A 1.8V 1.1-GHZ N0VEL
8 x 8-BIT DIGITAL MULTIPLIER

by

Amir Ali Khatibzadeh

B.Sc. in Electrical and Electronics
Khajeh-e Nasir University of Technology
Tehran, Iran
1996

A thesis

presented to Ryerson University

in partial fulfilment of the

requirement for the degree of

Master of Applied Science

in the program

Electrical & Computer Engineering

Toronto, Ontario, Canada, 2004

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Amir Ali Khatibzadeh

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Amir Ali Khatibzadeh

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Amir Ali Khatibzadeh

A 1.8V 1.1GHz Novel Digital Multiplier

Master of Applied Science in
Electrical & Computer Engineering

Ryerson University

Toronto, Ontario, Canada, 2004

This thesis presents the design of an 8 x 8-bit novel multiplier, which can provide a better performance than its counterparts in the sense that it has a fraction of the silicon area, delay and power consumption of the common architectures such as the conventional linear array multipliers.

At the system-level high performance is obtained by implementing a pair-wise multiplication algorithm. Also, parallel addition algorithm is used to add up partial products. Combining these two algorithms results in an efficient cell-based circuit realization. In the circuit-level, pseudo-NMOS full adder cell is chosen amongst the several existing full adder cells due to its superior speed and power performance.

The performance of this design has been evaluated by comparing it to those of the recently reported multipliers. The results of the comparison, both in theory and simulation, prove the superiority of the proposed multiplier.

Acknowledgement

During journey through the Master program support and help from friends, family, and faculty can be invaluable. To begin with, I cannot stress enough that the single most important person contributing to the success of the student is the principle dissertation advisor. With this in mind, I would like to thank Prof. Kaamran Raahemifar for his help and guidance. His extensive knowledge of circuit and keen insight into VLSI design were major assets.

I am also grateful to many people who influenced me throughout my career at Ryerson (sorry if I missed any of you). I would like to thank Prof. Sridhar Krishnan, chair and director of Graduate Program in Department of Electrical and Computer Engineering at Ryerson, who has put his effort and intelligence to make a very vigorous atmosphere in Graduate Studies in the department.

More recently, I would like to thank Prof. Vadim Geurkov for serving on my oral exam committee. His feedback and insightful comments on the material of this thesis are greatly appreciated.

I would like to express my appreciation to Prof. Alireza Sadeghian for reading this thesis and for his helpful comments.

My special thanks and acknowledgements go to Shahab Ardalan,, with whom I spent numerous hours designing VLSI circuits. It has truly been an enlightening experience, and I am thankful for having this opportunity.

I would also like to thank Haleh Vahedi and Majid Malekan for their technical advices regarding the layout.

Last, but certainly not least, I would like to thank my dear friend, Dana, and my family and loved ones who stood by me throughout these years. I will not name you all, but I am eternally grateful. Words cannot express the warmth and gratitude I feel for each of you.

Fabrication supports through Canadian Microelectronics Corporation (CMC) is also gratefully acknowledged.

This thesis is dedicated to the late eminent scholar, Professor Abbas Sahab, my grandfather, who has been named the father of Iran's cartography.

Table of Contents

Abstract iv
Acknowledgements v
List of Tables x
List of Figures xii

Chapter 1: Introduction **1**

1.1 Motivation 1
1.2 Application 2
1.3 Original Contributions 3
1.4 Thesis Organization 5

Chapter 2: Basic Concept of Multiplication **6**

2.1 Multiplication Definition 7
2.2 Binary Multiplication 7
2.3 Review of Parallel Multiplication Algorithms 8
    2.3.1 Bruan Algorithm 9
    2.3.2 Baugh-Wooley Algorithm 10
    2.3.3 The Modified Booth Algorithm 14
    2.3.4 Wallace Tree Algorithm 17
    2.3.5 The Proposed Pair-Wised Algorithm 19
2.4 Qualitative Comparisons of Parallel Algorithms 22

Chapter 3: Multiplier Design **24**

3.1 Pair-Wise Multiplier 24
    3.1.1 Circuit Level Review 25
        3.1.1.1 Full Adder 25
            3.1.1.1.1 Simulation Strategies 34
            3.1.1.1.2 Power Consumption Performance 37
            3.1.1.1.3 Delay Performance 38
            3.1.1.1.4 Performance Comparison 39
        3.1.1.2 Carry Lookahead Adder 41
        3.1.1.3 AND/NAND/OR/XOR Gates 45
    3.1.2 Cell Design 47

Chapter 4: Simulation Results & Layout Considerations **59**

4.1 Simulation Results of the Individual Circuits 59
4.2 Final Simulation Results 67
4.3 Layout Considerations 86
    4.3.1 Layout Strategies 86
    4.3.2.Pads, Package and Chip Size 89

Chapter 5: Conclusion **91**

5.1 Features of the Designed Multiplier 91

5.2 Comparison Results 93
5.3 Future Work 95

References **98**

List of Tables

Table 2.1 Partial products selection 15

Table 2.2 Partial product generation process 15

Table 2.3 Partial product generation relation 15

Table 3.1(a) Truth table of a full adder 26

Table 3.1(b) Truth table of a half adder 26

Table 3.2 Transistor dimension in complementary CMOS full adder 28

Table 3.3 Transistor dimension in complementary pass-transistor full adder 30

Table 3.4 Transistor dimension in double pass-transistor full adder (Sum) 31

Table 3.5 Transistor dimension in double pass-transistor full adder (C_{out}) 31

Table 3.6 Transistor dimension in transmission gate full adder 32

Table 3.7 Transistor dimension in pseudo-NMOS full adder 33

Table 3.8 Transistor dimension in XOR & transmission gate full adder 34

Table 3.9 Characteristic of the input signals 37

Table 3.10 Simulation results for the full adders sorted by power consumption 38

Table 3.11 Simulation results for the full adders sorted by propagation delay 38

Table 3.12 Simulation results for the full adders sorted by power-delay product 40

Table 3.13 Delay of the generate, propagate and sum signals of PFA 44

Table 3.14 Truth table of AND, NAND, OR and XOR 45

Table 3.15 Transistor dimension & delay of AND, NAND, XOR and OR gates 46

Table 4.1 Transitions covered by input pattern (a) 61

Table 4.2 Transitions covered by input pattern (b) 62

Table 4.3 Transitions covered by input pattern (c) 63

Table 4.4 Transitions covered by input pattern (d) 64

Table 4.5 The results of the worst-case delay measurement 71

Table 4.6 The numerical results of several intentional multiplications 82

Table 4.7 The numerical results of several random multiplications sorted by delay 82

Table 5.1 Performance of the proposed multiplier 92

Table 5.2 Summary of the performance of the recent publications on digital multiplier 94

List of Figures

Fig. 2.1 Partial products of an 8 x 8-bit unsigned integer multiplication 10

Fig. 2.2 Braun's array multiplier 10

Fig. 2.3. Illustration of the partial product terms in Baugh-Wooley algorithm 12

Fig. 2.4. Reorganization of the partial product terms of Fig. 2.3 12

Fig. 2.5. 8 x 8-bit Baugh-Wooley two's complement regular array 13

Fig. 2.6 Block diagram of the n x n multiplier using modified Booth algorithm 17

Fig. 2.7 Construction of Wallace's tree for an 8 x 8-bit multiplier, reduction of the 8 partial products with 4-2 compressors 18

