, equation (4.5) shows that β ratio depends on many other technology parameters like t_{ox}, V_{tn}, V_{tp}, C_{j0n}, C_{j0p}, C_{j0swn}, C_{j0swp}, built-in potential of PN junction (PB) and capacitance per unit area of metal layer. The technology selected for study is 0.5 μm with the assumption that $C_{j0n}= C_{j0p}$, $C_{j0swn} = C_{j0swp}$ and average interconnect length of 55λ. The lamda values for the 0.5 μm is selected using MOSIS scalable CMOS (SCMOS) design rules (Revision 8.0), λ=0.3 for 0.5 μm MOSIS, SCN_SUBM, 3.3V technology. Using the model parameters C_w, C_{dn1} and C_{gn2} are calculated for 0.5 μm technology, which gives the accurate value of β. The technology parameters are read from respective model files of both NMOS and PMOS devices for 0.5 μm technology.

4.3.3 Comparison of Different β Ratios for 0.5 μm Technology

Table 4.4 shows the comparison of C_w, C_{dn1} and C_{gn2}, r and β for 0.5 μm technology. Equation (4.5) shows the ratio β depends on x and r. Table 4.4 shows the calculated capacitances using appropriate technology parameters. The ratio β given by mobility ratio $\sqrt{\mu_n / \mu_p}$ are computed and shown in table 4.5. The values of β shown in table 4.4 are more accurate as compared to those given in table 4.5 and can be used as a basis for the selection of β for optimization of performance.

Table 4.4 Estimation of β ratio using more accurate method

Technology	C_{ox} (F/cm^2)	V_{DD} (v)	K_{eqn}	C_w (fF)	C_{dn1} (fF)	C_{gn2} (fF)	x	r	β
0.5 μm	3.59E10-7	3.3	0.6489	1.3959	1.665	3.877	0.252	4.97	2.49

Table 4.5 Estimation of β ratio using conventional method

Technology	μ_{nCox} (A/V^2)	$\mu_p C_{ox}$ (A/V^2)	β=Sqrt (μ_n/μ_p)
0.5 μm	196.47 E-6	48.74 E-6	2.00

4.3.4 Comparison of V_{th} for Different β Ratios

Since the static CMOS inverter has very sharp voltage transfer characteristic, the V_{th} is an important parameter characterizing the DC performance of the inverter. The inverter V_{th} is computed for three different values of β:

(i) $\beta = \mu_n / \mu_p$ (ii) $\beta = \sqrt{\mu_n / \mu_p}$ and (iii) $\beta = \sqrt{r(1 + \frac{C_W}{C_{dn1} + C_{gn2}})}$

Table 4.7 shows the comparison of V_{th} values calculated using the above three expressions for choosing β values for 0.5 μm technology. Table 4.7 shows the comparison between the calculated and simulated values of V_{th} using conventional and the more accurate method.

The ideal value of V_{th} must be $V_{DD}/2$ (1.65 V for 0.5 μm technology) for attaining equal high and low noise margins. Comparing V_{th} values shows that choosing $\beta = \sqrt{r(1+\frac{C_W}{C_{dn1}+C_{gn2}})}$ gives a better value of V_{th} for 0.5 μm technology as indicated in table 4.6.

Table 4.7 compares the calculated results with simulated results using spice. The table shows almost 3.5% improvement in V_{th} for calculated results and almost 5 % improvement in V_{th} for simulated results for an inverter designed using 0.5 μm technology.

Table 4.6 Comparison V_{th} for three different β ratios calculated using three different methods

Technology	Calculated V_{th}		
	$\beta = \mu_n / \mu_p$ (do not provide higher speed)	$\beta = \sqrt{\mu_n / \mu_p}$ (Conventional method)	$\beta = \sqrt{r(1+\frac{C_W}{C_{dn1}+C_{gn2}})}$ (More accurate method)
0.5 μm	1.517 V	1.368 V	1.414 V

Table 4.7 Comparison of V_{th} using conventional and the more accurate method

Technology	V_{th}			
	(Conventional method)		(More accurate method)	
	Calculated	Simulation	Calculated	Simulation
0.5 μm (V_{DD}=3.3V)	1.368 V	1.4 V	1.414 V (better)	1.464 V (better)

4.3.5 Comparison of τ_P for Different β Ratios

The τ_P is calculated for both the β ratios using standard τ_{PHL} and τ_{PLH} relations [1]. Table 4.8 shows the comparison of τ_P for both β ratios using standard mathematical relations and through spice simulation. The table shows 1.2 % speed improvement for calculated results and 0.6 % speed improvement for simulated results of a single inverter designed using 0.5 μm technology. The spice simulation result shows higher delays due to the presence of many other real parasitic effects.

Table 4.8 Comparison of τ_P using conventional and the more accurate method (for calculated and simulated values)

Technology	τ_P			
	$\beta = \sqrt{\mu_n / \mu_p}$ (Conventional method)		$\beta = \sqrt{r(1+\frac{C_W}{C_{dn1}+C_{gn2}})}$ (More accurate method)	
	Calculated	Simulation	Calculated	Simulation
0.5 μm	31.58 ps	92 ps	31.08 ps (better)	91.5 ps (better)

The β ratio calculated using conventional method excludes many relevant technology parameters. A more accurate method for estimating β ratio is presented and compared against conventional method for 0.5 μm technology. The V_{th} obtained using β ratio calculated taking into consideration relevant technology parameters shows almost 5 % improvement. This method of β estimation also decreases propagation delay.

In this research work β ratio of 2.5 is chosen for standard cell design using static CMOS logic, CMOS TG logic and TSPC logic to achieve higher speed of logic gates designed for 0.5 μm technology.

4.4 Sizing of Pre-charge and Pre-discharge Transistors for Domino Logic Design Style

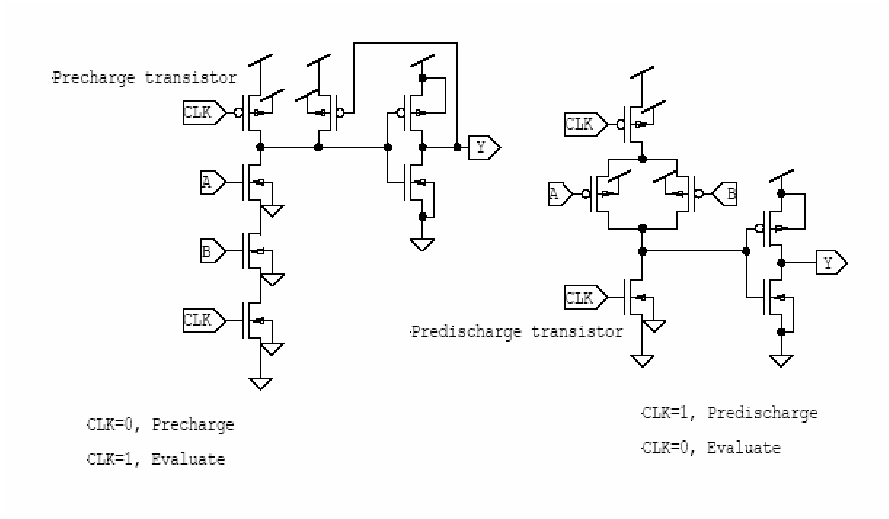
The domino logic circuits are very popular in the design of high performance processors because they offer fast switching speeds and reduced area implementations. The domino logic gates use pre-charge and pre-discharge transistors to charge the intermediate dynamic node. The size of pre-charge and pre-discharge transistors play an important role in achieving higher speeds and smaller areas. In this section we present a method for optimal selection of pre-charge and pre-discharge transistor sizes, based on the amount of capacitance required to be driven at dynamic nodes of CMOS domino logic gates. The product term (area×rise-delay) is used as a figure of merit in the selection of pre-charge transistor size and the product term (area × fall-delay) is used as a figure of merit in the selection of pre-discharge transistor size.

CMOS domino logic has become a popular logic family and is extensively used in most state-of-the-art processors due to its high performance capabilities [47], [48]. The static CMOS gates are slower because at a given time either the pull-up or pull-down network is activated but the input capacitance of the inactive network also loads the active evaluation path [9] [49]. Furthermore, the input gate capacitance increases because the low mobility PMOS transistors are sized up to attain comparable rise and fall delays [9]. Dynamic gates overcome this weakness by eliminating the PMOS transistor blocks and replacing them with a single pre-charge transistor [47], [49]. The dynamic nodes of domino logic gates are periodically pre-charged to 95% of V_{DD} or pre-discharged to 5% of V_{DD} before evaluation in each clock cycle [3]. The sizes of pre-charge and pre-discharge transistors play an important role in area and speed optimization [50]. The larger W/L may degrade the area and smaller W/L may lower the speed. In this section we present a method to optimally select pre-charge transistor size based on product (area×rise-delay) and pre-discharge transistor size based on product (area× fall-delay). The pre-charge PMOS and pre-discharge NMOS transistor sizes can be optimally selected by estimating the load capacitances to be driven by pre-charge and pre-discharge transistors at dynamic nodes. The dynamic switching power in pre-charge and pre-discharge operation will also get reduced due to use of optimal transistor sizes [51].

Section 4.4.1 explains the load capacitance estimation for pre-charge or pre discharge transistor, section 4.4.2 describes the computation of area-delay product, and section 4.4.3 shows the simulation results.

4.4.1 Load Capacitance Estimation for Pre-charge and Pre-discharge Transistors

The domino logic gates considered for estimating load capacitances are shown in figure 4.8. The figure 4.8 (a) shows a domino 2-input AND gate, which can be implemented using PMOS pre-charge transistor and NMOS evaluation transistor. Figure 4.8 (b) shows another implementation of domino 2-input AND gate, which uses a NMOS pre-discharge transistor and PMOS evaluation transistors [3]. The pre-charge transistor PMOS or the pre-discharge transistor NMOS drives the capacitance at the dynamic node. These capacitances are given by equations 4.7, 4.8, 4.9 and 4.10.



(a) (b)

Figure 4.8 (a) 2-input AND domino gate (b) another implementation 2-input AND domino gate

$$C_{T1} = C_{dp,precharge} + C_{L1} \tag{4.7}$$

$$where: C_{L1} = C_{dnA} + C_{dp,pullup} + C_{gp,(inv)} + C_{gn,(inv)} + C_W \tag{4.8}$$

$C_{dp,\ pre\text{-}charge}$ is equivalent drain diffusion capacitance of pre-charge PMOS transistor. $C_{dp,pullup}$ is the pull-up drain diffusion capacitance, C_{dnA} is NMOS evaluation transistor's drain diffusion capacitance. $C_{gp,(inv)}$, $C_{gn,(inv)}$ are the inverter gate capacitance and C_W is wire capacitance loads to be driven by pre-charge transistor.

$$C_{T2} = C_{dn,predischarge} + C_{L2} \tag{4.9}$$

$$where : C_{L2} = C_{dpA} + C_{dpB} + C_{gp,(inv)} + C_{gn,(inv)} + C_W \tag{4.10}$$

$C_{dn,pre\text{-}discharge}$ is equivalent drain diffusion capacitance of pre-discharge NMOS transistor. C_{dpA} and C_{dpB} are PMOS evaluation transistor drain diffusion capacitances. $C_{gp,(inv)}$, $C_{gn,(inv)}$ are inverter gate capacitance and C_W is wire capacitance loads to be driven by the pre-discharge transistor.

The dynamic node capacitances C_{L1} and C_{L2} that exclude drain parasitic capacitance of pre-charge and pre-discharge transistors are estimated using equations 4.8 and 4.10. The estimated capacitances C_{L1} and C_{L2} are to be replaced in the circuits shown in figure 4.9 (a) and figure 4.9 (b) respectively. The Pre-charge load capacitance is pre-assigned with an initial condition (ic=0 V) and pre-discharge load capacitance is pre-assigned with an initial condition (ic=3.3 V). Here, we assume a fixed capacitance of 0.1 pF acting as a load for the pre-charge and pre-discharge transistors with pre assigned initial conditions as stated earlier.

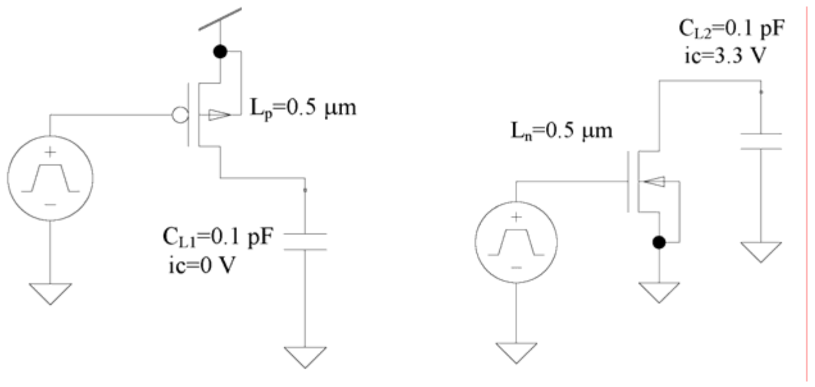


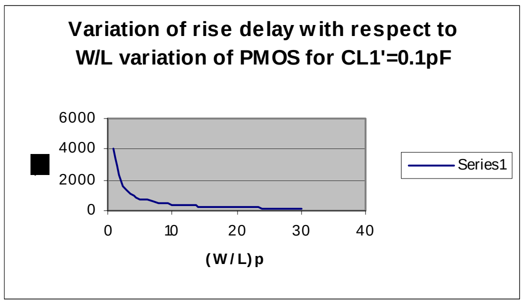
Figure 4.9 (a) PMOS driving the estimated capacitance C_{L1} (b) NMOS driving the estimated capacitance C_{L2}

4.4.2 Computation of Area Delay Product

The circuit shown in figure 4.9 (a) is simulated for different W/L ratios of PMOS pre-charge transistor and corresponding products (area×rise-delay) are calculated taking into account the drain capacitance of the pre-charge transistor besides the estimated connected load (C_{L1}= 0.1 pF). Similarly, the circuits shown in figure 4.9 (b) is simulated for different W/L ratios of NMOS pre-discharge transistor and corresponding products (area×fall-delay) are calculated taking into account the drain capacitance of the pre-discharge transistor besides the estimated load (C_{L2}= 0.1 pF). Transistor level circuit simulations are carried out using T-spice from M/s TANNER Research Inc. taking transistor model parameters of 0.5 μm technology. The channel lengths of NMOS and PMOS devices are kept fixed equal to the minimum channel lengths $L_n=L_p=0.5$ μm.

4.4.3 Simulation Result

The simulation is carried out for a fixed load ($C_{L1}= C_{L2}=0.1$ pF). Figure 4.10 shows the variation of rise- delay (for charging up to 95% of V_{DD}) with W/L ratio of PMOS transistor and figure 4.11 shows the variation of product, (area×rise-delay) with change in W/L ratio of PMOS transistor. Figure 4.12 shows the variation of fall-delay (for discharging up to 5% of V_{DD}) with W/L ratio of NMOS transistor and figure 4.13 shows the variation of product, (area × fall-delay) with change in W/L ratio of NMOS transistor. The L_n and L_p values were kept at 0.5 μm for all simulations.