Fig. 2.8 Architecture of Wallace's tree for an 8 x 8-bit multiplier 18

Fig. 2.9 Block diagram for the 8 x 8-bit pair-wise multiplier 21

Fig. 3.1 Schematic of complementary CMOS full adder 28

Fig. 3.2 Schematic of complementary pass-transistor full adder 29

Fig. 3.3(a) Schematic of double pass-transistor full adder (Sum) 30

Fig. 3.3(b) Schematic of double pass-transistor full adder (C_{out}) 31

Fig. 3.4 Schematic of transmission gate full adder 32

Fig. 3.5 Schematic of pseudo-NMOS full adder 33

Fig. 3.6 Schematic of XOR & transmission gate full adder 34

Fig. 3.7 (a) Input patterns used to evaluate the performance of the adders 35

Fig. 3.7 (b) Input patterns used to evaluate the performance of the adders 35

Fig. 3.7 (c) Input patterns used to evaluate the performance of the adders 36

Fig. 3.7 (d) Input patterns used to evaluate the performance of the adders 36

Fig. 3.8 Propagation delay measurement 39

Fig. 3.9 Block diagram of 4-bit carry lookahead adder 43

Fig. 3.10 Gate-level implementation of partial full adder (PFA) 43

Fig. 3.11 Block diagram of the 16-bit carry lookahead adder implemented by cascading four 4-bit carry lookahead modules 44

Fig. 3.12 Schematic of (a) AND (b) NAND (c) XOR (d) OR gates 46

Fig. 3.13 Block diagram of the proposed 8 x 8-bit multiplier showing detail of the required cells 47

Fig. 3.14 Block diagram of AND generator 48

Fig. 3.15 Gate-level of the AND plane (X_iY_j Cell) 49

Fig. 3.16 Block diagram of partial products generator 50

Fig. 3.17 Gate-level of one partial product generator (PP Cell) 52

Fig. 4.1 Circuit structure used for simulation of full adder cell 60

Fig. 4.2 Circuit structure used for simulation of AND/NAND/OR/XOR gates 60

Fig. 4.3 The simulation waveforms showing respond of the pseudo full adder to the input pattern (a) 61

Fig. 4.4 The simulation waveforms showing respond of the pseudo full adder to the input pattern (b) 62

Fig. 4.5 The simulation waveforms showing respond of the pseudo full adder to the input pattern (c) 63

Fig. 4.6 The simulation waveforms showing respond of the pseudo full adder to the input pattern (d) 64

Fig. 4.7 The simulation waveforms showing respond of the AND/NAND gate 65

Fig. 4.8 The simulation waveforms showing respond of OR gate 66

Fig. 4.9 The simulation waveforms showing respond of XOR gate 66

Fig. 4.10 The critical path of the proposed multiplier 68

Fig. 4.11 Delay of AND gate 69

Fig. 4.12 The worst-case delay of pseudo-NMOS adder occurring when $A=1, B=1$ and $C_{in}=1$ 69

Fig. 4.13 The worst-case delay of 16-bit carry lookahead adder 70

Fig. 4.14 Input waveforms of X = 11111111 and Y = 11111111 72

Fig. 4.15 The post layout simulation waveforms showing results of "11111111" x "11111111" 73

Fig. 4.16 The measured delay of the proposed multiplier corresponding to "11111111" x "11111111" 74

Fig. 4.17 The waveform of current drawn by V_{dd} node ("11111111" x "11111111") 78

Fig. 4.18 The waveform of current drawn by V_{dd} node ("10101010" x "01010101") 79

Fig. 4.19 waveform of the input patterns "10101010" x "01010101" 80

Fig 4.20 The simulation waveforms of multiplication product of the input patterns shown in Fig 4.19 81

Fig. 4.21 Adding two extra pins to V_{dd} and V_{ss} reducing the parasitic inductance 84

Fig. 4.22 The simulation results of the final outputs against temperature variation 85

Fig. 4.23 Layout of the pseudo-NMOS full adder (die size of 22 x 8.5 μm^2) 87

Fig. 4.24 Layout of AND gate (die size of 7.9 x 5.6 μm^2) 87

Fig. 4.25 Layout of NAND gate (die size of 5.4 x 5.6 μm^2) 88

Fig. 4.26 Layout of XOR gate (die size of 10.2 x 20.7 μm^2) 88

Fig. 4.27 Layout of core of the 8 x 8-bit proposed multiplier core (die size of 0.275 x 0.38 mm^2) 89

Fig. 4.28 The proposed 8 x 8-bit multiplier chip 90

Chapter 1

Introduction

1.1 Motivation

The core of every microprocessor, digital signal processor (DSP), and data processing application-specific integrated circuit (ASIC) is its data path. It is often the crucial circuit component if die area, power consumption, and especially operation speed are of concerns. At the heart of data path and addressing units are arithmetic units, such as comparators, adders, and multipliers. Finally, one of the basic operations found in most arithmetic components is binary multiplication. Besides simply multiplying two numbers, multipliers are also used in more complex operations like address calculation and division. Also, simpler operation like magnitude comparison is based on binary multiplication.

Multiplication is also very critical if implemented in hardware because it involves an expensive carry-propagation step, when partial product addition is performed. The efficient implementation of multiplication operation in an integrated circuit is a key problem in VLSI design. Designing fast and power-efficient multiplier has been of great theoretical and practical interest for computer scientists and engineers. Several algorithms and various VLSI implementations have been proposed [1, 2, 3, 4, 5, 6, 7] and practically used.

In order to achieve high performance multiplier, it is necessary to operate very efficiently in terms of speed and power trade-off in all design levels. Increasing the operating speed of the circuits to make more computations with lower power consumption is the main motivation in multiplier design.

The recent progress in use of Ultra Deep Sub-Micron Devices (UDSM) helps to overcome the area constraint. Employing advanced cell-based architectures constantly improves productivity in ASIC design. Taking all of these into account, implementing low-power circuit techniques based on a fast multiplication algorithm is still a cost-effective and feasible alternative for increasing the performance of the multipliers substantially. Therefore, this thesis deals with designing a novel multiplier with inherent high-speed characteristic and power efficient performance.

1.2 Applications

Wireless communication systems, including third generation cellular radio systems and wireless Local Area Network Systems (LANS), have become tremendously popular in recent years. These systems can be implemented using various platforms, such as digital signal processors, ASICs and Field Programmable Gate Arrays (FPGAs). Most digital signal processing systems incorporate a multiplication unit to implement algorithms such as correlation, convolution, filtering and frequency analysis. These algorithms are used in applications such as finite impulse filters (FIR), infinite impulse filters (IIR), discrete cosine transforms (DCT), and fast Fourier transforms (FFT). Moreover, there has been a rapid increase in the popularity of portable and wireless electronic devices, such as laptop computers, portable video players and cellular phones, which rely on embedded digital processors. Since the desire is to design digital systems for communication applications at the best performance without scarifying power, high performance and low power multipliers are inevitable.

1.3 Original Contributions

This thesis presents the design of novel multiplier architecture, with superior performance in speed, power consumption and area compared to traditional array multipliers.

In order to achieve architecture with high performance several foremost parallel multiplication algorithms have been studied and compared. Braun, Baugh-Wooley, Wallace and pair-wise algorithms have been reviewed in detail. Among these algorithms pair wise algorithm has been chosen due to its superiority in speed of operation.

The 8 x 8-bit multiplier based on this algorithm operates by:

1) generating four 8-bit (X_e, X_o, Y_e, Y_o) numbers using even and odd positions of the multiplicand (X) and multiplier (Y).
2) multiplying these four 8-bits numbers to generate the four 15-bit numbers ($P_{ee}, P_{eo}, P_{oe}, P_{oo}$) known as the even and odd elements of the partial products (P).
3) adding the result of the multiplication of elements of partial products. The addition is performed in four steps by using 3-to-2 adding technique which results in two 15-bit numbers.
4) adding two final 16-bit numbers (P_e, P_o) and thus generating the product of multiplication via a fast carry lookahead.

In the first step of design flow, topology selection, six full adder cells based on CMOS static logic styles are redesigned and examined at transistor-level in standard 0.18μ CMOS technology. The results of the extensive evaluation, which are further presented in

Chapter 3, prove that 14-transistor pseudo-NMOS full adder cell offers a better speed and power trade-off with less numbers of transistors [8, 9, 10].

The validity of the design strategy is by proven by testing the complete multiplier and measuring the speed and power. All the designs are simulated using Cadence Computer Aided Design (CAD) Tool in 0.18μm CMOS technology at 1.8V supply voltage.

In summary a speed/power efficient novel multiplier for medium bit width applications is designed in this thesis. Leading by a quantitative analysis of the characteristics of static CMOS logic adders, several topologies are examined to support the final circuit design.

The major contributions of the thesis are summarized as follows:

- An in-depth comparative analysis of the characteristics of static CMOS adder cells is conducted, and useful insights are obtained.
- Power reduction through algorithm selection is achieved by:
    a) Minimizing the number of operations and, hence, the number of hardware resources (half adder cells used anywhere possible)

    b) Reducing the number of complex operations by transforming mathematic expressions (cascading four 4-bit carry lookahead adders instead of implementing a 16-bit carry lookahead structure, which requires complex logic operation)
- Power reduction through circuit/logic is achieved by using static style rather than dynamic style. This causes the architectural level to be free from clock and related clocking issues such as clock skew and high dynamic power.
- Flexibility in delay modeling in system-level in such a way that modifying the entire multiplier for different speed requirements is straightforward.
- The performance of the proposed multiplier is well enhanced by considering transistor chaining, grouping, and signal sequencing in the adder layout which is

proven to provide substantial power saving and speed improvement at no area penalty.

These original contributions have been published in two conference proceedings [9, 10].

1.4 Thesis Organization

This thesis consists of 5 chapters and is organized as:

Following the introductory Chapter 1, Chapter 2 describes the basic concept of two's complement multiplication. The most known parallel multiplication algorithms used in VLSI implementation along with the pair-wise multiplication algorithm are introduced and a brief qualitative comparison of these algorithms is presented.

In Chapter 3, first the top-level design of pair-wise multiplier is presented. Topology selection of the main elements as a result of an extensive performance analysis on adder cells further reviewed. The circuit design of the required cells for pair-wise structure is also discussed.

Chapter 4, is dedicated to the simulation results of individual circuits and cells as well as the final simulation results of the proposed multiplier. Layout considerations are also discussed.

Finally, Chapter 5 presents the features of the Designed Multiplier. A comparative study of the previous works on multipliers is presented to better evaluate on this work. Drawing conclusion, summarizing the contributions of this thesis, and outlining the directions for the future investigations bring this chapter to an end.

Chapter 2

Basic Concepts of Multiplication

Multiplication is one of the main arithmetic operations. Multiplier represent a fundamental building block which is being widely used in many Very Large-Scale Integrated (VLSI) systems such as application-specific Digital Signal Processing (DSP) architectures, microprocessors and systems which implement filtering, encryption, security processing and image processing. In addition to their main task, which is multiplying two binary numbers, multipliers are the nucleus of many other useful operations such as division and address calculation. In these systems the multipliers are the part of the critical path that determines the overall performance of the system. That is why enhancing the performance of multiplier is a significant goal.

Parallel to high-speed system design [11], low-power systems [1] are highly in demand because of the fast growing technologies in mobile communication and computation. The battery technology does not advance at the same rates as the microelectronics technology. There is a limited amount of power available for mobile systems. Thus, designers are faced with more constraints; high-speed, high throughput, small silicon area and at the same time, low-power consumption. Therefore, low power, high-performance multiplier is of great interest.

Current architectures range from small, low performance array to tree multipliers. Conventional linear array multipliers achieve high performance in a regular structure, but require large area of silicon. Tree structures achieve even higher performance than linear arrays but the tree interconnection is more complex and less regular, making them even larger than linear arrays. Ideally, one would wish the speed benefits of a tree structure, the regularity of an array multiplier, and the small size of a shift and add multipliers.

The first section of this Chapter explains the basics of binary multiplication. A review on the most known parallel multiplication algorithms is presented in Section 2.3. The pair-wise multiplication algorithm that has been used in the proposed multiplier is also described. These algorithms are, then, briefly compared against each other at the end of this Chapter.

2.1 Multiplication Definition

Multiplication is defined as "a mathematical operation that at its simplest form is an abbreviated process of adding an integer to itself a specified number of times." A number **(multiplicand)** is added to itself a number of times as specified by another number **(multiplier)** to form a result **(product)**. Multiplication starts with placing the multiplicand on top of the multiplier. The multiplicand is then multiplied by each digit of the multiplier beginning with the rightmost, Least Significant Digit (LSD). Intermediate results **(partial products)** are placed one atop the other, offset by one digit to align digits of the same weight. The final product is determined by summation of all the partial products. This technique applies equally to any base, including binary.

2.2 Binary Multiplication

In the binary number system the digits, called bits, are limited to the set [0, 1]. The result of multiplying any binary number by a single binary bit is either 0, or the original number. This makes forming the intermediate partial products simple and efficient.

Summing these partial products is the time-consuming task for binary multipliers. One logical approach is to form the partial-products one at a time and sum them as they are generated. This technique works fine but is slow. For applications where this approach does not provide good enough performance, another approach is used which is known as parallel multiplication algorithms. In this latter approach all bit-products are generated in parallel and a multi-operand adder (i.e., an adder tree) is used for their accumulation. Multipliers that operate based on these algorithms are called parallel multipliers. Parallel multipliers are becoming the key components in Reduced Instruction Set Computers (RISCs), DSP and graphic accelerators due to their inherent higher speed of operation. This brings parallel multiplication to the main focus of our discussion.

2.3 Review of Parallel Multiplication Algorithms

Since multiplication is one of the most critical operations in many computational systems, many algorithms have been proposed to perform multiplication, each offering different advantages and having tradeoffs in terms of speed, circuit complexity, area and power consumption. Among the multipliers reported parallel multipliers have been of great theoretical and practical interests for VLSI designers not only for their speed of operation but also for their ease of implementation.

The structure of all parallel multipliers can be partitioned into three parts performing three major tasks:

a) Partial product generation.

b) Carry-free addition.

c) Carry-propagation addition.

These three parts can be implemented using different schemes such as simple AND gate or Booth algorithm to generate partial products. The carry-free addition task is often implemented by using a Wallace tree or redundant binary addition tree.

In the following section four well-known parallel algorithms as well as pair-wise algorithm, which have been used in VLSI implementation of digital multipliers, are briefly presented. The readers can consult references [11,12] for more details on parallel multiplication algorithms.

2.3.1 Braun Algorithm

Consider two unsigned numbers $X = X_{n-1} \dots X_1X_0$ and $Y = Y_{n-1} \dots Y_1Y_0$, where

$$X = \sum_{i=0}^{i=n-1} X_i 2^i , \tag{2.1}$$

$$Y = \sum_{i=0}^{i=n-1} Y_i 2^i . \tag{2.2}$$

The product $P = P_{2n-1} \dots\dots P_1P_0$, which results from multiplying the multiplicand X by the multiplier Y, can be written in the following form

$$P = \sum_{i=0}^{i=n-1} \sum_{j=0}^{j=n-1} (X_iY_j)2^{i+j} . \tag{2.3}$$

Each of the partial product terms $P_k = X_iY_j$ is called a summand. Fig.2.1 shows an example of an 8 x 8-bit multiplication.