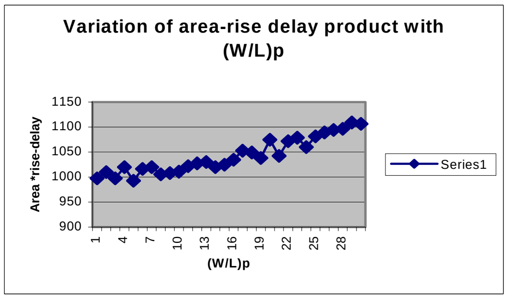
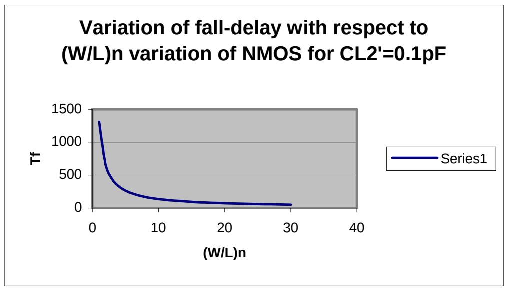


Figure 4.10 Variation of rise- delay   Figure 4.11 Variation of (area×rise-delay) product



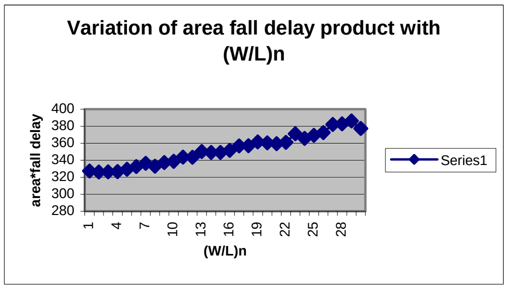


Figure 4.12 Variation of fall-delay   Figure 4.13 Variation of (area × fall-delay) product

Figure 4.11 shows that the product (area×rise-delay) is minimum at a particular W/L ratio (≈5.5) of PMOS pre-charge transistor for ($C_{L1}=0.1$ pF). The transistor sizes smaller than this value have lower speed and the transistor sizes greater than this value have larger area without significant contribution to speed. The optimal W/L selection varies with the amount of capacitance to be driven at dynamic node (C_{L1}) as given by equation (4.8). For given capacitive load at dynamic node, the increase of W/L ratio of PMOS pre-charge transistor beyond an optimum point has little speed advantage, mainly due to increase in the self parasitic capacitance load of the PMOS pre-charge transistor in comparison to the estimated load to be driven (C_{L1}). Figure 4.13 show that product (area×fall-delay) is minimum at a particular W/L ratio (≈3) of NMOS pre-discharge transistor for (C_{L2}=0.1 pF). Transistor sizes smaller than this value reduce speed and transistor sizes greater than this value increase area without any significant contribution to speed. Optimal W/L selection varies with the amount of capacitance to be driven at the dynamic node (C_{L2}) as given by equation (4.10). For a given capacitive load at the dynamic node, the increase in W/L ratio of the NMOS pre-discharge transistor beyond an optimum point has little speed advantage-mainly due to increase in the self parasitic capacitance load of NMOS pre-discharge transistor in comparison to the estimated load to be driven (C_{L2}).

The optimal selection of pre-charge PMOS and pre-discharge NMOS transistor sizes in CMOS domino logic circuits can be achieved by estimating the load capacitances at the dynamic nodes to be driven by the pre-charge and the pre-discharge transistors. The transistor sizes with minimum product terms (area×rise-delay) and (area×fall-delay) are suitable candidates for an optimized design. Increase in the size of these transistors beyond the optimal point increases the device area without significant improvement in speed due to self-parasitic capacitance loading effect. The suggested technique can be effectively used in digital circuit design using domino logic gates to achieve faster and compact designs. In this research work the W/L ratio of pre-charge PMOS transistor in the dual rail domino logic cell design is kept same as the W/L ratio of the top NMOS transistor as shown in figure 4.14, because increase in the size of these transistors beyond this optimal point increases the device area without significant improvement in the speed due to self-parasitic capacitance loading effect. The cell implements Y=A.B.C; the lower NMOS transistors in the cascade of a domino logic cell are graded in size, to improve its transient response [1].

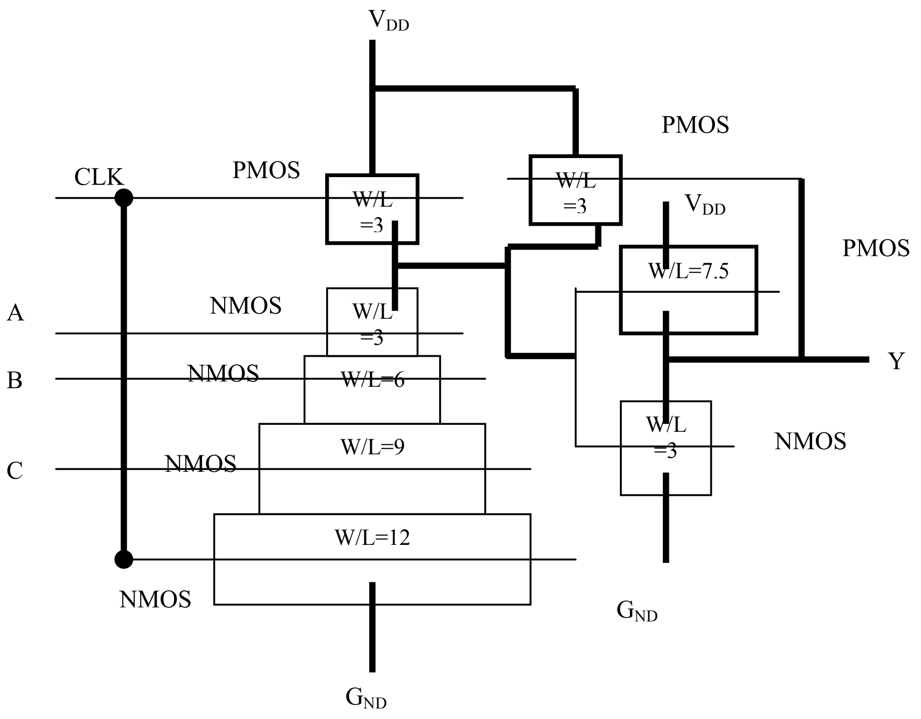


Figure 4.14 Design of a domino logic cell

4.5 Chapter Summary

This chapter presents the fundamentals of the logic design styles considered for level implementation of schematic level designs. These logic design styles are static CMOS logic and dynamic CMOS logic. Static CMOS logic design styles are static logic and TG logic while, dynamic CMOS logic design styles are dual rail domino logic and TSPC logic. It also looks at the problem of transistor sizing for these logic design styles to maximize their performance. These techniques and approaches have been used in the subsequent circuit level implementations of the schematics.

CHAPTER 5

STUDY OF NBTI DEGRADATION IN DIGITAL LOGIC CIRCUITS USING VERILOG HDL

NBTI is identified as one of the most critical reliability concerns for nanometer scale digital integrated circuits. Degradation occurring in PMOS devices is the most critical as it decides the lifetime of CMOS devices in deep sub micron technologies. Research on NBTI is active only within the community of device and reliability physics and leading industrial companies are developing their own models and tools to handle this effect. In this chapter we develop and present a Verilog HDL based switch level circuit modeling technique that incorporates the device-level NBTI modeling to dynamically model the growth of NBTI effect in transistor switches and its evolutionary impact on circuit performance with time. A one bit full adder circuit is used as vehicle to demonstrate the technique. The circuit model describes basic static CMOS logic gates using switch level Verilog description, which also incorporates the model for computing the change in threshold voltage (ΔV_t) and delay (t_p) of PMOS devices after every NBTI stress phase and recovery phase. NBTI stress can be computed by knowing the time for which particular PMOS transistor remains under negative bias (i.e $V_{gs}<0$). Higher modules can be described hierarchically using these basic gates. In this study, a set of random input vectors corresponding to 0.5 years of operation (15768000 random vectors, changing every second) is applied to the digital circuit in order to observe the effect of NBTI degradation on threshold voltage shift of all PMOS devices. The techniques for estimating the degradation in the switching speed of the circuit are also discussed.

It has been reported [52], [53] that NBTI occurring in PMOS devices has emerged as a key reliability degradation issue which affects the lifetime of CMOS devices. The degradation in NMOS devices is far less as compared to that in PMOS and is hence neglected in the presented analysis. NBTI manifests as an increase in the threshold voltage and consequent decrease in drain current and trans-conductance of PMOS transistors [52], [53]. NBTI degradation increases threshold voltage by about 25 to 30% in 10 years [54]. The degradation exhibits power law dependence with time, which can be described using reaction-diffusion model. When a negative gate to source voltage is applied, it initiates a field-dependent reaction at the Si/SiO_2 interface that generates interface traps by breaking the passivated Si–H bonds, which release hydrogen. The released hydrogen diffuses away from the interface, leaving behind a positively charged interface state (Si^+), which is responsible for the increase in threshold voltage [53]. On removing the applied stress the Si-H bond reforms, which recovers of threshold voltage [54]. Gate sizing can tolerate the NBTI degradation in digital circuits, by assigning delay degradation margins to the transistors, so that the expected circuit delay would not exceed the original design specification before the end of the specified lifetime of the circuit [55].

The CAD tools for modeling and managing the NBTI degradation are not widely available due to this effect's complexity and emerging status [33]. Presently research work on NBTI is actively pursued only within the community of device and reliability physicists and leading industrial companies do develop their models and tools to handle this effect [33], [34]. Furthermore, the first order estimation of NBTI degradation on digital logic circuits at the logic level is based on the probability that PMOS transistors in the circuit will be affected by NBTI stress. A stress is the condition when a negative gate to source voltage is applied to the PMOS transistor, the probability of being stressed is non-uniform among PMOS transistors in a circuit [52]. Such methods based on probability are less accurate. In this chapter we suggest a more accurate method using widely available Verilog HDL tool to individually compute the V_t degradation of all PMOS devices. The design can be re-simulated with modified V_t. Since timing verification of complete circuit through simulation requires large computational effort, the circuit delay can also be computed by including the PMOS transistor's V_t degradation on the longest path (critical path) in the logic network. Since longest path in the logic circuit can change over time, the top 10% of the longest paths can be considered for simulation [52] [53]. This chapter presents the model of a 1-bit full adder using Verilog HDL. This switch level model also incorporates necessary description to compute the change in threshold voltage (ΔV_{ts} & ΔV_{tr}) and the delay (t_p) of all PMOS devices due to NBTI stress. It also presents a comparative study of V_t degradation for all the PMOS devices when input vectors corresponding to 0.5 years of operation are applied to the circuit. The computational model used for NBTI degradation process is described in Section 5.1. Section 5.2 describes the incorporation of NBTI degradation model in Verilog simulation. Verilog simulation results estimating NBTI degradation for 1-bit full adder are described in section 5.3

5.1 Model for NBTI Degradation

The predictive model of PMOS NBTI effect enables efficient design examination within the standard CAD environment [33]. For a PMOS device there are two phases of NBTI. In phase-I the NBTI stress condition is reached when a negative voltage is applied between the gate and source terminals of a PMOS device (i.e. when V_g=0, V_{gs}=-V_{DD}). In phase-II the NBTI degradation is recovered (i.e. when V_g= V_{DD}, V_{gs}=0), this phase is referred to as recovery phase. Change in threshold voltage under dynamic NBTI stress or recovery condition is given by equation (5.1) and equation (5.2), which are together taken as the predictive model given in reference [33]. This model provides a solid basis for the tool development. The dynamic NBTI behavior for 90 nm technology node using appropriate technology data is analyzed below.

$$\text{In stress Phase:} \quad \Delta V_{ts} = \sqrt{[\mathrm{K_v}(\mathrm{t - t_0})^{1/2} + \Delta V_{th0}^2]} + \delta_V \qquad (5.1)$$
$$where: \Delta V_{th0} = \Delta V_{tr}$$

$$\text{In recovery Phase:} \Delta V_{tr} = (\Delta V_{th0} - \delta_V)\,[1 - \sqrt{\eta(\mathrm{t - t_0})/\mathrm{t}}\ ] \qquad (5.2)$$
$$where: \Delta V_{th0} = \Delta V_{ts}$$

Where, K_v represent the rate of generation of interface trap levels, ΔV_{th0} represents initial deviation in threshold voltage before device's stress or recovery phase, δ_V accounts for non-H based mechanisms like oxide and other charge residues trapped in Si-SiO_2 interface (=5 mv) and η is empirical constant (=0.35).

$$\mathrm{K_v} = \mathrm{A} \bullet \mathrm{t_{ox}} \sqrt{C_{ox}(V_{gs} - V_{th})}\,[1 - V_{ds} / \alpha(V_{gs} - V_{th})] \bullet \exp[E_{ox}/E_0] \bullet \exp[-E_a/KT]$$

Where, A is an empirical constant, t_{ox} is the thickness of oxide layer in nanometer regime, C_{ox} is capacitance per unit area of oxide layer, V_{ds} & α account for sub threshold leakage effect which alleviates NBTI stress (since E_{ox} at the drain end is smaller than at the source end), E_{ox} is electric field experienced by oxide layer. Note that E_0 is intrinsic electric field in the oxide and E_a is activation energy, which are technology independent characteristics of the reaction.

$$A = 1.8 \times 10^{-3} \quad V/nm/C^{0.5}$$

$$t_{ox} = 1.5nm$$

$$C_{ox} = \varepsilon_{ox}/t_{ox}$$

$$= \frac{3.97 * 8.854 \times 10^{-14}}{1.5 \times 10^{-7}}$$

$$= 2.343 \ \mu F/cm^2$$

$$\Rightarrow \sqrt{C_{ox}(Vgs - Vth)} = \sqrt{2.343 \times 10^{-6} \times 0.8} = 1.3691 \times 10^{-3}$$

$$V_{ds} = 1V$$

$$\alpha = 1.3$$

$$\Rightarrow 1 - [\frac{V_{ds}}{\alpha(V_{gs} - V_{th})}] = 1 - \frac{1}{(1.3 \times 0.8)} = 38.461 \times 10^{-3}$$

$$E_{ox} = (V_{gs} - V_{th})/t_{ox}$$

$$= (1 - 0.2)/1.5e-7$$

$$= 5.34 \ \text{MV/cm}$$

$$E_0 = 2.0 \ \text{MV/cm}$$

$$\Rightarrow \exp(E_{ox}/E_0) = \exp(5.34/2) = 14.44$$

$$E_a = 0.13 \ \text{eV}$$

$$T = 273 + 50 = 323^0 K$$

$$\Rightarrow KT = \frac{kT}{q} = \frac{1.38 \times 10^{-23} * 323}{1.6 \times 10^{-19}} = 0.027V$$

$$\therefore \exp(-E_a/KT) = \exp(-0.13/0.027) = 9.15 \times 10^{-3}$$

$$\Rightarrow K_v = A * t_{ox}\sqrt{C_{ox}(V_{gs} - V_{th})}[1 - Vds/\alpha(V_{gs} - V_{th})] \times \exp[E_{ox}/E_0] \times \exp[-E_a/KT]$$

$$= 1.8 \times 10^{-3} \bullet 1.5 \bullet 1.3691 \times 10^{-3} \bullet 38.461 \times 10^{-3} \bullet 14.44 \bullet 9.15 \times 10^{-3} = 1.88 \times 10^{-8}$$

Putting these values equation (5.1) reduces to equation (5.3) and equation (5.2) reduces to equation (5.4). Equation (5.3) gives ΔV_{ts}, which is the accumulated variation in V_t after stress phase, and equation (5.4) gives ΔV_{tr}, which is accumulated variation in V_t after recovery phase. These variations accumulate for all the stress and recovery cycles.

The value of V_t after any stress or recovery can be obtained by adding accumulated changes to V_t value at the beginning. Note that the ΔV_{ts} is function of time and different technologies give different final values for ΔV_{ts}, hence final value of ΔV_{ts} is independent of frequency [33].

$$\Delta V_{ts} = \sqrt{1.88 \times 10^{-8} (t - t_o)^{0.5} + \Delta V_{tr}^2} + 0.005$$
$$\Rightarrow V_{ts} = V_t + \Delta V_{ts} \tag{5.3}$$

$$\Delta \mathrm{V}_{\mathrm{tr}} = (\Delta V_{ts} - 0.005)[1 - \sqrt{\frac{0.35(t - t_0)}{t}}] \tag{5.4}$$
$$\Rightarrow \mathrm{V}_{\mathrm{tr}} = V_t + \Delta V_{tr}$$

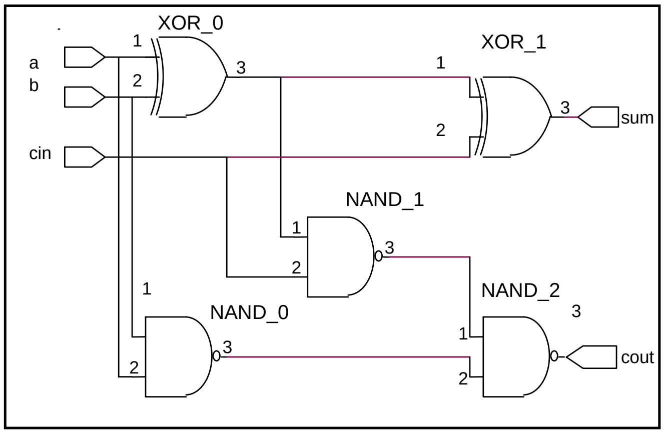
The propagation delay of the device is calculated using alpha-power law MOSFET model [56], as given in equation (5.5),

$$t_p = C \cdot \frac{V_{DD}}{\left(V_{DD} - |V_t|\right)^{\alpha}} = \frac{1}{\left(1 - |V_t|\right)^{1.3}} \tag{5.5}$$

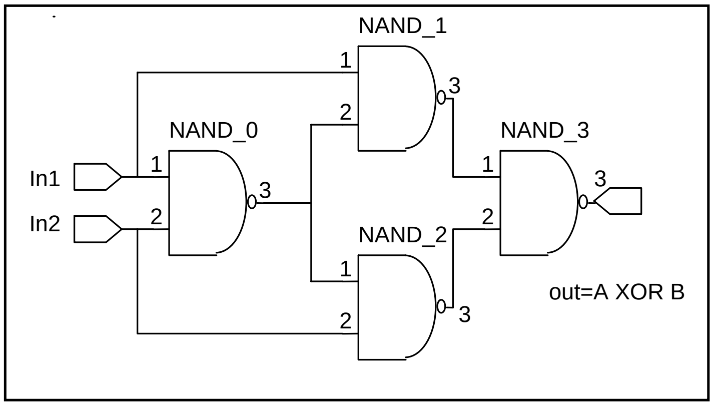
Where, V_{DD} is the supply voltage (V_{DD} =1V) ; α is a technology based constant (α =1.3) and for normalized delay the C =1 pF, hence delays are in pico seconds. The V_t is the degraded threshold voltage of the PMOS after any stress or recovery phase. Thus, decrease in V_t of PMOS will increase delay of PMOS.

5.2 Incorporation of NBTI Degradation Model in Verilog Simulation

In our study, a 1-bit full adder is considered. Similar modeling can be done for any other combinational or sequential circuit to study the NBTI effect. The full adder circuit is implemented hierarchically i.e. a 1-bit full adder is implemented using two 2-input XOR gates and three 2-input NAND gates. The 2-input XOR gate is implemented using four 2-input NAND gates, as shown in figure 5.1. Each 2-input NAND gate has static CMOS implementation. Thus, the 2-input NAND gate is the basic building block for the whole circuit.



(a)



(b)

Figure 5.1. (a) Full Adder schematic (b) XOR gate schematic

A Verilog code has been written to describe the 1-bit full adder. A NAND gate is modeled using switch-level model, which also incorporates the description for computing the change in threshold voltage (ΔV_{ts} and ΔV_{tr}) and the delay (t_p) of PMOS devices after any stress or recovery phase. Inside each NAND gate module, the time for which each of the PMOS switch is getting stressed (i.e. time for each gate input signal remains logic '0') is found and then ΔV_{ts} and ΔV_{tr} is calculated using equation (5.3) and equation (5.4). Now, accumulated change in ΔV_{ts} is added to value V_t at the beginning to get the new threshold voltage. The Verilog code for NAND2 gate is as shown below:

```
module my_nand (out,in1,in2);
output out;
input in1,in2;
wire w;
real
vt1=0.2,vt2=0.2,k=0.0000000188,vt1s,vt1r,vt2s,vt2r,dv1s,dv2s,dv1r,dv2r,delay1,delay2;
                      //threshold voltage Vt and change in threshold voltage dv=ΔV
time t1_in1,t2_in1,t1_in2,t2_in2,dt_in1,dt_in2; //time variables
supply1 pwr;
supply0 gnd;

pmos (out,pwr,in1),
(out,pwr,in2);  //switch-level modelling with static CMOS implementation
nmos (out,w,in1),
(w,gnd,in2);
initial
begin
 dv1r=0;
 dv1s=0;
 dv2r=0;
 dv2s=0;
 end

 always @ (negedge in1)
t1_in1 = $time;            //noting the time when PMOS1 get stressed

always @ (posedge in1)
begin
t2_in1 = $time;    //noting the time when PMOS1 is relieved from stress
dt_in1 = t2_in1 - t1_in1;  //calculating the time for which PMOS1 remains stressed

dv1s = ((((k*k)*((dt_in1)**0.5))+(dv1r*dv1r))**0.5+0.005); // dv1s=ΔVts
dv1r = ((dv1s-0.005)*(1-(0.35*(dt_in1/t2_in1))**0.5)); // dv1r=ΔVtr
```

```
vt1s = vt1 + dv1s; //degraded Vt = original Vt + (change in Vt= ΔVts)
vt1r = vt1 + dv1r; //degraded Vt = original Vt + (change in Vt= ΔVtr)
        delay1 = 1/((1-vt1r)**1.3);  //delay for the PMOS1 taking C=1 pF, alpha=1.3
end

always @ (negedge in2)
t1_in2 = $time;    //noting the time when PMOS2 get stressed

always @ (posedge in2)
begin
t2_in2 = $time;    //noting the time when PMOS2 is relieved from stress
dt_in2 = t2_in2 - t1_in2;   //calculating the time for which PMOS1 remains stressed
dv2s = ((((k*k)*((dt_in2)**0.5))+(dv2r*dv1r))**0.5+0.005); // dv2s=ΔVts
 dv2r = ((dv2s-0.005)*(1-(0.35*(dt_in2/t2_in2))**0.5)); // dv2r=ΔVtr
vt2s = vt2 + dv2s;  //degraded Vt = original Vt + (change in Vt= ΔVts)
vt2r = vt2 + dv2r; //degraded Vt = original Vt + (change in Vt= ΔVtr)
        delay2 = 1/((1-vt2r)**1.3);  //delay for the PMOS1 taking C=1 pF, alpha=1.3
end
endmodule
```

Hierarchical modeling is used to describe XOR gate and 1-bit full adder. The final V_t of each PMOS device can be read after every input vector application. These input vectors are generated randomly in the test bench and applied to the design under simulation after every 1 sec and simulated for a period of 0.5 years and corresponding degradation in V_t and delays are obtained through simulation. The simulation tool used is "Modelsim" from Mentor Graphics. Since the ΔV_{ts} is function of time, the value of ΔV_{ts} corresponding to warranty period can be found and used for robust circuit design to meet the specification.

5.3 Verilog Simulation Results Showing NBTI Degradation for 1-bit Full Adder

Initially, the $|V_t|$ of all the PMOS switches is set to 0.2 V. The initial delay t_p obtained with this value of V_t is 1.336 psec. Then after applying input vectors corresponding to 0.5 years of operation to the 1-bit full adder, we observed the threshold voltages shifts and delay changes of all the PMOS switches of 1-bit full adder. Table 5.1 shows the increment in V_t's and delay (t_p's) of the 22 PMOS devices of 1-bit full adder.

Table 5.1 Degradation in V_t and propagation delay for all PMOS devices in a 1-bit full adder

S. No	Level 01	Level 02	PMOS	$\|V_t\|$ (after 0.5 years)	Delay (after 0.5 years) ps
1		NAND-0	P-1	0.212377	1.35274
2			P-2	0.212377	1.35274
3		NAND-1	P-1	0.212377	1.35274
4			P-2	0.212377	1.35274
5		NAND-2	P-1	0.212376	1.35274
6			P-2	0.212376	1.35274
7	XOR-0	NAND-0	P-1	0.212377	1.35274
8			P-2	0.212377	1.35274
9		NAND-1	P-1	0.212377	1.35274
10			P-2	0.212377	1.35274
11		NAND-2	P-1	0.212376	1.35274
12			P-2	0.212376	1.35274
13		NAND-3	P-1	0.210848	1.34935
14			P-2	0.210848	1.34935
15	XOR-1	NAND-0	P-1	0.212377	1.35274
16			P-2	0.212377	1.35274
17		NAND-1	P-1	0.212377	1.35274
18			P-2	0.212377	1.35274
19		NAND-2	P-1	0.210848	1.34935
20			P-2	0.210848	1.34935
21		NAND-3	P-1	0.21169	1.35121
22			P-2	0.21169	1.35121

The degradation in the switching speed of the circuit can be computed by re-simulating the circuit with modified V_t. The circuit delay can also be computed by simulating the PMOS transistors on the longest path (critical path) in the logic network in order to reduce computational effort [52], [57]. Since longest path in the logic circuit can change over time, the top 10% of the longest paths can be considered for estimation.

NBTI is one of the most critical reliability issues for deep sub micron technology. In this chapter we presented a new simple method to study NBTI degradation of any digital circuit using Verilog HDL. Since the research on NBTI is active only within the community of the device and reliability physicists and leading industrial companies develop their own models and tools to handle this effect, such NBTI study using Verilog HDL provides an open general method for modeling and estimation of NBTI degradation in CMOS circuits implemented using nanometer scale technologies. The degradation in V_t of all PMOS devices in the circuit can be obtained using Verilog HDL switch level modeling. Simulation result shows the degradation in the V_t of all PMOS devices in 1-bit full adder. The technique to compute degradation in the switching speed of the circuit has also been discussed.

5.4 Chapter Summary

This chapter presents NBTI degradation in nanometer scale digital VLSI circuits, which is identified as one of the most critical reliability concerns more recently discovered. The chapter studies the theoretical device level models and proposes a technique for incorporation of NBTI degradation in given circuit using switch level Verilog description. The developed technique has been used to dynamically simulate NBTI degradation in a full adder modeled using static CMOS logic design style. The NBTI degradation study for different data path elements like signed multiplier (i.e Baugh Wooley multiplier and Booth encoded Wallace tree multiplier), unsigned multiplier (i.e. MUX based and 2×2 cell based multiplier) and barrel shifter architectures (i.e MUX based barrel shifter and Pereira's barrel shifter) designed using different logic design style is reserved as future scope of the work.

CHAPTER 6
VLSI IMPLEMENTATION AND SIMULATION RESULTS OF DIFFERENT MULTIPLIERS AND BARREL SHIFTER ARCHTECTURES

In this chapter a comparison of VLSI implementation results of different architectures chosen for study for data path elements including signed multiplier (i.e Baugh Wooley multiplier and Booth encoded Wallace tree multiplier), unsigned multiplier (i.e. MUX based and 2×2 cell based multiplier) and barrel shifter (i.e MUX based barrel shifter and Pereira's barrel shifter) is presented.

A cell library consisting of functional cells was defined. Corresponding to the functional cell library, three different schematic libraries were designed using static logic, TG logic and dual rail domino logic design styles using the basic design principles. Three different physical versions of each schematic library were developed by respectively sizing the W/L ratios of the NMOS transistor to values of 3, 5 and 7. As discussed in chapter 1 the W/L values smaller than 3 were also experimented with but not considered further as they resulted in parasitic dominated slower speeds due to weak drives of transistors and were not considered good candidates for high performance. All the physical library versions were implemented in 0.5 μm, N-well CMOS process (SCN_SUBM, lambda=0.3) of MOSIS.

The layout assemblies for the 4-bit, 8-bit, 12-bit and 16-bit multiplier and barrel shifter circuits were carried out using these cell libraries and automatic placement and routing tool LEDIT (SPR) from M/s Tanner Research Inc [35], [36]. The generated layouts were then simulated after parasitic extraction using circuit simulator, ELDO spice. Supply voltage V_{DD} was kept at 3.3 V.

The performance parameters and design attributes for comparison were propagation delay, average power, maximum power, leakage power, transistor count, core layout area, routing length and number of vias.

6.1 Comparison Between Baugh Wooley Multiplier and Booth Encoded Wallace Tree Multiplier Implementations

Table 6.1 (a), 6.1 (b) and 6.1 (c) show the comparison of the Baugh Wooley signed multiplier and the Booth encoded Wallace tree signed multiplier.

Comparison of the two architectures shows that the Baugh Wooley multiplier is much faster than the Booth encoded Wallace tree multiplier and consumes much less power due to fewer numbers of transistors and smaller layout area. The routing length and number of vias are also much smaller in Baugh Wolley multiplier implementation. The Baugh Wolley multiplier implementation features smaller leakage power as compared to Booth encoded Wallace tree multiplier due to its smaller transistor count.

For the Baugh Wooley multiplier architecture the implementations using TG logic design style and domino logic design style result in much larger delay and power as compared to static logic design style implementation because routing length is increased for these implementations.

By comparison in Booth encoded Wallace tree multiplier the TG logic design style implementation is slightly faster and consumes lower power as compared to static logic implementation, since it has almost comparable routing length, but the domino logic design style implementation is slower and consumes more power compared to static logic design style mainly due to much larger routing length and use of many simpler logic cells in the design implementation.

The increasing of sizes of devices does not show appreciable improvements in delay and only increases the core layout area and power consumption.

Table 6.1 (a) Performance and Characteristics of Baugh Wooley and Booth encoded Wallace tree multiplier for W/L=3

Size	Performance Indices	Multiplier Architecture Type					
		Baugh Wooley			Booth encoded Wallace tree		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	9.33	19.07	20.18	19.15	17.35	32.64
	Average Power (mw) at 20 MHz	4.52	14.61	22.65	27.949	17.96	24.32
	Max Power (mw)	192.19	2455.28	1653.71	177.25	403.77	4085.4
	Leakage Power (nW)	29.719	134.79	54.69	231.15	215.00	1265.5
	Transistor Count	8404	9862	13790	15288	14664	27312
	Core Area (mm^2)	16.40	32.17	77.65	44.17	58.56	156.98
	Total Routing Length (mm)	1005	1778	5117	2335	2750	9436
	No. of Via	2524	2428	5586	8266	7717	17211
12-bit	τ (ns)	6.87	13.92	14.99	18.08	10.00	30.102
	Average Power (mw) at 20 MHz	2.50	7.91	12.34	9.71	11.50	23.82
	Max Power (mw)	113.55	1228.47	916.46	106.57	410	2329.8
	Leakage Power (nW)	22.05	78.62	35.6	119.74	128.93	28.81
	Transistor Count	4692	5502	7686	9298	8838	16450
	Core Area (mm^2)	8.40	17.46	37.44	23.56	30.42	75.18
	Total Routing Length (mm)	503.54	914	2495	1110	1335	4446
	No. of Via	1380	1424	3161	4837	4664	10299
8-bit	τ (ns)	4.43	8.79	10.072	13.49	8.49	21.78
	Average Power (mw) at 20 MHz	1.089	3.29	5.286	2.66	4.94	10.93
	Max Power (mw)	55.53	494.12	398.62	59.9	145.44	1082.4
	Leakage Power (nW)	8.16	32.77	16.67	28.97	62.55	17.11
	Transistor Count	2068	2422	3374	4622	4186	7846
	Core Area (mm^2)	2.99	6.13	14.71	9.69	11.20	30.66
	Total Routing Length (mm)	159.28	264.51	889.86	420.015	467.20	1730.48
	No. of Via	617	596	1425	2161	2049	4791
4-bit	τ (ns)	2.02	3.69	5.33	6.32	7.03	12.13
	Average Power (mw) at 20 MHz	0.2739	0.59	1.197	0.44	2.43	3.59
	Max Power (mw)	18.64	93.29	93.94	30.56	81.95	372.9
	Leakage Power (nW)	2.79	8.31	0.465	17.16	22.19	7.75
	Transistor Count	532	622	854	1638	1474	2738
	Core Area (mm^2)	0.73	1.39	3.04	2.790	3.38	8.36
	Total Routing Length (mm)	29.81	50.022	145.10	109.604	125.33	413.67
	No. of Via	142	129	364	712	657	1647

Table 6.1 (b) Performance and Characteristics of Baugh Wooley and Booth encoded Wallace tree multiplier for W/L=5