The summands are generated in parallel with AND gates. Fig. 2.2 shows the Braun's array multiplier [4]. Such an n x n multiplier requires n x (n -1) adders and n^2 AND gates. The delay of such a multiplier is determined by the delay of the full adder cell and the final adder in the last row. In the multiplier array a full-adder with balanced carry and sum delays is desirable because the sum and carry signals are both on the critical path. For the large arrays, the speed and power of the full adder are both very important.

								X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1
								Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1
								X_8Y_1	X_7Y_1	X_6Y_1	X_5Y_1	X_4Y_1	X_3Y_1	X_2Y_1	X_1Y_1
							X_8Y_2	X_7Y_2	X_6Y_2	X_5Y_2	X_4Y_2	X_3Y_2	X_2Y_2	X_1Y_2	0
						X_8Y_3	X_7Y_3	X_6Y_3	X_5Y_3	X_4Y_3	X_3Y_3	X_2Y_3	X_1Y_3	0	0
					X_8Y_4	X_7Y_4	X_6Y_4	X_5Y_4	X_4Y_4	X_3Y_4	X_2Y_4	X_1Y_4	0	0	0
				X_8Y_5	X_7Y_5	X_6Y_5	X_5Y_5	X_4Y_5	X_3Y_5	X_2Y_5	X_1Y_5	0	0	0	0
			X_8Y_6	X_7Y_6	X_6Y_6	X_5Y_6	X_4Y_6	X_3Y_6	X_2Y_6	X_1Y_6	0	0	0	0	0
		X_8Y_7	X_7Y_7	X_6Y_7	X_5Y_7	X_4Y_7	X_3Y_7	X_2Y_7	X_1Y_7	0	0	0	0	0	0
	X_8Y_8	X_7Y_8	X_6Y_8	X_5Y_8	X_4Y_8	X_3Y_8	X_2Y_8	X_1Y_8							
P_{16}	P_{15}	P_{14}	P_{13}	P_{12}	P_{11}	P_{10}	P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1

Fig. 2.1 Partial products of an 8 x 8-bit unsigned integer multiplication

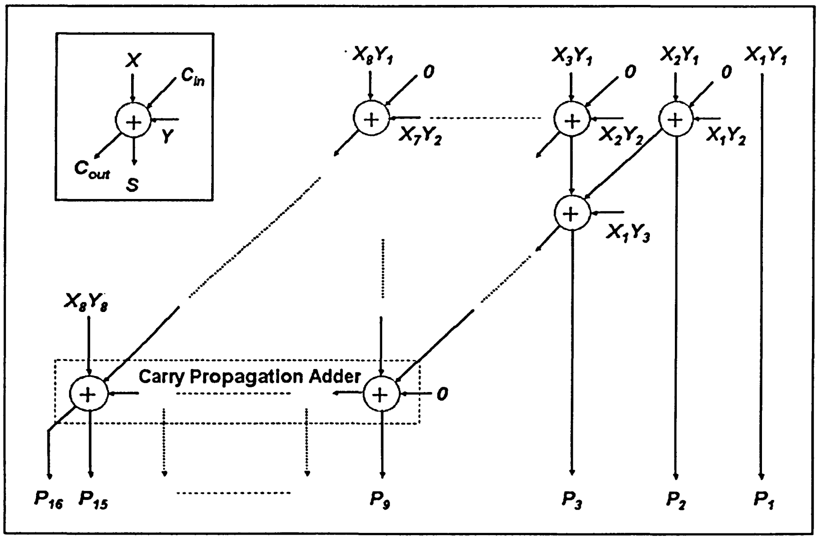


Fig. 2.2 Braun's array multiplier

2.3.2 Baugh-Wooley Algorithm

Baugh-Wooley is one of the developed algorithms for parallel multiplication, which has been used in VLSI architectures [12]. Multipliers based on this algorithm are used for direct multiplication of two's complement numbers. This direct approach does not need any two's complementing operations prior to multiplication. Using the Baugh-Wooley algorithm, the product of two numbers X and Y expressed in two's complement,

$$X = -X_{n-1}2^{n-1} + \sum_{i=0}^{i=n-2} X_i 2^i , \tag{2.4}$$

$$Y = -Y_{n-1}2^{n-1} + \sum_{i=0}^{i=n-2} Y_i 2^i , \tag{2.5}$$

is given by

$$P = XY = X_{n-1}Y_{n-1}2^{2n-2} + \sum_{i=0}^{i=n-2}\sum_{j=0}^{j=n-2} X_i Y_j 2^{i+j} - X_{n-1}\sum_{i=0}^{i=n-2} Y_i 2^{n+i-1} - Y_{n-1}\sum_{i=0}^{i=n-2} X_i 2^{n+i-1} . \tag{2.6}$$

In order to avoid the use of subtractor cells and use only adders, the negative terms should be transformed. So

$$-X_{n-1}\sum_{i=0}^{i=n-2} Y_i 2^{n+i-1} = X_{n-1}(-2^{2n-2} + 2^{n-1} + \sum_{i=0}^{i=n-2} \overline{Y}_i 2^{n+i-1}) . \tag{2.7}$$

Using this property in equation (2.5), the product P becomes

$$P = XY = -2^{2n-1} + (\overline{X}_{n-1} + \overline{Y}_{n-1} + X_{n-1}Y_{n-1}).2^{2n-2} +$$

$$\sum_{i=0}^{i=n-2}\sum_{j=0}^{j=n-2} X_i Y_j 2^{i+j} + (X_{n-1} + Y_{n-1}).2^{n-1} + X_{n-1}\sum_{i=0}^{i=n-2} \overline{Y}_i 2^{n+i-1} + Y_{n-1}\sum_{i=0}^{i=n-2} \overline{X}_i 2^{n+i-1} \tag{2.8}$$

From Equation 2.8, it can be seen that the multiplication of two numbers, expressed in two's complement representation, can be written in a form which involves only positive bit products. The product is, then, obtained by adding a constant to the final result. All the partial product terms to generate the above product are explicitly shown in Fig. 2.3. A simple reorganization of Fig. 2.3 results in the array of partial product shown in Fig. 2.4, which is a modified version of the original Baugh-Wooley algorithm.

It can be seen that half adder, full adder, NAND and AND gates are the required elements by Baugh-Wooley algorithm to perform two's complement multiplication.

								Y_8	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1
								X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1
									X_1Y_7	X_1Y_6	X_1Y_5	X_1Y_4	X_1Y_3	X_1Y_2	X_1Y_1
								X_2Y_7	X_2Y_6	X_2Y_5	X_2Y_4	X_2Y_3	X_2Y_2	X_2Y_1	0
							X_3Y_7	X_3Y_6	X_3Y_5	X_3Y_4	X_3Y_3	X_3Y_2	X_3Y_1	0	0
						X_4Y_7	X_4Y_6	X_4Y_5	X_4Y_4	X_4Y_3	X_4Y_2	X_4Y_1	0	0	0
					X_5Y_7	X_5Y_6	X_5Y_5	X_5Y_4	X_5Y_3	X_5Y_2	X_5Y_1	0	0	0	0
				X_6Y_7	X_6Y_6	X_6Y_5	X_6Y_4	X_6Y_3	X_6Y_2	X_6Y_1	0	0	0	0	0
			X_7Y_7	X_7Y_6	X_7Y_5	X_7Y_4	X_7Y_3	X_7Y_2	X_7Y_1	0	0	0	0	0	0
0	X_8Y_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	$\overline{X_8Y_7}$	$\overline{X_8Y_6}$	$\overline{X_8Y_5}$	$\overline{X_8Y_4}$	$\overline{X_8Y_3}$	$\overline{X_8Y_2}$	$\overline{X_8Y_1}$	1	1	1	1	1	1	1
1	1	$\overline{X_7Y_8}$	$\overline{X_6Y_8}$	$\overline{X_5Y_8}$	$\overline{X_4Y_8}$	$\overline{X_3Y_8}$	$\overline{X_2Y_8}$	$\overline{X_1Y_8}$	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Fig. 2.3 Illustration of the partial product terms in Baugh-Wooley algorithm

								$\overline{X_1Y_8}$	X_1Y_7	X_1Y_6	X_1Y_5	X_1Y_4	X_1Y_3	X_1Y_2	X_1Y_1
							$\overline{X_2Y_8}$	X_2Y_7	X_2Y_6	X_2Y_5	X_2Y_4	X_2Y_3	X_2Y_2	X_2Y_1	
						$\overline{X_3Y_8}$	X_3Y_7	X_3Y_6	X_3Y_5	X_3Y_4	X_3Y_3	X_3Y_2	X_3Y_1		
					$\overline{X_4Y_8}$	X_4Y_7	X_4Y_6	X_4Y_5	X_4Y_4	X_4Y_3	X_4Y_2	X_4Y_1			
				$\overline{X_5Y_8}$	X_5Y_7	X_5Y_6	X_5Y_5	X_5Y_4	X_5Y_3	X_5Y_2	X_5Y_1				
			$\overline{X_6Y_8}$	X_6Y_7	X_6Y_6	X_6Y_5	X_6Y_4	X_6Y_3	X_6Y_2	X_6Y_1					
		$\overline{X_7Y_8}$	X_7Y_7	X_7Y_6	X_7Y_5	X_7Y_4	X_7Y_3	X_7Y_2	X_7Y_1						
1	X_8Y_8	$\overline{X_8Y_7}$	$\overline{X_8Y_6}$	$\overline{X_8Y_5}$	$\overline{X_8Y_4}$	$\overline{X_8Y_3}$	$\overline{X_8Y_2}$	$\overline{X_8Y_1}$							

Fig. 2.4 Reorganization of the partial product terms of Fig. 2.3

This algorithm is suitable for applications where operands with less than 16 bits are to be processed. Digital filters where small operands are used (e.g. 6, 8 and 12), are examples of such applications.

Fig. 2.5 shows the array architecture of an 8 x 8-bit Baugh-Wooley two's complement.

For operands equal to or greater than 16-bits, the Baugh-Wooley scheme becomes area consuming and slow. Hence, techniques to reduce the size of the array, while maintaining the regularity, are required.

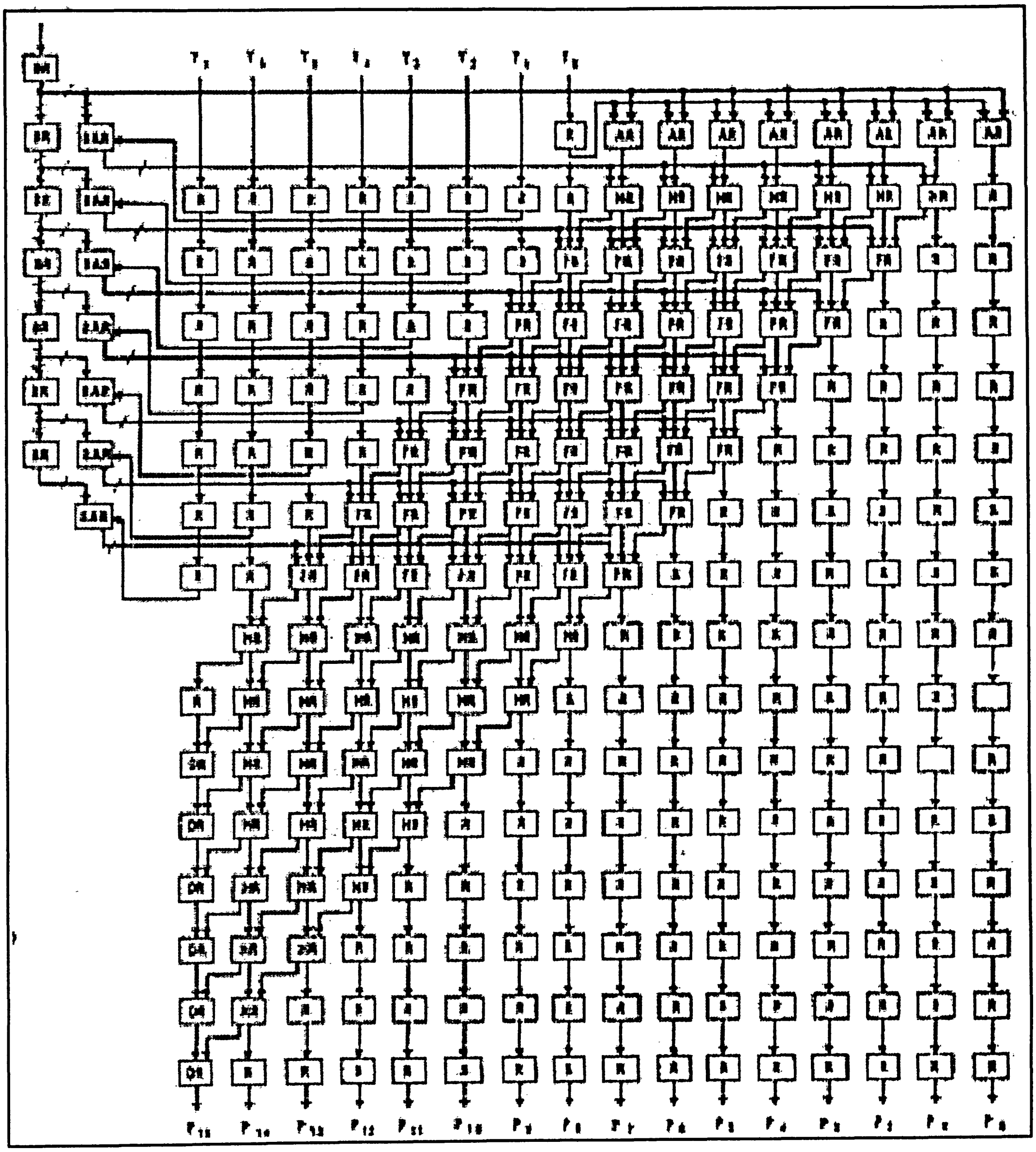


Fig. 2.5 8 x 8-bit Baugh-Wooley two's complement regular array

2.3.3 The Modified Booth Algorithm

For operands equal to or greater than 16-bits, the modified Booth algorithm [13] has been extensively used. It is based on encoding the two's complement operand (i.e., multiplier) in order to reduce the number of partial products to be added.

This makes the multiplier faster and uses less hardware (area). For example, the modified Radix-2 algorithm is based on partitioning the multiplier into overlapping groups of 3-bits, and each group is decoded to generate the correct partial product.

The multiplier, Y, in the two's complement can be written as:

$$Y = -Y_{n-1}2^{n-1}\sum_{i=0}^{i=n-2} Y_i 2^i . \qquad (2.9)$$

This can be rewritten as:

$$Y = \sum_{i=0}^{i=n-2}(Y_{2i-1} + Y_{2i} - 2Y_{2i+1}).2^{2i} \text{ with } Y_{-1} = 0. \qquad (2.10)$$

In Equation 2.10, the terms in brackets assume values from the set {-2, -1, 0, +1, +2}. The encoding of Y, using the modified Booth algorithm, generates another number with the following five signed digits, -2, -1, 0, +1, +2. As illustrated in Table 2.1, each encoded digit in the multiplier performs a certain operation on the multiplicand X.