Size	Performance Indices	Multiplier Architecture Type					
		Baugh Wooley			Booth encoded Wallace tree		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	9.25	18.47	18.67	14.24	16.12	31.85
	Average Power (mw) at 20 MHz	7.49	23.79	36.63	83.14	27.46	64.68
	Max Power (mw)	322.67	3724.37	2762.18	311.22	660.82	6303
	Leakage Power (nW)	37.16	214.69	32.07	281.64	357.47	2681
	Transistor Count	8404	9862	13790	17404	14664	27312
	Core Area (mm^2)	16.09	37.12	87.43	53.24	59.91	172.81
	Total Routing Length (mm)	947.69	2134.5	5564.03	2447.05	2553.89	10027
	No. of Via	2457	2414	5536	7999	7888	17213
12-bit	τ (ns)	6.86	13.51	14.09	14.85	9.26	28.63
	Average Power (mw) at 20 MHz	4.158	12.86	20.07	58.59	18.34	37.87
	Max Power (mw)	191.03	1898.98	1580.25	259.59	470.50	3851
	Leakage Power (nW)	23.74	119.51	38.50	402.81	214.26	43.68
	Transistor Count	4692	5502	7686	10530	8838	16450
	Core Area (mm^2)	8.52	17.55	43.04	26.66	31.62	93.46
	Total Routing Length (mm)	484.61	872.9	2711	1195.14	1278.41	5356
	No. of Via	1385	1354	3136	4697	4582	10202
8-bit	τ (ns)	4.42	8.54	9.52	11.10	8.39	20.57
	Average Power (mw) at 20 MHz	1.815	5.4	8.64	7.79	8.20	17.35
	Max Power (mw)	93.91	796.28	672.98	105.92	243.33	1778
	Leakage Power (nW)	10.13	52.40	26.77	151.76	104.53	17.78
	Transistor Count	2068	2422	3374	5120	4186	7846
	Core Area (mm^2)	3.05	6.76	16.11	10.78	13.30	37.72
	Total Routing Length (mm)	155.66	285.95	889.8	458.57	496.58	1840.9
	No. of Via	615	612	1421	2122	2117	4770
4-bit	τ (ns)	2.03	3.6	5.07	6.01	6.95	11.46
	Average Power (mw) at 20 MHz	0.462	0.99	2.00	1.38	4.04	5.77
	Max Power (mw)	31.21	148.10	162.89	52.21	134.79	613
	Leakage Power (nW)	4.48	13.46	2.71	27.80	37.22	10.16
	Transistor Count	532	622	854	1804	1474	2738
	Core Area (mm^2)	0.79	1.56	3.54	3.29	3.92	9.95
	Total Routing Length (mm)	33.13	51.81	150.32	120.71	132.10	443.6
	No. of Via	159	141	372	715	668	1585

Table 6.1 (c) Performance and Characteristics of Baugh Wooley and Booth encoded Wallace tree multiplier for W/L=7

Size	Performance Indices	Multiplier Architecture Type					
		Baugh Wooley			Booth encoded Wallace tree		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	9.15	18.22	18.65	13.83	15.89	30.98
	Average Power (mw) at 20 MHz	10.42	37.09	50.75	157.44	38.42	88.20
	Max Power (mw)	461.93	4861.99	3789.06	438.83	1163.78	8943
	Leakage Power (nW)	65.01	287.45	28.53	620.57	501.43	2095
	Transistor Count	8404	9862	13790	17404	14664	27312
	Core Area (mm^2)	18.18	36.71	911.24	54.69	61.38	189.53
	Total Routing Length (mm)	1086.3	1849.2	5569.79	2543.18	2558.43	10700
	No. of Via	2445	2446	5629	7856	7696	17309
12-bit	τ (ns)	6.77	13.31	14.02	14.69	9.01	27.62
	Average Power (mw) at 20 MHz	5.77	17.84	27.97	29.46	25.38	60.67
	Max Power (mw)	274.42	2557.17	2112.69	188.21	649.40	5286
	Leakage Power (nW)	39.22	165.72	30.12	391.18	300.42	40.14
	Transistor Count	4692	5502	7686	10530	8838	16450
	Core Area (mm^2)	8.96	18.77	44.07	26.60	32.62	98.20
	Total Routing Length (mm)	475.13	897.33	2657.5	1151.38	1353.37	5434
	No. of Via	1382	1363	3102	4651	4531	10276
8-bit	τ (ns)	4.38	8.41	9.48	11.10	8.32	19.97
	Average Power (mw) at 20 MHz	2.508	7.47	11.96	14.86	11.46	23.92
	Max Power (mw)	135.53	1073.62	928.09	147.88	333.09	2488
	Leakage Power (nW)	11.64	71.92	5.19	134.30	146.28	18.16
	Transistor Count	2068	2422	3374	5120	4186	7846
	Core Area (mm^2)	3.45	7.17	17.13	11.19	13.77	38.41
	Total Routing Length (mm)	166.12	288.03	934.8	457.45	522.34	1876.5
	No. of Via	606	574	1376	2190	2089	4790
4-bit	τ (ns)	2	3.55	4.96	6.00	6.91	11.34
	Average Power (mw) at 20 MHz	0.6435	1.39	2.78	2.72	5.62	8.03
	Max Power (mw)	43.79	204.33	226.10	74.10	185.65	867
	Leakage Power (nW)	4.16	18.62	1.35	84.17	51.91	6.90
	Transistor Count	532	622	854	1804	1474	2738
	Core Area (mm^2)	0.78	1.62	3.56	3.20	4.04	11.27
	Total Routing Length (mm)	31.40	51.12	145.35	112.12	129.98	524.6
	No. of Via	136	134	358	678	682	1652

6.2 Comparison Between MUX Based Multiplier and 2×2 Cell Based Multiplier Implementations

Tables 6.2 (a), 6.2 (b) and 6.2 (c) show the comparison of the unsigned multipliers i.e. MUX based and 2×2 cell based multiplier.

Comparison of these two architectures shows that the MUX based multiplier architecture is slower as compared to the 2×2 cell based multiplier but it consumes lesser power and features reduced transistor count, smaller core layout area and reduced routing length. This shows that MUX based multiplier architecture is inherently slower.

In both MUX based and 2×2 cell based multiplier architectures the implementations using the TG logic design style show larger delay and power consumption as compared to the static logic implementation mainly due to larger routing lengths and increased number of vias. On the other hand the domino logic design style implementation shows comparable or slightly improved delay performance at the cost of large power consumption, this may be due to the use of more complex logic cells and hierarchical approach followed in designing these architectures.

Increasing the sizes of devices does not show appreciable improvement in delays but only results in larger core layout areas and power consumption.

Table 6.2 (a) Performance and characteristics of MUX based and 2×2 cell based multiplier for W/L=3

Size	Performance Indices	Multiplier Architecture Type					
		MUX based			2×2 Cell based		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	14.15	38.35	13.81	10.94	17.48	10.82
	Average Power (mw) at 20 MHz	22.05	54.63	24.26	31.58	67.67	43.89
	Max Power (mw)	623.46	2956.48	1841.31	543.34	4097.19	2620.89
	Leakage Power (nW)	53.34	120.32	68.03	64.02	288.42	127.87
	Transistor Count	10168	8758	15678	16032	21744	29232
	Core Area (mm^2)	23.76	33.86	99.81	35.07	75.87	178.41
	Total Routing Length (mm)	1386.71	1861.1	6651.38	2101.75	4015.68	11509.3
	No. of Via	3452	3795	8331	5704	5594	12596
12-bit	τ (ns)	11.67	21.95	10.86	7.8	13.74	8.64
	Average Power (mw) at 20 MHz	7.12	19.98	14.22	16.83	35.34	23.62
	Max Power (mw)	363.56	1599.63	1083.68	342.90	2227.56	1447.11
	Leakage Power (nW)	31.86	73.39	33.12	43.16	156.09	76.89
	Transistor Count	6048	5286	9382	8668	11806	15918
	Core Area (mm^2)	12.68	16.99	50.6	17.86	36.32	88.79
	Total Routing Length (mm)	692.22	814.45	3160.92	997.09	1838.01	5708.2
	No. of Via	2063	2059	4833	3073	2997	6791
8-bit	τ (ns)	9.01	12.31	8.00	4.52	12.25	5.65
	Average Power (mw) at 20 MHz	1.77	8.09	7.637	17.82	13.78	9.57
	Max Power (mw)	197.85	769.16	534.67	463.35	833.52	608.94
	Leakage Power (nW)	15.13	37.19	5.18	7.623	65.34	14.74
	Transistor Count	2952	2646	4622	3588	4926	6678
	Core Area (mm^2)	4.75	6.88	19.18	6.26	13.84	31.30
	Total Routing Length (mm)	230.07	315.08	1115.12	317.53	605.51	1881.6
	No. of Via	976	980	2524	1390	1370	3034
4-bit	τ (ns)	6.41	8.36	5.93	2.92	6.22	3.92
	Average Power (mw) at 20 MHz	0.134	1.83	2.015	1.08	2.15	1.79
	Max Power (mw)	51.24	237.93	159.31	66	197.04	123.98
	Leakage Power (nW)	4.16	11.86	5.37	0.38	12.40	3.99
	Transistor Count	880	838	1398	652	934	1302
	Core Area (mm^2)	1.19	1.93	4.75	0.98	2.05	4.04
	Total Routing Length (mm)	47.69	66.26	215.35	42.68	72.92	196.17
	No. of Via	249	250	657	237	260	593

Table 6.2 (b) Performance and characteristics of MUX based and 2×2 cell based multiplier for W/L=5

Size	Performance Indices	Multiplier Architecture Type					
		MUX based			2×2 Cell based		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	13.50	36.05	12.57	11.16	17.96	9.9
	Average Power (mw) at 20 MHz	57.158	63.80	40.42	52.14	107.74	68.93
	Max Power (mw)	1092.77	4729.06	3158.46	933.53	6047.99	4343.22
	Leakage Power (nW)	150.24	194.49	77.10	136.45	467.94	280.7
	Transistor Count	10196	8758	15678	16032	21744	29232
	Core Area (mm^2)	24.96	39.21	108.12	39.63	83.34	177.28
	Total Routing Length (mm)	1450.89	2048.8	7001.75	2297.2	4346.8	11514.2
	No. of Via	3443	3784	8217	5731	5700	12472
12-bit	τ (ns)	11.08	21.27	10.18	10.86	14.02	8.15
	Average Power (mw) at 20 MHz	18.69	32.77	23.75	27.06	56.22	37.19
	Max Power (mw)	616.16	2595.57	1942.58	534.69	3230.36	2372.7
	Leakage Power (nW)	88.82	119.33	42.88	65.34	254.76	29.40
	Transistor Count	6068	5286	9382	8668	11806	15918
	Core Area (mm^2)	13.30	19.18	55.34	18.53	37.50	87.18
	Total Routing Length (mm)	755.77	874	3348.87	1007.3	1790.62	5591.8
	No. of Via	1980	1979	4899	3105	3007	6706
8-bit	τ (ns)	8.69	11.88	7.57	5.47	12.45	5.17
	Average Power (mw) at 20 MHz	4.64	13.25	12.72	11.88	21.97	15.18
	Max Power (mw)	299.01	1259.30	936.23	259.47	1322.20	1004.45
	Leakage Power (nW)	42.59	60.20	11.97	24.68	128.04	14.52
	Transistor Count	2952	2646	4622	3588	4926	6678
	Core Area (mm^2)	4.89	8.24	23.85	6.93	14.69	32.00
	Total Routing Length (mm)	227.18	332.47	1342.11	344.73	608.54	1905.06
	No. of Via	958	981	2676	1380	1441	2979
4-bit	τ (ns)	6.30	8.12	5.59	3.09	6.05	4.15
	Average Power (mw) at 20 MHz	0.355	2.99	3.40	1.64	3.48	2.41
	Max Power (mw)	87.63	389.04	280.91	87.15	293.32	206.77
	Leakage Power (nW)	11.51	19.33	3.32	7.05	9.37	5.18
	Transistor Count	884	838	1398	652	934	1302
	Core Area (mm^2)	1.23	2.06	5.12	1.09	2.19	4.80
	Total Routing Length (mm)	50.74	65.08	224.15	44.69	73.17	209.71
	No. of Via	264	250	669	251	244	589

Table 6.2 (c) Performance and characteristics of MUX based and 2×2 cell based multiplier for W/L=7

Size	Performance Indices	Multiplier Architecture Type					
		MUX based			2×2 Cell based		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	13.25	36.51	12.12	10.77	18.01	9.86
	Average Power (mw) at 20 MHz	110.04	87.69	55.42	71.28	150.26	94.05
	Max Power (mw)	1528.38	6217.71	4361.28	1300.36	8082.12	6036.42
	Leakage Power (nW)	299.24	271.05	86.90	176.91	650.43	434.9
	Transistor Count	10196	8758	15678	16032	21744	29232
	Core Area (mm^2)	25.19	34.97	115.56	39.27	85.88	183.39
	Total Routing Length (mm)	1485.99	1786	7005.00	2176.5	4485.9	11294.4
	No. of Via	3416	3405	8071	5616	5594	12112
12-bit	τ (ns)	10.92	21.32	9.91	10.9	16.47	8.07
	Average Power (mw) at 20 MHz	36.08	45.49	32.92	36.96	77.90	50.49
	Max Power (mw)	862.07	3547.97	2618.94	751.54	4349.61	3305.44
	Leakage Power (nW)	178.12	166.76	52.68	97.41	352.11	49.5
	Transistor Count	6068	5286	9382	8668	11806	15918
	Core Area (mm^2)	13.66	19.75	65.01	19.78	41.48	99.13
	Total Routing Length (mm)	726.21	886	3702.14	1072.9	1863.56	5499.2
	No. of Via	1989	2007	4836	3108	3025	6693
8-bit	τ (ns)	8.60	11.72	7.53	5.52	12.01	5.22
	Average Power (mw) at 20 MHz	9.07	18.35	17.52	13.53	30.50	20.98
	Max Power (mw)	418.24	1734.28	1293.75	312.44	1804.79	1402.07
	Leakage Power (nW)	43.23	84.16	16.59	35.97	147.84	21.45
	Transistor Count	2952	2646	4622	3588	4926	6678
	Core Area (mm^2)	5.61	8.38	23.57	7.20	14.60	33.10
	Total Routing Length (mm)	260.4	347.84	1190.83	349.32	583.30	1754.2
	No. of Via	1054	1060	2551	1371	1422	3004
4-bit	τ (ns)	6.24	8.01	5.56	3.14	5.94	3.7
	Average Power (mw) at 20 MHz	0.686	4.15	4.73	2.15	4.84	3.92
	Max Power (mw)	122.50	539.66	388.48	110.15	428.63	286.50
	Leakage Power (nW)	23.31	26.97	7.43	10.42	25.97	6.40
	Transistor Count	884	838	1398	652	934	1302
	Core Area (mm^2)	1.32	2.10	5.86	1.10	2.35	5.41
	Total Routing Length (mm)	49.45	64.24	267.04	44.06	69.62	224.68
	No. of Via	250	250	680	228	244	590

6.3 Comparison Between MUX Based Barrel Shifter and Pereira’s Barrel Shifter Implementations

Tables 6.3 (a), 6.3 (b) and 6.3 (c) show the comparison of the cores of barrel shifter circuit using MUX based and Pereira’s design approaches.

Comparison of the two architectures given in tables 6.3 (a), 6.3 (b) and 6.3 (c) shows that MUX based barrel shifter architecture is faster as compared to Pereira’s barrel shifter, but transistor count, core layout area and routing length are much larger for MUX based barrel shifter architecture. This shows that MUX based barrel shifter architecture is inherently faster.

On the power count, MUX based architecture implemented using TG logic design style shows the lowest power consumption for bit widths of 4, 8 and 12. However, for the bit width of 16, Pereira’s architecture implemented using static logic features lower power consumption.

In MUX-based architecture the implementation using TG logic design style is faster as compared to static logic implementation due to decrease in routing length, but domino implementation is much slower as compared to static logic implementation due to increase in routing length and use of many simpler logic cells in the design.

Pereira’s architecture shows a comparable performance for static implementation and TG implementation, but performance is degraded in domino implementation due to increased routing length and use of many simpler logic cells in the design

Increasing the sizes of devices does not show appreciable improvement in delay and on the contrary shows larger core layout areas and increased power consumption.