The bits of the multiplier (Y) are partitioned into groups of overlapping 3-bits and each group permits the generation of certain partial products. The five possible multiplies of the multiplicand are generated based on the procedure given in Table 2.2.

The general partial product is related to the multiplicand for each encoded digit by the relationships presented in Table 2.3. PP_i is the partial product and PP_i is also the sign bit of the partial product with $P_n=P_{n-1}$ when no shifting of the partial product is performed. Note that the partial product is requested on (n +1) bits.

Table 2.1. Partial products selection

Y_{2i+1}	Y_{2i}	Y_{2i-1}	**Recoded Digit**	**Operation on X**
0	0	0	0	0 x X
0	0	1	+1	+1 x X
0	1	0	+1	+1 x X
0	1	1	+2	+2 x X
1	0	0	-2	-2 x X
1	0	1	-1	-1 x X
1	1	0	-1	-1 x X
1	1	1	0	0 x X

Table 2.2. Partial product generation process

Recoded Digit	**Operation on X**
0	Add 0 to the partial product
+1	Add X to the partial product
+2	Shift left X one position and add it to the partial product
-1	Add two's complement of X to the partial product
-2	Take two's complement of X and shift left one position

Table 2.3. Partial product generation relation

Recoded Digit	**Operation on X**		**Added to LSB**
0	$PP_i = 0$	for i = 0, ..., n	0
+1	$PP_i = X_i$	for i = 0, ..., n	0
+2	$PP_i = X_{i-1}$	for i = 0, ..., n	0
-1	$PP_i = X_i$	for i = 0, ..., n	1
-2	$PP_i = X_{i-1}$	for i = 0, ..., n	1

Bits are grouped into 3-bit groups overlapping by one bit. A bit with a value of zero is added on the right side of Y as Y_{-1}. So the multiplication of two 8-bit numbers generates only 4 partial products. The number of partial products is then reduced by half.

In order to make the array rectangular and thus more regular for VLSI implementation, the problem of the sign extension must be addressed. This problem is more crucial when the operand lengths are wide, where each partial product must be sign-extended to the length of the product. The basic idea is to use two extra bits in the partial product. For the first partial product, the two additional bits, PP_{n+1} and PP_{n+2} are equal to the sign bit of the partial product

$$PP_{n+2} = PP_{n+1} = PP_n \ . \tag{2.11}$$

For the second partial product, if the first partial product was positive, then the two additional bits for this second partial product are given by the Equation 2.11, otherwise we have two different cases

$$PP_{n+2} = PP_{n+1} = 1 \qquad \text{if} \qquad PP_n = 0, \tag{2.12}$$

and

$$PP_{n+2} = PP_{n+1} = 1 \qquad \text{if} \qquad PP_n = 0. \tag{2.13}$$

So it is more interesting to use a third bit F as a flag to indicate whether there is, from the previous partial, a negative sign bit to be propagated. F_1 is the flag generated by the first partial product to the next one. This flag is expressed by the following Boolean equation

$$F_{j+1} = F_j + PP_{n,j}, \tag{2.14}$$

where $PP_{n,j}$ is the sign bit of the j$^{\text{th}}$ partial product.

Fig. 2.6 shows the block diagram of an n x n modified Booth multiplier. Furthermore, the figure gives an idea about the floorplan of this subsystem.

The diagram is composed of the following blocks:

a) The multiplier array containing partial product's generation and 1-bit adders.

b) The Booth encoder and the sign extension bits (PP_{n+2}, PP_{n+1}, F).

c) The Booth encoder generates the five signals (0, +1, +2x, -1x, and -2x) for each group of 3-bit of Y.

d) The final stage adder performs 2n bits addition.

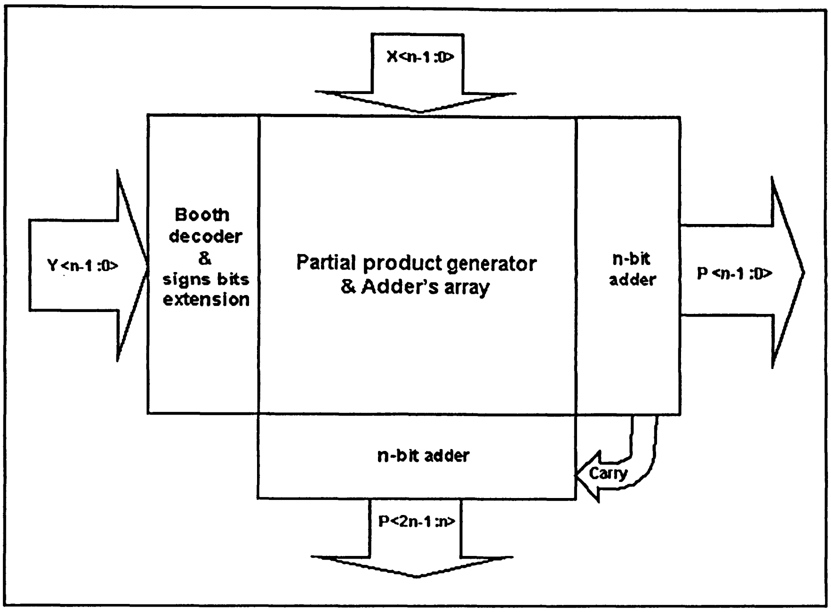

X<n-1 :0>
Y<n-1 :0>
Booth decoder & signs bits extension
Partial product generator & Adder's array
n-bit adder
P <n-1 :0>
n-bit adder
Carry
P<2n-1 :n>

Fig. 2.6 Block diagram of the n x n multiplier using modified Booth algorithm

The Booth multiplier exhibits some glitches. The main reason for glitches is the race condition between the multiplicand and the multiplier due to the Booth encoder.

2.3.4 Wallace Tree Algorithm

As seen in the previous section, applying the Booth algorithm reduces the number of partial products by half. However, for large multipliers such as 32-bit and over, the number of the partial products is over 16 bits. In such cases, better performance is achieved by adopting the Wallace tree using 4-2 compressors [12]. A 4-2 compressor accepts 4 numbers and a carry in, and sums them to produce 2 numbers and a carry out. Fig. 2.7 shows an example of such a tree on partial products of an unsigned 8 x 8-bit multiplier. Eight partial products are produced. Using 4-2 compressors, two levels of additions (stages) are needed. The final two summands are added using a fast 16-bits adder. Some zeros are added to the array. This example shows that the bits which are not used in this 1st stage (level) jump to the next stage to be combined with the ones produced by the compressors.

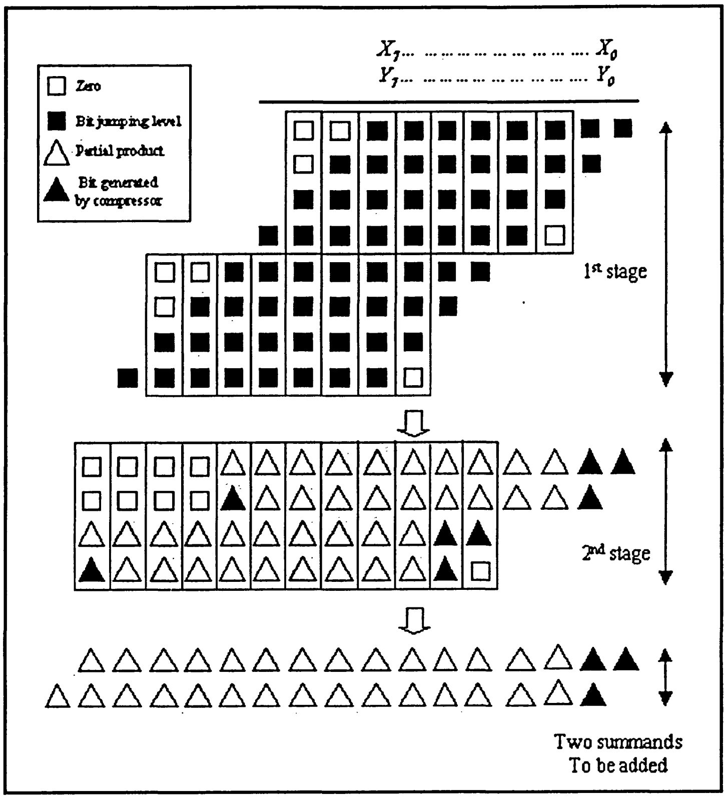


Fig. 2.7 Construction of Wallace's tree for an 8 x 8-bit multiplier, reduction of the 8 partial products with 4-2 compressors

Fig 2.8 shows the architecture of the 8 x 8-bit multiplier. As one can see for the first stage of the tree two blocks, A and B, are required.

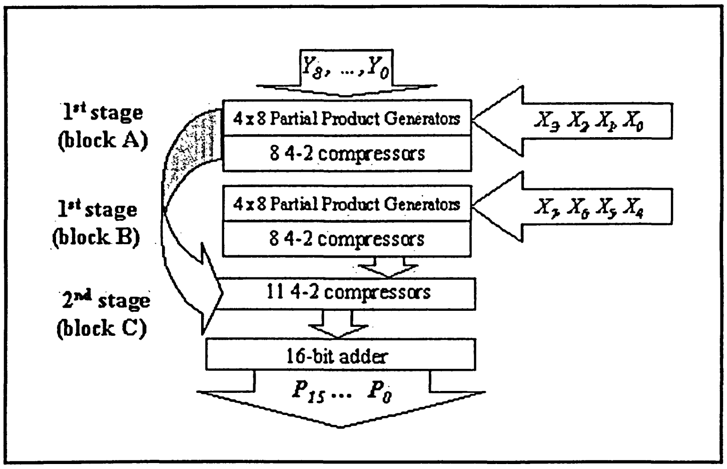


Fig.2.8 Architecture of Wallace's tree for an 8 x 8-bit multiplier

The block A of the compressor would group the first (last) four partial products, respectively.

To further enhance the performance of Wallace tree multiplier, the modified Booth algorithm can be used to reduce the number of partial products by half in a carry-save adder array. This architecture exhibits some irregularities in the layout since it has a complicated interconnection scheme. Hence, the interconnection wires affect the speed and power consumption of the adder.

2.3.5 The Proposed Pair-Wise Algorithm

This algorithm is based on generating n-bit numbers using even and odd positions of the two n-bit numbers [14]. Then, parallel addition algorithm is used to add up partial products.

If we assume the two multiplicands X and Y are 8-bit numbers as follow:

$$X = < X_8, X_7, X_6, X_5, X_4, X_3, X_2, X_1 >, \qquad (2.15)$$

$$Y = < Y_8, Y_7, Y_6, Y_5, Y_4, Y_3, Y_2, Y_1 >, \qquad (2.16)$$

each can be represented by the sum of two numbers, namely,

$X = X_e + X_o$ and $Y = Y_e + Y_o$, which are defined as follows;

$$X_e = < X_8, 0, X_6, 0, X_4, 0, X_2, 0 >, \qquad (2.17a)$$

$$X_o = < 0, X_7, 0, X_5, 0, X_3, 0, X_1 >, \qquad (2.17b)$$

$$Y_e = < Y_8, 0, Y_6, 0, Y_4, 0, Y_2, 0 >, \qquad (2.18a)$$

$$Y_o = < 0, Y_7, 0, Y_5, 0, Y_3, 0, Y_1 >. \qquad (2.18b)$$

Consequently, the product $X \times Y$ can be written as

$$\begin{aligned} X \times Y &= (X_e + X_o)(Y_e + Y_o) = X_eY_e + X_eY_o + X_oY_e + X_oY_o \\ &= P_{ee} + P_{eo} + P_{oe} + P_{oo} \ . \qquad (2.19) \end{aligned}$$

Expanding these terms allows one to see the advantages of writing the multiplication in the form of Equation 2.19.

$$P_{ee} = (X_8Y_8) \text{ x } 2^{14} + 0 \text{ x } 2^{13} + (X_8Y_6+X_6Y_8) \text{ x } 2^{12} + 0 \text{ x } 2^{11} + (X_8Y_4+X_6Y_6+X_4Y_8) \text{ x } 2^{10} + 0 \text{ x } 2^{9} + (X_8Y_2+X_6Y_4+X_4Y_6+X_2Y_8) \text{ x } 2^{8} + 0 \text{ x } 2^{7} + (X_6Y_2+X_4Y_4+X_2Y_6) \text{ x } 2^{6} + 0 \text{ x } 2^{5} + (X_4Y_2+X_2Y_4) \text{ x } 2^{4} + 0 \text{ x } 2^{3} + (X_2Y_2) \text{ x } 2^{2} + 0 \text{ x } 2^{1} + 0 \text{ x } 2^{0} \tag{2.20}$$

$$P_{oo} = 0 \text{ x } 2^{14} + 0 \text{ x } 2^{13} + (X_7Y_7) \text{ x } 2^{12} + 0 \text{ x } 2^{11} + (X_5Y_7+X_7Y_5) \text{ x } 2^{10} + 0 \text{ x } 2^{9} + (X_3Y_7+X_5Y_5+X_7Y_3) \text{ x } 2^{8} + 0 \text{ x } 2^{7} + (X_1Y_7+X_3Y_5+X_5Y_3+X_7Y_1) \text{ x } 2^{6} + 0 \text{ x } 2^{5} + (X_1Y_5+X_3Y_3+X_5Y_1) \text{ x } 2^{4} + 0 \text{ x } 2^{3} + (X_1Y_3+X_3Y_1) \text{ x } 2^{2} + 0 \text{ x } 2^{1} + (X_1Y_1) \text{ x } 2^{0} \tag{2.21}$$

$$P_{eo} = 0 \text{ x } 2^{14} + (X_8Y_7) \text{ x } 2^{13} + 0 \text{ x } 2^{12} + (X_8Y_5+X_6Y_7) \text{ x } 2^{11} + 0 \text{ x } 2^{10} + (X_8Y_3+X_6Y_5+X_4Y_7) \text{ x } 2^{9} + 0 \text{ x } 2^{8} + (X_8Y_1+X_6Y_3+X_4Y_5+X_2Y_7) \text{ x } 2^{7} + 0 \text{ x } 2^{6} + (X_6Y_1+X_4Y_3+X_2Y_5) \text{ x } 2^{5} + 0 \text{ x } 2^{4} + (X_4Y_1+X_2Y_3) \text{ x } 2^{3} + 0 \text{ x } 2^{2} + (X_2Y_1) \text{ x } 2^{1} + 0 \text{ x } 2^{0} \tag{2.22}$$

$$P_{oe} = 0 \text{ x } 2^{14} + (X_7Y_8) \text{ x } 2^{13} + 0 \text{ x } 2^{12} + (X_5Y_8+X_7Y_6) \text{ x } 2^{11} + 0 \text{ x } 2^{10} + (X_3Y_8+X_5Y_6+X_7Y_4) \text{ x } 2^{9} + 0 \text{ x } 2^{8} + (X_1Y_8+X_3Y_6+X_5Y_4+X_7Y_2) \text{ x } 2^{7} + 0 \text{ x } 2^{6} + (X_1Y_6+X_3Y_4+X_5Y_2) \text{ x } 2^{5} + 0 \text{ x } 2^{4} + (X_1Y_4+X_3Y_2) \text{ x } 2^{3} + 0 \text{ x } 2^{2} + (X_1Y_2) \text{ x } 2^{1} + 0 \text{ x } 2^{0} \tag{2.23}$$

Note that the zero positions in the bit pattern alternate with non-zero summations. The zero position can be used to hold the carry from corresponding summation in the non-zero position. A full adder can be used to calculate each of the sums and the carry out of the full adder generates the bit in the zero positions. Here, we have considered $(X_2Y_8 \text{ x } 2^8)$ separately. Spares bits are collected together to form one or two distinct numbers. That is, we have $[(X_2Y_8) \times 2^8 + (X_2Y_7 + X_7Y_2) \times 2^7 + (X_7Y_1) \times 2^6]$. These numbers are treated separately. The propagation of carry is preserved in the body of multiplication and postponed at the last stage. This algorithm uses adder to convert three k-bit numbers to two $(k + 1)$-bit numbers. By using this technique, partial product numbers are, then, summed together via adder planes repeatedly to generate two distinct numbers. At the last stage the final two partial products are added by a fast adder to speed up multiplication operation. This approach is discussed in more details in Section 3.1.2.