Table 6.3 (a) Performance and characteristics of MUX based and Pereira's barrel shifter for W/L=3

Size	Performance Indices	Barrel shifter Architecture Type					
		MUX based			Pereira's		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	1.163	0.721	4.28	4.95	4.85	8.40
	Average Power (mw) at 20 MHz	7.18	2.27	22.26	1.328	1.428	5.72
	Max Power (mw)	65.87	276.26	1665.51	37.23	278.43	668.18
	Leakage Power (nW)	65.75	28.78	64.34	41.39	24.07	*2273*
	Transistor Count	12108	6054	18162	2042	2050	4588
	Core Area (mm^2)	28.34	23.38	106.44	4.91	5.88	16.66
	Total Routing Length (mm)	1901.42	1732.41	8053.4	266.7	292.21	988.9
	No. of Via	4316	4459	10192	1307	1225	2880
12-bit	τ (ns)	1.10	0.675	3.65	4.85	4.79	6.76
	Average Power (mw) at 20 MHz	0.61	0.76	12.60	1.076	1.416	4.35
	Max Power (mw)	53.94	208.57	1156.87	27.82	233.58	524.35
	Leakage Power (nW)	62.38	14.41	45.43	18.69	33.59	*1704*
	Transistor Count	9840	4920	13626	1610	1618	3580
	Core Area (mm^2)	20.19	17.21	64.244	3.59	4.62	12.09
	Total Routing Length (mm)	1297.00	1184.07	5083.4	175.26	225.73	743.08
	No. of Via	3324	3400	7360	898	936	2137
8-bit	τ (ns)	1.04	0.629	3.09	4.17	4.06	5.47
	Average Power (mw) at 20 MHz	0.37	0.22	4.66	0.537	0.841	2.25
	Max Power (mw)	34.00	125.90	408.85	18.51	127.35	280.55
	Leakage Power (nW)	14.96	8.51	17.75	5.81	4.43	1135
	Transistor Count	2988	1494	4482	880	884	1948
	Core Area (mm^2)	5.09	4.30	17.496	1.86	2.32	5.98
	Total Routing Length (mm)	322.09	296.57	1356.9	88.20	101.47	346.05
	No. of Via	1048	1092	2538	497	504	1144
4-bit	τ (ns)	0.51	0.209	2.35	3.18	3.09	4.38
	Average Power (mw) at 20 MHz	0.06	0.042	1.06	0.223	0.324	0.89
	Max Power (mw)	10.15	35.70	100.08	9.24	59.39	111.57
	Leakage Power (nW)	18.91	1.12	4.33	3.20	0.62	*567*
	Transistor Count	732	366	1098	364	355	802
	Core Area (mm^2)	1.09	0.94	3.35	0.652	0.76	2.035
	Total Routing Length (mm)	59.73	57.30	237.4	24.89	29.82	107.4
	No. of Via	255	266	635	175	168	446

Table 6.3 (b) Performance and characteristics of MUX based and Pereira's barrel shifter for W/L=5

Size	Performance Indices	Barrel shifter Architecture Type					
		MUX based			Pereira's		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	0.994	0.669	3.77	4.38	4.70	7.05
	Average Power (mw) at 20 MHz	13.25	3.58	35.25	1.64	2.37	8.95
	Max Power (mw)	106.96	420.15	2771.99	46.76	288.3	1148.71
	Leakage Power (nW)	329.34	33.56	107.92	58.27	59.04	*3835*
	Transistor Count	12108	6054	18162	2042	2050	4588
	Core Area (mm^2)	28.46	26.58	126.75	5.60	6.52	17.71
	Total Routing Length (mm)	2037.5	1820.6	9683.6	270.2	296.74	969.57
	No. of Via	4266	4554	10445	1212	1183	2924
12-bit	τ (ns)	1.00	0.609	3.70	4.35	4.53	6.45
	Average Power (mw) at 20 MHz	0.98	0.60	19.77	1.27	2.49	6.83
	Max Power (mw)	88.32	325.83	1920.2	35.27	373.89	895.66
	Leakage Power (nW)	1229.44	32.50	56.63	33.35	46.41	*2876*
	Transistor Count	9840	4920	13626	1610	1618	3580
	Core Area (mm^2)	20.58	18.71	81.37	4.44	4.87	12.91
	Total Routing Length (mm)	1303.7	1220.9	5875.05	218.10	222.39	718.98
	No. of Via	3324	3432	7274	920	941	2188
8-bit	τ (ns)	1.01	0.578	2.82	3.75	3.87	5.20
	Average Power (mw) at 20 MHz	0.60	0.38	7.55	0.645	1.37	3.62
	Max Power (mw)	55.95	192.27	679.04	23.43	224.27	445.31
	Leakage Power (nW)	478.23	14.61	22.67	1.89	25.90	1916
	Transistor Count	2988	1494	4482	880	884	1948
	Core Area (mm^2)	5.34	4.80	22.50	2.00	2.52	6.76
	Total Routing Length (mm)	320.41	314.03	1622.06	87.36	103.48	383.5
	No. of Via	1031	1097	2514	504	502	1152
4-bit	τ (ns)	0.517	0.205	2.28	2.89	3.14	3.98
	Average Power (mw) at 20 MHz	0.10	0.071	1.76	0.26	0.52	1.44
	Max Power (mw)	17.55	60.80	166.43	11.52	97.14	201.92
	Leakage Power (nW)	110.53	2.24	5.21	7.99	10.51	*957*
	Transistor Count	732	366	1098	364	355	802
	Core Area (mm^2)	1.11	1.10	4.65	0.67	0.84	2.25
	Total Routing Length (mm)	60.86	57.71	282.26	24.85	30.47	111.7
	No. of Via	246	275	623	173	167	451

Table 6.3 (c) Performance and characteristics of MUX based and Pereira's barrel shifter for W/L=7

Size	Performance Indices	Barrel shifter Architecture Type					
		MUX based			Pereira's		
		Static	TG	Domino	Static	TG	Domino
16-bit	τ (ns)	1.02	0.631	3.56	4.77	4.53	6.28
	Average Power (mw) at 20 MHz	19.37	4.84	48.13	2.3	3.31	12.15
	Max Power (mw)	148.48	559.31	3869.87	81.43	454.81	1644.20
	Leakage Power (nW)	1127.10	47.19	110.73	22.42	82.36	*5157*
	Transistor Count	12108	6054	18162	2042	2050	4588
	Core Area (mm^2)	29.58	26.49	128.29	5.94	6.86	19.36
	Total Routing Length (mm)	1963.67	1770.7	9465.7	287.3	320.36	1054.35
	No. of Via	4282	4480	10271	1222	1243	2927
12-bit	τ (ns)	0.98	0.598	3.45	4.45	4.45	6.13
	Average Power (mw) at 20 MHz	1.36	0.90	27.14	1.8	3.48	9.40
	Max Power (mw)	122.9	432.63	2679.3	57.79	528.19	1210.8
	Leakage Power (nW)	1184.99	41.17	64.73	47.15	67.09	*3868*
	Transistor Count	9840	4920	13626	1610	1618	3580
	Core Area (mm^2)	22.76	18.71	84.66	4.33	5.03	14.03
	Total Routing Length (mm)	1444.3	1200.5	5913.02	195.93	218.74	748.9
	No. of Via	3276	3386	7266	945	941	2170
8-bit	τ (ns)	0.948	0.541	2.75	3.66	3.81	5.00
	Average Power (mw) at 20 MHz	0.82	0.58	10.29	0.67	1.90	4.98
	Max Power (mw)	77.81	263.01	947.69	30.82	301.54	637.46
	Leakage Power (nW)	947.89	21.38	27.12	6.26	35.93	2578
	Transistor Count	2988	1494	4482	880	884	1948
	Core Area (mm^2)	5.83	4.85	22.98	2.16	2.49	7.10
	Total Routing Length (mm)	349.56	300.8	1619.51	96.43	104.33	391.9
	No. of Via	1032	1106	2554	488	472	1146
4-bit	τ (ns)	0.501	0.201	2.15	2.83	2.96	3.91
	Average Power (mw) at 20 MHz	0.15	0.095	2.41	0.36	0.73	2.00
	Max Power (mw)	24.40	77.92	232.25	19.04	133.45	296.10
	Leakage Power (nW)	117.23	3.16	7.10	9.29	34.66	*1288*
	Transistor Count	732	366	1098	364	355	802
	Core Area (mm^2)	1.22	1.14	4.82	0.72	0.88	2.56
	Total Routing Length (mm)	61.13	58.16	275.35	25.22	29.63	123.4
	No. of Via	248	276	629	157	156	453

6.4 Comparison of Different Barrel Shifter Architectures for TSPC Logic Design Style

Table 6.4 shows the comparison of different barrel shifter architectures implemented using TSPC logic design style. The TSPC logic design style shows pipelining behavior between two logic cells. The depth of circuit in terms of logic cells decides the number of clock cycles required to obtain correct output. In this circuit we may force the input every clock cycle, thereby improving the throughput of the circuit.

The performance parameters and attributes for comparison are number of clock cycles, average power, maximum power, leakage power, transistor count, core layout area, routing length and number of vias.

Table 6.4 shows the TSPC logic implementation for MUX based barrel shifter and Pereira's barrel shifter, which take almost equal clock cycles to generate correct output. The MUX based barrel shifter and Pereira's barrel shifter circuits are fully pipelined working at the clock speed of 500 MHz. The comparison shows that Pereira's implementation is better in terms of transistor count, core layout area, total routing length and number of vias.

Table 6.4 Performance and characteristics of MUX based and Pereira's barrel shifter for TSPC logic

Size	Performance Indices	Barrel shifter Architecture Type (TSPC Logic)					
		MUX based (500MHz)			Pereira's 1(500MHz)		
		W./L=3	W/L=5	W/L=7	W./L=3	W/L=5	W/L=7
16-bit	No of Cycles	7	7	7	6	6	6
	Average Power (mw) at 500 MHz	173.28	302.59	424.31	37.95	63.03	87.12
	Max Power (W)	2.66	4.38	6.09	0.528	0.861	1.194
	Leakage Power (uW) when clk=1	*10.43*	*18.04*	*29.00*	71	83.6	79.06
	Transistor Count	16144	16144	16144	2294	2294	2294
	Core Area (mm^2)	46.67	48.65	52.43	4.46	4.58	5.05
	Total Routing Length (mm)	3170.6	3232.8	3153.0	267.15	254.51	271.57
	No. of Via	5399	5431	5344	1096	1099	1107
12-bit	No of Cycles	7	7	7	6	6	6
	Average Power (mw) at 500 MHz	93.67	161.85	223.87	39.6	65.34	91.41
	Max Power (w)	1.94	3.11	4.36	0.4488	0.7128	1.039
	Leakage Power (uW) when clk=1	5.67	7.88	11.97	59.86	70.48	66.65
	Transistor Count	12112	12112	12112	1958	1958	1958
	Core Area (mm^2)	31.26	41.41	33.60	3.55	3.81	4.24
	Total Routing Length (mm)	1981.9	1905.8	1998.8	191.14	201.8	218.34
	No. of Via	3854	3802	3915	867	902	885
8-bit	No of Cycles	6	6	6	5	5	5
	Average Power (mw) at 500 MHz	49.09	84.72	117.63	21.05	34.55	48.21
	Max Power (W)	0.68	1.08	1.50	0.234	0.396	0.590
	Leakage Power (uW) when clk=1	4.00	6.64	9.90	34.80	40.97	38.75
	Transistor Count	3984	3984	3984	1091	1091	1091
	Core Area (mm^2)	8.55	9.62	9.81	1.70	1.87	2.02
	Total Routing Length (mm)	536.7	567.8	585.9	93.62	99.20	95.01
	No. of Via	1329	1327	1376	482	490	463
4-bit	No of Cycles	5	5	5	4	4	4
	Average Power (mw) at 500 MHz	13.63	23.72	32.76	10.78	17.65	24.58
	Max Power (W)	0.14	0.274	0.371	0.099	0.168	0.247
	Leakage Power (uW) when clk=1	1.91	2.10	3.14	14.85	17.48	16.53
	Transistor Count	976	976	976	481	481	481
	Core Area (mm^2)	1.89	1.98	2.15	0.63	0.698	0.772
	Total Routing Length (mm)	103.99	101.13	103.50	31.05	31.55	33.66
	No. of Via	327	329	324	188	198	184

6.5 Chapter Summary

This chapter presents the layout level implementation results of different architectures of chosen data path elements including signed multiplier (i.e Baugh Wooley multiplier and Booth encoded Wallace tree multiplier), unsigned multiplier (i.e. MUX based and 2×2 cell based multiplier) and barrel shifter architectures (i.e MUX based barrel shifter and Pereira's barrel shifter). The layout assemblies for the 4-bit, 8-bit, 12-bit and 16-bit multiplier and barrel shifter circuits were carried out using different high performance logic design styles and transistor sizes. The conclusions drawn are very useful for practicing designers since it describes the results after architectural exploration, logic design style exploration and transistor size exploration and physical design level exploration.

CHAPTER 7

CONCLUSION AND SCOPE OF FURTHER WORK

7.1 Conclusion

The thesis presents an exploratory study of the different high performance architectures for important data path elements including signed multipliers, unsigned multipliers and barrel shifters for 4-bit, 8-bit, 12-bit and 16-bit configurations. It also discusses the results of different VLSI logic design style based implementations of these architectures which include signed multiplier (i.e Baugh Wooley multiplier and Booth encoded Wallace tree multiplier), unsigned multiplier (i.e. MUX based and 2×2 cell based multiplier) and barrel shifter architectures (i.e MUX based barrel shifter and Pereira’s barrel shifter). The general comparison of performance and attributes like average power, maximum power, leakage power, transistor count, core layout area, routing length and number of vias for different architectures implemented using different logic design styles and different device sizes shows that:

- ❖ For the case of signed multipliers, for any operand size (4-bit, 8-bit, 12-bit and 16-bit) and for any logic design style (static, TG, domino) Baugh Wooley multiplier is significantly faster than the Booth encoded Wallace tree multiplier and consumes much less power due to fewer number of transistors required and a smaller core area. The Baugh Wooley multiplier implementation also shows smaller leakage power compared to Booth encoded Wallace tree multiplier due to its smaller transistor count.
- ❖ For the case of unsigned multipliers, for any operand size (4-bit, 8-bit, 12-bit and 16-bit) and for any logic design style (static, TG, domino) MUX based multiplier architecture is slower as compared to 2×2 cell based multiplier architecture, but consumes lesser power and features reduced transistor count, a smaller core area and reduced routing length.
- ❖ For the case of barrel shifters, for any operand size (4-bit, 8-bit, 12-bit and 16-bit) and for any logic design style (static, TG, domino) MUX based barrel shifter architecture is faster as compared to Pereira’s barrel shifter architecture; but the transistor count, core layout area and routing length are much larger for MUX based barrel shifter architecture as compared to Pereira’s architecture. This shows that MUX based barrel shifter architecture is inherently faster. On the power

count MUX based architecture implemented using TG logic design style shows the lowest power consumption for bit widths of 4, 8 and 12. However, for the bit width of 16, Pereira's architecture implemented using static logic design style features lower power consumption.

- The multiplier and barrel shifter VLSI implementations using TG logic design style may show speed advantage as compared to static logic design style implementation only when the total routing length is smaller or comparable with the static logic implementation. The average power, maximum power and leakage power for TG implementation are architecture dependent.
- The multiplier and barrel shifter VLSI implementations using dual rail domino logic design style may be faster compared to static logic implementation only when more complex logic cells are used in design and total routing length is not too large compared to static logic implementation. The improvement in speed may be obtained at the cost of increased average power consumption; hence the designer needs to be careful of this aspect.
- The multiplier and barrel shifter designs using simple dual rail domino logic cells show increased propagation delay due to one additional transistor in evaluate path and due to weak pull up transistor, which increases the contention current during evaluation. Another reason for increase in propagation delay for dual rail domino implementation using simple cells is the increased routing complexity and increased total routing length. Power consumption may also increases in simple domino gate based implementation due to higher switching activity than in equivalent static logic gate because all the domino nodes are pre-charged to V_{DD} during each clock cycle. The large total routing length also demands larger power consumption. The core layout area is more due to increased transistor count and increased total routing length.
- The static logic is most suitable for data path VLSI circuit implementation even for the circuits designed with simpler logic cells and longer critical paths.
- TSPC logic circuits give correct operation up to frequency of 500 MHz in our implementation but leakage power and average switching power is high because of much higher switching activity as compared to other CMOS logic design styles. Maximum power is also too large for TSPC circuits.
- Increasing the sizes of device does not show appreciable improvement in circuit delay and only features increased core area and power consumption.

Based on our research results a ready reckoner for selection of architecture and logic design style for high-speed signed multipliers is shown in table 7.1. Similar reckoners for unsigned multipliers and barrel shifters are shown in tables 7.2 and 7.3 respectively.

Table 7.1 Ready reckoner for high-speed signed multipliers

Attributes	Bit Width			
	4-bit	8-bit	12-bit	16-bit
Low power	Baugh Wooley (static)	Baugh Wooley (static)	Baugh Wooley (static)	Baugh Wooley (static)
High speed	Baugh Wooley (static)	Baugh Wooley (static)	Baugh Wooley (static)	Baugh Wooley (static)
Smallest area	Baugh Wooley (static)	Baugh Wooley (static)	Baugh Wooley (static)	Baugh Wooley (static)

Table 7.2 Ready reckoner for high-speed unsigned multipliers

Attributes	Bit Width			
	4-bit	8-bit	12-bit	16-bit
Low power	MUX based (static)	MUX based (static)	MUX based (static/domino)	MUX based (static/domino)
High speed	2×2 cell based (static)	2×2 cell based (static/Domino)	2×2 cell based (static/ domino)	2×2 cell based (domino)
Smallest area	2×2 cell based (static)	MUX based (static)	MUX based (static)	MUX based (static)

Table 7.3 Ready reckoner for high-speed barrel shifters

Attributes	Bit Width			
	4-bit	8-bit	12-bit	16-bit
Low power	MUX-based (TG)	MUX based (TG)	MUX based (static/TG)	Pereira's (static)
High speed	MUX-based (TG)	MUX-based (TG)	MUX-based (TG)	MUX-based (TG)
Smallest area	Pereira's (static)	Pereira's (static)	Pereira's (static)	Pereira's (static)

For the NBTI, which is identified as one of the most critical reliability concerns for nanometer scale digital integrated circuits, we have presented a Verilog HDL based general technique to model the circuit performance degradation due to NBTI. As an example we have presented the NBTI degradation study for a 1-bit full adder circuit using Verilog HDL. The degradation in V_t of all PMOS devices in the circuit can be obtained using Verilog HDL switch level modeling. Simulation result shows differing NBTI degradation in the Vt of different PMOS devices in a 1-bit full adder. Techniques to compute consequent degradation in circuit switching speed due to NBTI have also been indicated.

7.2 Scope of Further Work

An extensive design space exploration is necessary to meet performance, power and area trade-offs for data path elements such as signed/unsigned multipliers and barrel shifters. Based on our study a CAD tool can be developed for signed/unsigned multiplier and barrel shifter which will accept operand size and architecture type as an input and generates a cell/gate level HDL net-list in terms of predefined basic cells/gates. Such cell/gate level net-list then can be fed to commercial synthesis tools offering RTL to GDSII flow to generate VLSI implementation corresponding to the selected architecture type and selected operand size. Such a tool will be very useful for practicing designers in order to select an appropriate implementation out of many possible implementation options in view of the requirements of performance, power and area for any technology node, and across technology nodes.

While the thesis has studied only the multiplier and barrel shifter circuits in a systematic manner to understand optimal design approaches for high performance design of these blocks, similar studies need to be carried out for other elements such as divider circuit, special function units computing trigonometric functions, statistical functions, mathematical functions and other specialized functions required in a host of application environments. Based on such studies, a similar CAD tools for the optimal synthesis of these functions/blocks can be developed. Also NBTI degradation effect can be incorporated in the evaluation of designs. Such tools can add a great deal of value to the hardware software co-design approaches required in embedded real time system developments and research in re-configurable computing systems being carried out globally at this point of time.

REFERENCES

[1] Sung-Mo Kang and Yusuf Leblebici, “CMOS digital integrated circuits,” Tata McGraw-Hill Publishers, New Delhi, Third Edition, 2003.

[2] Neil H.E. Weste and K. Eshraghian, “Priniciples of CMOS VLSI design,” Tata McGraw-Hill Publishers, New Delhi, Second Edition, 1998.

[3] Rabaey, J. M, Anantha Chandrakasan and Borivoje Nikolic, “Digital integrated circuits,” Prentice Hall of India Pvt Ltd, 1996.

[4] Huy T. Nguyen and Abhijit Chatterjee, “Number-splitting with shift and add decomposition for power and hardware optimization in linear DSP synthesis,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 8, NO. 4, pp. 419-424, August 2000.

[5] John P. Uyemura, “Introduction to VLSI circuits and systems,” John Wiley and Sons, Inc. 2002.

[6] Russell Henning and Chaitali Chakrabarti, “An approach to switching activity consideration during high level, low power design space exploration,” IEEE Transactions on Circuits and Systems-II, Analog and Digital Signal Processing, VOL. 49, NO. 5, pp. 399-351, May 2002.

[7] David I Cheng, Kwang-Ting Cheng, Deborah C. Wang and Malgorzata Marek-Sadowska, “A hybrid methodology for switching activities estimation,” IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, VOL.17, NO. 4, pp. 357-365, April 1998.

[8] D.A, Pucknell and K. Eshraghian, “Basic VLSI design system and circuits,” Prentice Hall of India Pvt Ltd, 1994.

[9] Mohab Anis, Mohamed Allam and Mohamed Elmasry, “Impact of technology scaling on CMOS logic styles,” IEEE Transactions on Circuits and Systems-II, Analog and Digital Signal Processing, VOL. 49, NO. 8, pp. 577-587, August 2002.

[10] V. Chanramouli, Erik Brunvand and Kent F. Smith, “Self-timed design in GaAs-case study on a high-speed, parallel multiplier,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 4, NO. 1, pp. 146-149, March 1996.

[11] Adel S. Sedra and Kenneth C. Smith, “Microelectronic circuits,” Oxford University Press, New York, Fifth Edition, 2004.

[12] W. Qinghong, C.Y. Roger and Bradely S. Carlson, “LILA: Layout generation for iterative logic arrays,” IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, VOL. 14, NO. 11, pp. 1359-1369, November 1995.

[13] L. Wanhammar, “DSP integrated circuits,” Addison–Wesley, Wokingham, England, pp. 475-482, 1999.

[14] P. Kornerup, “A systolic, linear-array multiplier for a class of right-shift algorithms,” IEEE Transactions on Computers, VOL. 43, NO. 8, pp. 892-898, August 1994.

[15] L. Ciminiera and Paolo Montuschi, “Carry-save multiplication schemes without final addition,” IEEE Transactions on Computers, VOL. 45, NO. 9, pp. 1050-1055, September 1996.

[16] Kiamal Z. Pekmestzi, “Multiplexer-based array multipliers,” IEEE Transactions on Computers, VOL. 48, NO.1, pp. 15-23, January 1999.

[17] Shivaling S. Mahant Shetti and Poras T. Balsara, “High performance low power multiplier using temporal tiling,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 7, NO. 1, pp. 121-124, March 1999.

[18] Uwe Sparmann and Sudhakar M. Reddy, “On the effectiveness of residue code checking for parallel two’s complement multipliers,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 4, NO. 2, pp. 227-239, June 1996.

[19] Shyue Kung, Lu, Jen-Cuan Wang and Cheng-Wem Wu, “C-testable design techniques for iterative logic array,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 3, NO. 1, pp. 146-152, March 1995.

[20] W. Zhongde, Graham A. Jullien and William C. Miller, “A new design technique for column compression multipliers,” IEEE Transactions on Computers, VOL. 44, NO. 8, pp. 962-970, August 1995.

[21] Hesham A., Al-Twaijry and Michael J. Flynn, “Technology scaling effects on multipliers,” IEEE Transactions on Computers, VOL. 47, NO. 11, pp. 1201-1215, November 1998.

[22] Jalil Fadavi Ardekani, “M*N Booth encoded multiplier generator using optimized Wallace trees,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 1, NO. 2, pp. 120-125, June 1993.

[23] Vojin G. Okhlobdzija and David Villeger, “Improving multiplier design by using improved column compression tree and optimized final adder in CMOS

technology," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 3, NO. 2, pp. 292-301, June 1995.

[24] Behrooz Parhami, "High speed area efficient multiplier design using multiple valued current mode circuits," IEEE Transactions on Computers, VOL. 45, NO. 5, pp. 637-638, May 1996.

[25] Shoji Kawahito, Makoto Ishida, Tetsuro Nakamura, Michitaka Kameyama and Tatsuo Higuchi, "High speed area efficient multiplier design using multiple valued current-mode circuits," IEEE Transactions on Computers, VOL. 43, NO. 1, pp. 34-42, January 1994.

[26] Dinesh Somasekhar and V. Visvanathan, "A 230-MHz half-bit level pipelined multiplier using true single-phase clocking," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 1, NO. 4, pp. 415-422, December 1993.

[27] Debabrata Ghosh, "Design and realization of high-performance wave-pipelined 8×8 bit multiplier in CMOS technology," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, VOL. 3, NO. 1, pp. 36-48, March 1995.

[28] Pascal C. H. Meier, Rob A. Rutenbar and L. Richard Carley, "Exploring multiplier architecture and layout for low power," Custom Integrated Circuits Conference, Sandiago, CA, USA, Proceedings of IEEE, 1996.

[29] R. Pereira, J.A. Michell and J.M. Solana, "Fully pipelined TSPC barrel shifter for high speed applications," IEEE Journal of Solid State Circuits, VOL. 30, NO.6, pp. 686-690, June 1995.

[30] G. M. Tharkan and S.M.Kang, "A new design of a fast barrel switch network," IEEE Journal of Solid State Circuits, VOL. 27, NO. 2, pp. 217-221, February 1992.

[31] B. Zhu, J.S. Suehle, J.B. Bernstein and Y. Chen, "Mechanism of dynamic NBTI of pMOSFETs," IRW Final Report, 2004.

[32] S. Aota, S. Fuji, Z. W. Jin, Y. Ito, K. Utsumi, E. Morifuji, S. Yamada, F. Matsuoka, and T. Noguchi, "A new method for precise evaluation of dynamic recovery of negative bias temperature instability," Proceedings of lEEE International Conference on Microelectronic Test Structures, VOL. 18, April 2005.

[33] Rakesh Vattikonda, Wenping Wang and Yu Cao, "Modeling and minimazation of PMOS NBTI effect for robust nanometer design," Design Automation Conference, July 2006.

[34] Sarvesh Bhardwaj, Wenping Wang, Rakesh Vottikonda, Yu Cao and Sarma Vrudhula, “Predictive modeling effect of the NBTI effect for reliable design,” Custom Integrated Circuits Conference, Sandiago, CA, USA, Proceedings of IEEE, September 1996.

[35] Tanner Tools, L-Edit^{TM} User’s Manual version 6, Tanner Research, Inc., 1996.

[36] Tanner Tools, T-Spice Pro^{TM} User’s Manual version 4, Tanner Research, Inc., 1996.

[37] Paul Chauduri, “Computer organisation and design,” Prentice Hall of India Private Limited, 1994.

[38] C.W Chiou, L.C. Fin, F.H.Chou and S.F.Shu, “Low complexity finite field multipliers using irreducible trinomials,” Electronics Letters, VOL. 39, NO. 24, November 2003.

[39] John D. Carpinelli, “Computer systems organization & architecture,” Pearson Education Asia, 2001.

[40] Gensuke Goto, Tamio Sato, Masso Nakajima and Takao Sukemura, “A 54*54 regularly structurd tree multiplier,” IEEE Journal of Solid State Circuits, VOL. 27, NO. 9, pp. 1229-1235, September 1992.

[41] Gensuke Goto, Atsuki Inoue, Ryoichi Ohe, Shoichiro Kashiwakura, Shin Mitarai, Takayuki Tsuru and Tetsuo Izawa, “A 4.1-nS compact 54*54-b multiplier utilizing sign-select Booth encoders,” IEEE Journal of Solid State Circuits, VOL. 32, NO.11, pp. 1676-1681, November 1997.

[42] Jerry D. Daniels, “Digital design from zero to one,” John Wiley and Sons, Inc. 1996.

[43] Richard F. Tinder, “Engineering digital design,” Academic Press, 2001.

[44] Ingvar Aberg, Cait Ní Chleirigh, and Judy L. Hoyt, “Ultra-thin body strained Si and SiGe heterostructure on insulator MOSFETs,” IEEE Transactions on Electron Devices, VOL. 53, NO. 5, pp. 1021-1029, May 2006.

[45] D. A. Antoniadis, I. Aberg, C. Nı´ Chleirigh, O. M. Nayfeh, A. Khakifirooz and J. L. Hoyt, “Continuous MOSFET performance increase with device scaling: The role of strain and channel material innovations,” IBM Journal of Research & Development. VOL. 50, NO. 4, pp. 363-376, July 2006.

[46] Indranil De and Carlton M. Osburn, “Impact of super-steep-retrograde channel doping profiles on the performance of scaled devices,” IEEE Transactions on Electron Devices, VOL. 46, NO. 8, pp. 1711-1717, August 1999.

[47] Min Zhao and Sachin S. Sapatnekar, “Dual monotonic domino gate mapping and optimal output phase assignment of domino logic,” IEEE International Symposium on Circuits and Systems, VOL. 2, pp. 309-312, 2000.

[48] P. Srivastava, A. Pua and L. Welch, "Issues in the design of domino logic circuits," Great Lakes Symposium on VLSI, pp 108-113, 1998.

[49] David Harris and M.A. Horowitz, “Skew tolerant domino circuits,” IEEE Journal of Solid State Circuits, VOL. 32, NO. 11, pp. 1702-1711, November 1997.

[50] Shrirang K. Karandikar and Sachin S. Sapatnekar, “Technology maping for SOI domino logic incorporating solutions for the parasitic bipolar effect,” IEEE Transactions on Very Large Scale Integration (VLSI) VOL. 11, NO. 6. December 2003.

[51] Shang-Jyh Shieh and Jinn-Shyam Wang, “Design of low power domino circuits using multiple supply voltages,” IEEE International Conference on Electronics, Circuits and Systems, VOL. 2, pp. 711-714, 2001.

[52] Xiangning Yang, Eric Weglarz and Kewal Saluja, “On NBTI degradation process in digital logic circuits,” 20th International Conference on VLSI Design, 2007.

[53] M.A. Alam and S. Mahapatra, “A comprehensive Model for PMOS NBTI degradation,” Elsevier Journal of Microelectronics Reliability, pp 71-81, 2005.

[54] Sanjay V. Kumar, Chris H. Kim and Sachin S. Sapatnekar “NBTI aware synthesis of digital circuits,” Design Automation Conference 2007, San Diego, CA, USA, June 2007.

[55] Xiangning Yang and Kewal Saluja, “Combating NBTI degradation via gate sizing,” Proceedings of the 8th International Symposium on Quality Electronic Design, 2007.

[56] Takayasu Sakurai and A. Richard Newton, “Alpha-power law MOSFET model and its applications to CMOS inverter and other formulas,” IEEE Journal of Solid State Circuits, VOL. 25, pp 584-594, April 1990.

[57] Bipul C. Paul, Kunhyuk Kang, Haldun Kufluoglu and Muhammad A. Alam, “Impact of NBTI on the temporal performance degradation of digital circuits,” IEEE Electron Device Letters, VOL. 26, NO. 8, pp- 560-562, August 2005.

APPENDIX A1

TABLE A1.1: BASIC CELLS USED FOR DIFFERENT MULTIPLIERS AND BARREL SHIFTERS FOR DIFFERENT LOGIC DESIGN STYLES

Logic design style	***STANDARD/BASIC CELLLS***					
	BAUGH WOOLEY MULTIPLIER	**BOOTH ENCODED WALLACE TREE MULTIPLIER**	**MUX-BASED MULTIPLIER**	**2×2 CELL BASED MULTIPLIER**	**MUX BASED BARREL SHIFTER**	**PAREIRAS BARREL SHIFTER**
Static CMOS	INV_static AND2_static FA_static	MUX2-1_static_ABCHG INV_static NAND2_static NOR2_static XOR2_static OR2_static AND2_static	MUX2-1_static XOR2_static OR3_static OR2_static FA_static AND2_static AND3_static	INV_static NAND2_static NOR2_static FA_static	MUX2-1_static	XOR2_static OR3_static OR2_static INV_static AND2_static AND3_static

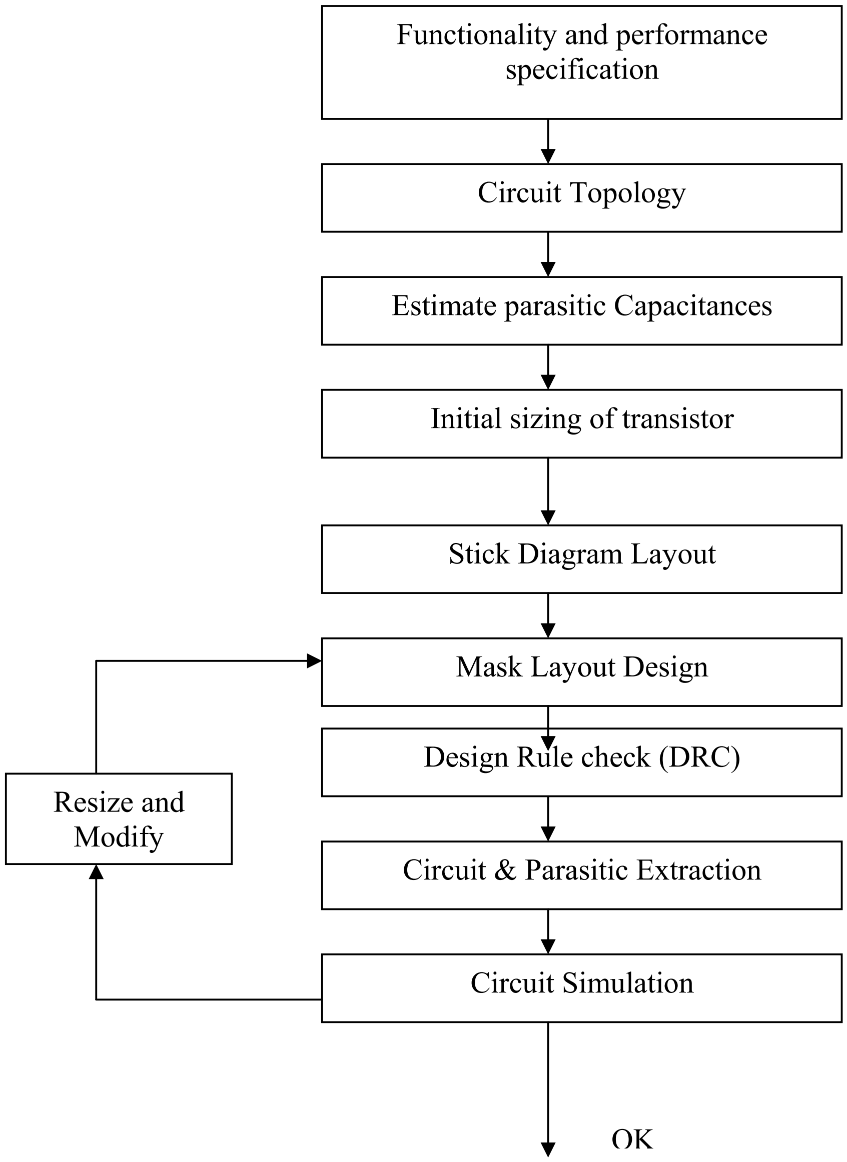
TG	INV_static AND2_TG FA_TG	MUX2-1_TG_ ABCHG INV_static NAND2_TG NOR2_TG XOR2_TG OR2_TG AND2_TG	MUX2-1_TG XOR2_TG OR3_TG OR2_TG FA_TG AND2_TG AND3_TG	INV_static NAND2_TG NOR2_TG FA_TG	MUX2-1_TG	XOR2_TG OR3_TG OR2_TG INV_static AND2_TG AND3_TG
DOMINO	AND2_DOMINO FA_DOMINO	MUX2-1_DOMINO_ ABCHG XOR2_DOMINO OR2_DOMINO AND2_DOMINO	MUX2-1_DOMINO XOR2_DOMINO OR3_DOMINO OR2_DOMINO FA_DOMINO AND2_DOMINO AND3_DOMINO	AND2_DOMINO OR2_DOMINO FA_DOMINO	MUX2- 1_DOMINO	XOR2_DOMINO OR3_DOMINO OR2_DOMINO AND2_DOMINO AND3_DOMINO

TSPC					MUX2-1_TSPC	AND3_TSPC AND_OR_TSPC BASICMODULE_TSPC INV_Static INV_BUF_TSPC LATCHINV_TSPC LATCH_TSPC OR3_TSPC OR2INV_TSPC NAND2_TSPC NOR2_TSPC NOT_TSPC XOR2_Static AND2INV_TSPC OR_AND_TSPC

APPENDIX A2

A2.1 LAYOUT DESIGN FLOW

The design of physical layout is very tightly linked to overall circuit performance (area, speed, power dissipation) since the physical structure directly determines the trans-conductance of the transistors, the parasitic capacitances and resistance and obviously, the silicon area, which is used for a certain function.


Functionality and performance specification
Circuit Topology
Estimate parasitic Capacitances
Initial sizing of transistor
Stick Diagram Layout
Mask Layout Design
Design Rule check (DRC)
Circuit & Parasitic Extraction
Circuit Simulation
Resize and Modify
OK

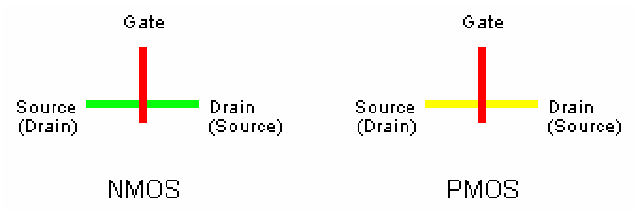
The initial phase of layout design can be simplified significantly by the use of stick diagrams or symbolic layouts. Then the actual layout is done using the design rules.

A2.2 STICK DIAGRAMS

Stick level in VLSI Design is an abstraction between transistor schematic level and layout level. Stick diagrams assist in planning for layout drawing quickly and easily. They need not be to scale. Also designing complete layout in terms of rectangles can be overwhelming so first we will draw stick diagrams. It is a representation of the layout. In stick diagram we capture layer topology of layout proposition through lines (sticks) of different colors. It is a metric free notation and therefore does not show exact placement, transistor sizes, wire lengths, wire widths and tub boundaries.

(a) Transistors

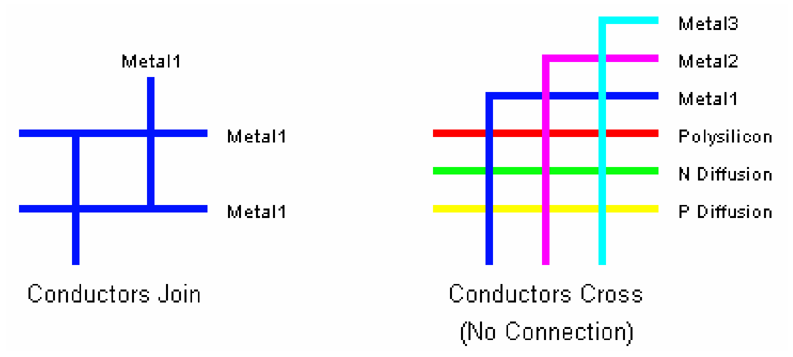
A transistor exists where a poly-silicon stick crosses either an N diffusion stick (NMOS transistor) or a P diffusion stick (PMOS transistor).



Note that there is no difference in the construction of a transistor source and a transistor drain. The source is determined as the source of conductors (electrons for NMOS / holes for PMOS) when current flows through the channel. In some pass transistor circuits, the source and drain may swap over during use.

(b) Implied Connections and Crossovers:

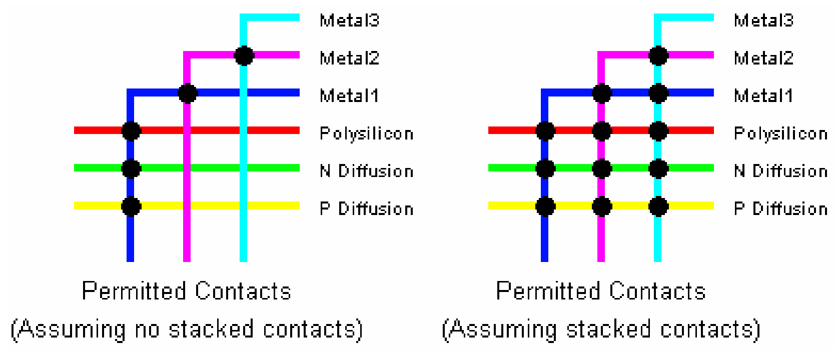
When two sticks of the same color meet or cross there is always a connection between them. When two sticks of different colors meet or cross there is no implied connection between them.



The N and P diffusions may not cross each other. Where poly silicon crosses diffusion a transistor is formed.

(c) Contact cut or Via:

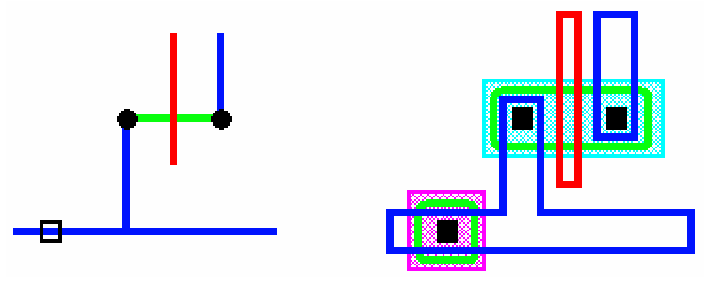
A connection may be explicitly defined using a filled black circle. A connection is allowed when just one layer of insulator separates mask layers. This connection is defined as a "contact cut". Thus P diffusion may connect to Metal1 but not directly to Metal2.



In a process if “**stacked contacts”** are permitted then we may draw a contact between non-adjacent conductors; e.g. between Poly and Metal3, in which case the connection to intermediate layers (Metal1 and Metal2) is implied.

(d) Taps

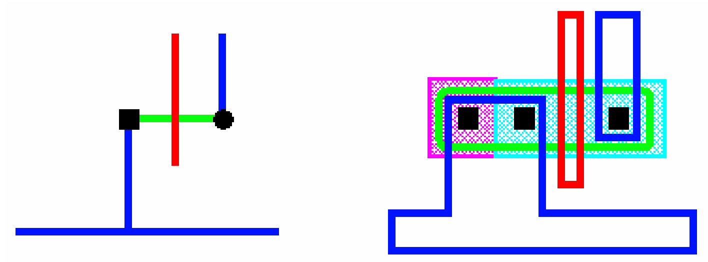
The tap represents a connection to N-well or P-well. A tap is defined using an unfilled black square. Here there will be only one conductor crossing the square (Metal1 power or ground rail).



If connection is from a power rail then it is N-Well Tap, otherwise if the connection is from a ground rail then it is a Substrate Tap.

(e) Combined Contacts & Taps

We can often save space by using a combined contact and tap. Here the tap shares the same Active Area as the contact. A combined contact and tap is defined using a filled black square instead of a filled black circle as shown for the source contact.

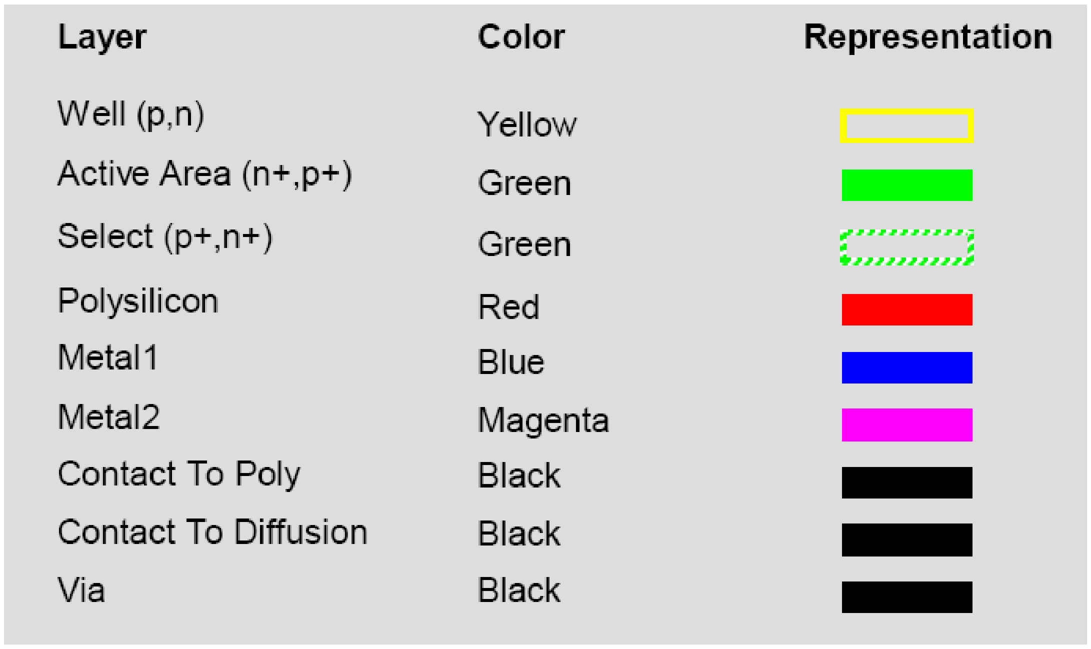


A combined contact and tap can only be used where the end of a diffusion stick coincides with a contact to the power or ground rail.

(f) Stick Diagram Color Code

P diffusion	:	Yellow/Brown
Metal1	:	Blue
N diffusion	:	Green
Metal2	:	Magenta/Purple/ Gray
Polysilicon	:	Red
Metal3	:	Cyan/Light Blue
Contacts & Taps	:	Black

A2.3 Layout Layer Representation

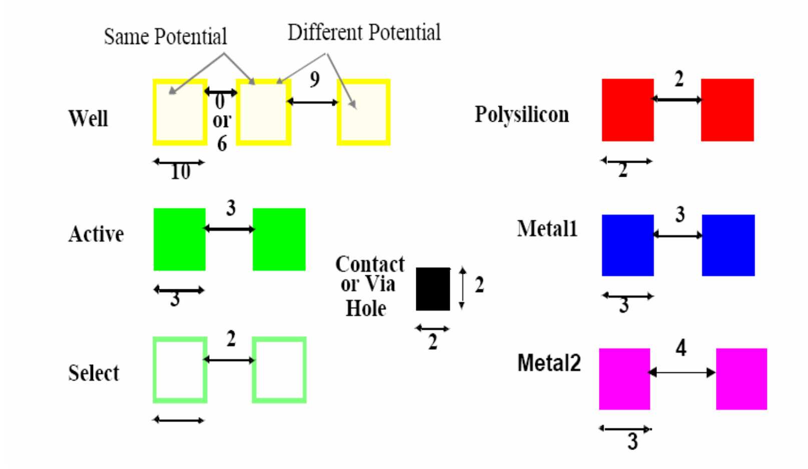


Layer	Color	Representation
Well (p,n)	Yellow	🟨
Active Area (n+,p+)	Green	🟩
Select (p+,n+)	Green	⬚
Polysilicon	Red	🟥
Metal1	Blue	🟦
Metal2	Magenta	🟪
Contact To Poly	Black	⬛
Contact To Diffusion	Black	⬛
Via	Black	⬛

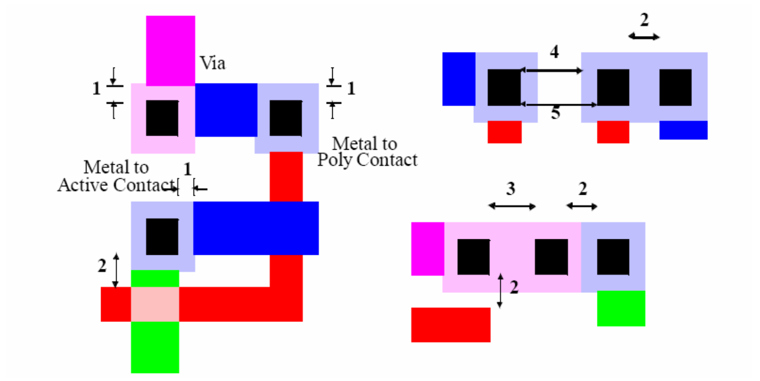
A2.4 DESIGN RULES

The physical mask layout of any circuit to be manufactured using a particular process must conform to a set of geometric constraints , which are generally called layout design rules. These rules usually specify the minimum allowable line widths for physical objects on-chip such as metal and polysilicon interconnects or diffusion areas, minimum feature dimensions, and minimum allowable separations between two such features

(i) Intra-Layer Design Rules



(ii)Via's and Contacts



(iii) Table of design rules:

Table A2.1 Design rules

	Rules	Size
1	**Active area rules** ❖ Minimum active area width ❖ Minimum active area spacing	3λ 3λ
2	**Polysilicon rules** ❖ Minimum polywidth ❖ Minimum gate extension of poly over active ❖ poly-active edge spacing (Poly outside active area) ❖ Minimum poly-active edge spacing(poly inside active area)	2λ 2 λ 2λ 2λ
3	**Metal rules** ❖ Minimum metal width ❖ Minimum metal spacing	3λ 3λ
4	**Contact rules** ❖ Poly contact size ❖ Minimum poly contact spacing ❖ Minimum poly contact to poly edge spacing ❖ Minimum poly contact to metal edge spacing ❖ Minimum poly contact to active edge spacing ❖ Active contact size ❖ Minimum active contact spacing (on the same active region)	2λ 2λ 1λ 1λ 3λ 2λ 2λ

	❖ Minimum active contact to active edge spacing ❖ Minimum active contact to metal edge spacing ❖ Minimum active contact to poly edge spacing ❖ Minimum active contact spacing (on different active regions)	1λ 1λ 3λ 6λ

A2.5 Cell Heights for Different Cells

The heights of cells for different logic design styles and different W/L size of transistors are presented in table A2.2. These heights are in terms of λ. Where λ=0.3 μm.

Table A2.2

Name	W/L=3	W/L=5	W/L=7
Static CMOS	73.5	75	85.5
TG	75	85	95
Domino	78	96	107
TSPC	76	85	99

APPENDIX A3

A3.1 HP05.md (Model Parameter File)

```
*******************************************************************
*                   MOSIS PARAMETRIC TEST RESULTS
*
*       RUN: N5BO                          VENDOR: HP-NID
*    TECHNOLOGY: SCN05H                    FEATURE SIZE: 0.5 microns
*
*
*INTRODUCTION: This report contains the lot average results obtained by MOSIS from
*measurements of MOSIS test structures on each wafer of this fabrication lot.
* SPICE parameters obtained from similar measurements on a selected wafer are also attached.
*
*COMMENTS: Hewlett Packard CMOS14TB.
*
*
*TRANSISTOR PARAMETERS     W/L      N-CHANNEL    P-CHANNEL     UNITS
*
* MINIMUM                 0.9/0.60
*  Vth                                  0.68         -0.90        Volts
*
* SHORT                   15/0.60
*  Vth                                  0.61         -0.88        Volts
*  Vpt                                  11.4          -9.4        Volts
*  Vbkd                                 11.4          -9.5        Volts
*  Idss                                 396          -188          uA/um
*
* WIDE                    15/0.60
*  Ids0                                 10.5         1.6          pA
*
* LARGE                    5.4/5.4
*  Vth                                  0.69         -0.95        Volts
*  Vjbkd                                11.5          -10.1       Volts
*  Ijlk                                -19.2           8.1         pA
*  Gamma                                 0.60          0.49        V^0.5
*
* Delta length                          0.14         0.09         microns
*  (L_eff = L_drawn-DL)
* Delta width                           0.44         0.32          microns
*  (W_eff = W_drawn-DW)
* K' (Uo*Cox/2)                          72.1        -22.0        uA/V^2
*
*COMMENTS: Delta L varies with design technology as a result of the different
*        mask biases applied for each technology. Please adjust the delta L
*        in this report to reflect the actual design technology of your
*        submission.
*             Design Technology                        Delta L
*             -----------------        -------
*             SCN_SUBM (lambda=0.3), CMOSH,
*              HP_CMOS14TB                            no adjustment
*             SCN (lambda=0.35)                        add 0.1 um
*
*
*FOX TRANSISTORS       GATE          N+ACTIVE            P+ACTIVE        UNITS
* Vth                      Poly          >15.0                 <-15.0          Volts
*
```

```
*
*
*
*PROCESS PARAMETERS     N+DIFF  P+DIFF   POLY  METAL1  METAL2  METAL3  UNITS
* Sheet Resistance         2.1      2.0       1.9     0.07      0.07      0.03    ohms/sq
* Width Variation         -0.36   -0.29      -0.04    0.16     -0.04     -0.30    microns
*  (measured - drawn)
* Contact Resistance       2.3     2.2        2.2                0.82      0.87     ohms
* Gate Oxide Thickness     94                                                     angstroms
*
*
*CAPACITANCE PARAMETERS  N+DIFF  P+DIFF   POLY  METAL1  METAL2   METAL3  UNITS
* Area (substrate)           546     929      92     47       15        11    aF/um^2
* Area (poly)                                        59       18        11    aF/um^2
* Area (metal1)                                               37        14    aF/um^2
* Area (metal2)                                                         33    aF/um^2
* Area (N+active)                           3684                              aF/um^2
* Area (P+active)                           3500                              aF/um^2
* Fringe (substrate)         195     234                                      aF/um
* Fringe (N+active)                          105                              aF/um
********************************************************************************
*
*  N5BO SPICE LEVEL3 PARAMETERS
*
********************************************************************************

.MODEL NMOS NMOS LEVEL=3 PHI=0.700000 TOX=9.6000E-09 XJ=0.200000U TPG=1
+ VTO=0.6566 DELTA=6.9100E-01 LD=4.7290E-08 KP=1.9647E-04
+ UO=546.2 THETA=2.6840E-01 RSH=3.5120E+01 GAMMA=0.5976
+ NSUB=1.3920E+17 NFS=5.9090E+11 VMAX=2.0080E+05 ETA=3.7180E-02
+ KAPPA=2.8980E-02 CGDO=3.0515E-10 CGSO=3.0515E-10
+ CGBO=4.0239E-10 CJ=5.62E-04 MJ=0.559 CJSW=5.00E-11
+ MJSW=0.521 PB=0.99
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 4.1080E-07
.MODEL PMOS PMOS LEVEL=3 PHI=0.700000 TOX=9.6000E-09 XJ=0.200000U TPG=-1
+ VTO=-0.9213 DELTA=2.8750E-01 LD=3.5070E-08 KP=4.8740E-05
+ UO=135.5 THETA=1.8070E-01 RSH=1.1000E-01 GAMMA=0.4673
+ NSUB=8.5120E+16 NFS=6.5000E+11 VMAX=2.5420E+05 ETA=2.4500E-02
+ KAPPA=7.9580E+00 CGDO=2.3922E-10 CGSO=2.3922E-10
+ CGBO=3.7579E-10 CJ=9.35E-04 MJ=0.468 CJSW=2.89E-10
+ MJSW=0.505 PB=0.99
* Weff = Wdrawn - Delta_W
* The suggested Delta_W is 3.6220E-07
```

A3.2 mHP_nS5.ext (Layout Parasitic Extraction Definition File)

```
# File:          mHP_nS5.ext
# For:           Extract definition file
# Vendor:          MOSIS/HP
# Technology:     0.5u (Lambda = 0.30um) / N-well (SCN3M_SUBM) Sub-Micron
# Technology File: mHP_nS5.tdb
# Copyright © 1991-2001 Tanner EDA, A Division of Tanner Research, Inc.
# All Rights Reserved
#
# This file will work only with L-EDIT Version 7 and greater.
# ***************************************************************************

connect(n well wire, ndiff, ndiff)
connect(subs, pdiff, pdiff)
connect(allsubs, subs, subs)
connect(ndiff, Metal1, Active Contact)
connect(pdiff, Metal1, Active Contact)
connect(poly wire, Metal1, Poly Contact)
connect(Metal1, Metal2, Via1)
connect(Metal2, Metal3, Via2)
connect(LPNP emitter, pdiff, LPNP emitter)
connect(LPNP collector, pdiff, LPNP collector)

# NMOS transistor with poly1 gate
device = MOSFET(
            RLAYER=ntran;
            Drain=ndiff, AREA, PERIMETER;
            Gate=poly wire;
            Source=ndiff, AREA, PERIMETER;
            Bulk=subs;
            MODEL=NMOS;
            )

# PMOS transistor with poly1 gate
device = MOSFET(
            RLAYER=ptran;
            Drain=pdiff, AREA, PERIMETER;
            Gate=poly wire;
            Source=pdiff, AREA, PERIMETER;
            Bulk=n well wire;
            MODEL=PMOS;
            )

# PNP transistor
device = BJT(
          RLAYER=LPNP ID, AREA;
          Collector=LPNP collector;
          Base=n well wire ;
          Emitter=LPNP emitter;
          Substrate=allsubs;
          MODEL=PNP;
          NominalArea = 1.0;
          )

# Linear capacitor using Cap-Well
device = CAP(
          RLAYER=Cap-Well Capacitor, AREA;
          Plus=poly wire;
          Minus=ndiff;
          MODEL=;
          )

# NMOS capacitor
device = CAP(
          RLAYER=NMOS Capacitor, AREA;
          Plus=poly wire;
          Minus=ndiff;
          MODEL=;
          )

# PMOS capacitor
device = CAP(
          RLAYER=PMOS Capacitor, AREA;
          Plus=poly wire;
          Minus=pdiff;
          MODEL=;
          )

# Poly resistor
device = RES(
          RLAYER=Poly Resistor;
          Plus=poly wire;
          Minus=poly wire;
          MODEL=;
          )

# N Diffusion resistor
device = RES(
          RLAYER=N Diff Resistor;
          Plus=ndiff;
          Minus=ndiff;
          MODEL=;
          )

# P Diffusion resistor
device = RES(
          RLAYER=P Diff Resistor;
          Plus=pdiff;
          Minus=pdiff;
          MODEL=;
          )

# N Well resistor
device = RES(
          RLAYER=N Well Resistor;
          Plus=n well wire;
          Minus=n well wire;
          MODEL=;
          )

# Bonding Area Capacitance
 device = CAP(
           RLAYER=Pad Comment, AREA;
           Plus=Metal1;
           Minus=allsubs;
           MODEL=;
           )

# Diodes
device = DIODE(
            RLAYER=diode pdiff, AREA;
            Plus=pdiff;
            Minus=n well wire;
            MODEL=Dpdiff;
            NominalArea = 1.0;
            ) IGNORE_SHORTS

device = DIODE(
            RLAYER=diode ndiff, AREA;
            Plus=subs;
            Minus=ndiff;
            MODEL=Dndiff;
            NominalArea = 1.0;
            ) IGNORE_SHORTS

# Lateral Diode
device = DIODE(
            RLAYER=diode_lat, AREA;
            Plus=pdiff;
            Minus=ndiff;
            MODEL=D_lateral;
            NominalArea = 1.0;
            ) IGNORE_SHORTS
```

BRIEF BIOGRAPHY OF CANDIDATE

Abhijit R. Asati received his B.E. degree in electrical engineering from Amravati University, India (1996). He worked in 1100 MW thermal power station at Koradi, Nagpur from 1996 to 1997. From 1997 to 2000 he served as as a faculty member at the Visveswarya national Institute of technology, Nagpur. In the year 2000 he joined M.E. program in microelectronics at BITS, Pilani, India. After completing M.E. program he joined as faculty member in electrical and electronics engineering group of BITS, Pilani. He has published several papers in the area of microelectronics and VLSI design. Candidate is pursuing research in the area of microelectronics and VLSI design.

BRIEF BIOGRAPHY OF SUPERVISOR

Dr Chandra Shekhar is Director of Central Electronics Engineering Research Institute (CEERI), pilani, - a constituent national laboratory of the council of Scientific and Industrial Research (CSIR). He received UNESCO/ROSTSCA Young Scientist Award in 1986 for contributions in the area of Informatics and Applications of Computers in Scientific Research and Merit Award as a Project Leader of the UNDP Project - ``Design of ASICs'' on CEERI Foundation Day-1988. His research interests are VLSI Design and Design Methodologies, Analog IC Design and Mixed Signal Design, Processors and Application Specific Processors (Architecture and Design), CAD for VLSI, Physics and Modeling of MOS Devices, VLSI System Applications. He is a fellow member of many professional societies like IETE, Indian Physics, Association, Semiconductor Society of India, Indo-French Technical Association and has played many other professional roles. He has published more than 60 research papers (contributed and invited) in various reputed international/national journals and conferences. He designed the country's first dedicated full-custom LSI processor chip - 16-bit processor for pulse-width modulation (PWM) control of variable frequency AC drives. His research groups designed a serial data communication controller VLSI semi-custom chip and in CDOT's RAX and MAXtelephone switches, also the nation's first general-purpose microprocessor chip - a Motorola 68010 equivalent. He Pioneered the spreading of the new high-productivity state-of-the-art hardware description language based VLSI design methodology in the country. Currently he is working on the design of two application specific VLSI processors and an embedded system based on these processors (to be realized throughs FPGAs) to convert Hindi text into speech.

LIST OF PUBLICATIONS

Journals:

[1] Abhijit Asati and Chandra Shekhar, "Comparison of trans-conductance ratio (β) for a high speed inverter design," ICFAI University Journal of Electrical & Electronics Engineering. VOL. II, NO. 1, pp. 7-13, 2009.

[2] Abhijit Asati and Chandra Shekhar, "A high speed pipelined dynamic circuit implementation using modified TSPC logic design style with improved performance," International Journal of Recent Trends in Engineering, VOL. 1, NO. 3, pp. 191-194, June 2009.

[3] Abhijit Asati and Chandra Shekhar, "Digital CMOS high-speed level shifter design," International Journal of Computers Information Technology and Engineering, VOL. 3, NO. 1, pp. 1-3, January-June 2009.

[4] Abhijit Asati and Chandra Shekhar, "Sizing of pre-charge and pre-discharge transistors for domino logic design style," International Engineering and Technology (IETECH), Journal of Communication Techniques, VOL. 3, NO. 1, pp. 1-4, 2009.

[5] Abhijit Asati and Chandra Shekhar, "A 16×16 MUX based multiplier design using optimized static CMOS logic style," International Journal of Electronic Engineering Research, VOL. 1, NO. 1, pp. 53-61, 2009.

[6] Abhijit Asati and Chandra Shekhar, "VLSI Implementation of a high performance barrel shifter architecture using three different logic design styles," International Journal of Recent Trends in Engineering, VOL. 2, NO. 7, pp. 22-26, November 2009.

Conferences:

[1] Abhijit Asati and Chandra Shekhar, "An improved high speed fully pipelined 500 MHz 8×8 Baugh-Wooley multiplier design using 0.6 μm CMOS TSPC logic design style," IEEE Region 10 Colloquium and 3rd International Conference on Industrial and Information Systems (ICIIS-2008), IIT Kharagpur, India, December 2008.

[2] Abhijit Asati and Chandra Shekhar, "An optimized approach for a CISC microprocessor design using micro coded controller technique," International Conference on Wireless and Embedded systems (WECON-2008), Rajpura, India, October 2008.

[3] Abhijit Asati and Chandra Shekhar, “A high-Speed hierarchical, 16*16 array of array multiplier design,” IEEE, International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT), A.M.U. Aligarh, India, March 2009.

[4] Abhijit Asati and Chandra Shekhar, “Digital CMOS high-speed level shifter design,” International Conference on VLSI and Communication (ICVCom), Santgitts College of Engineering, Kottayam, India, April 2009.

[5] Abhijit Asati, S.K.Sahoo and Dr. Chandra Shekhar, "Selection of optimum device size and trans-conductance ratio for high speed digital CMOS inverter design for a given fan out load," IEEE International Conference On Emerging Trends In Engineering and Technology, (ICETET-09), Nagpur, India, December 2009.

[6] Abhijit Asati and Dr. Chandra Shekhar, "A purely MUX based high speed barrel shifter design," International conference on Advances in Computer Vision and Information Technology, (ACVIT-09), Aurangabad, India, December 2009.