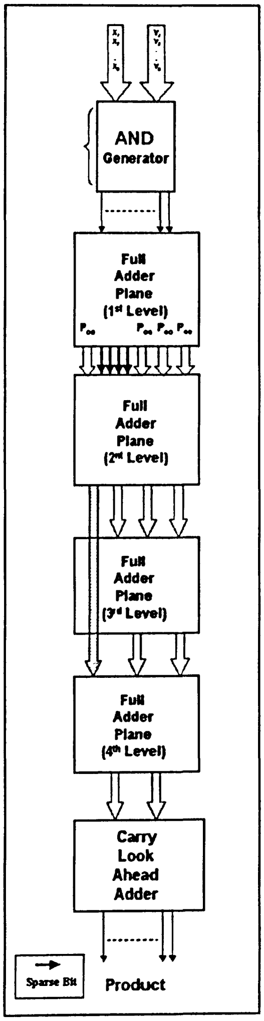


Fig. 2.9 Block diagram for the 8 x 8-bit pair-wise multiplier

2.4 Qualitative Comparisons of Parallel Algorithms

In order to choose the appropriate algorithm for the required applications one has to have a clear view of advantages and drawbacks of different algorithms that have been introduced. In the following a brief comparison of parallel algorithm is presented.

The basic array multipliers, such as the Baugh-Wooley scheme, consume low power and exhibit relatively good performance. However, they are limited to applications with the process operands with less than 16 bits. For operands of 16 bits and over, the modified Booth algorithm reduces the partial product's numbers by half and hence the speed of the multiplier is increased. In this case power consumption is comparable to that of Baugh-Wooley multiplier due to the circuitry overhead in Booth algorithm. However, by using circuit techniques one can make this multiplier have low-power characteristic. The fastest multipliers adopt the Wallace tree with modified Booth encoding. Due to its interconnecting wiring a Wallace tree would generally lead to larger power consumption and area. Hence, it is not recommended for low-power applications. Finally, the pair-wise multiplier shows faster operation by preventing the carry propagation in the intermediate stages of multiplication. This multiplication algorithm postpones the carry-propagation to the last stage where 2(n-1)-bit numbers are added. By using a fast addition circuitry such as carry lookahead adder (CLA) at the last stage of pair-wise multiplier one can accelerate the multiplication operation performance. Besides high-speed characteristic and simplicity of architecture of this algorithm, employing low power techniques [1] in circuit-level designs makes pair-wise algorithm a viable candidate for high performance multiplier. Baugh-Wooley algorithm is shown to be suitable for medium size (6 or 8) bit words [10]. It can be concluded that Baugh-Wooley multiplier is a suitable candidate to be used as test vehicle for the purpose of quantitative evaluation of pair-wise multiplier.

However, the entire Baugh-Wooley architecture should be redesign in order to perform a fair comparison.

Chapter 3

Multiplier Design

In this Chapter, the design of novel 8 x 8-bit multiplier is described in the circuit level. The building blocks are identified and the design of the cells based on these building blocks is, then, discussed. This Chapter begins with a brief description of some of the terms used hereafter in order to assess the circuits' performance.

Propagation delay of digital cells: duration from the moment that the first signal (50% transition point on input waveform) reaches the inputs of the cell to the moment that the last output signal (50% transition point on output waveforms) reaches the output nodes [21].

Power consumption of digital cells: The value of the power consumption of one cell is measured individually during testing the circuits. It means that the power consumed by the other cells in the test circuit is not included in the final measured value. This has been done by inserting a power meter in the form of Analog Hardware Description Language (AHDL) block in Cadence CAD tool in the route of the main supply to measure the power dissipation. This approach has been used as standard power measurement method throughout this work.

3.1 Pair-Wise Multiplier

Based on the pair-wise algorithm described in Chapter 2, the top level design of the proposed multiplier is built as shown in Fig 2.9. The following decisions were made in order to implement pair-wise algorithm. First, Due to inherent speed characteristic of pair-wise algorithm, a frequency of multiplication over 1GHz is targeted in this design. The power consumption of each element has been taken into account in topology selection. These points are discussed further in this Chapter where the circuit-level design of the proposed multiplier is reviewed. Also several low-power techniques are applied in layout extraction in order to achieve the power efficient design. These techniques are discussed in Section 5 of Chapter 4 where the layout considerations are reviewed.

3.1.1 Circuit-Level Review

In this Section first the elements required in pair-wise multiplier are introduced and then topology selection for the key elements is briefly presented. The architecture of the pair-wise multiplier (Fig. 2.9) shows that full adder, half adder, carry lookahead adder, AND, NAND, OR and XOR gates are the building blocks of the multiplier.

3.1.1.1 Full Adder

Full adder (FA) is the most critical circuit for two reasons. First, full adders cause a large percentage of the core propagation delay. Second, full adders ultimately consume the large percentage of power in the whole multiplier architecture. In order to select the best FA suited for high-performance application, a study was done on the existing FA circuits [2]. The result of this extensive study has directed to the selection the most speed/power efficient circuitry for the pair-wise multipliers. A summary of topology selection is provided next. First, note that the Boolean expression for a half adder (HA) is:

$$S = A \oplus B, \tag{3.1}$$

$$C_{out} = A.B, \tag{3.2}$$

and for full adder (FA) is:

$$S = A \oplus B \oplus C_{in}, \tag{3.3}$$

$$C_{out} = A.B + C_{in}(A \oplus B). \tag{3.4}$$

Table 3.1(a) Truth table of a full adder (b) Truth table of a half adder

A	B	C_{in}	Sum	C_{out}
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	1	0
0	1	1	0	1
0	1	0	1	0
0	0	1	0	1
0	0	0	0	0

A	B	Sum	C_{out}
1	1	1	1
1	0	1	0
0	1	1	0
0	0	0	0

The above Boolean expressions can be realized by different circuitries, each with their own advantages and disadvantages. In the following a brief review of the result of the study of six most well known CMOS full adder structures is presented. These adders have been compared in a wide range of static logic styles, which is viable candidate for low-power circuit design.

They include:

1. Complementary CMOS full adder cell
2. Complementary pass-transistor full adder cell
3. Double pass-transistor full adder cell
4. Transmission gate CMOS full adder cell
5. Pseudo-NMOS full adder cell
6. XOR and transmission gate full adder cell

The HA circuits are then generated from the optimized FAs by eliminating the circuitry which implements the function of the input carry.

Transistor Sizing: Sizing of the transistors in the full adder cells has been carried out in an iterative process consisted of the following steps.

1) Set all the transistors (NMOS and PMOS) to the minimum length (L_{min}) and the minimum width size (W_{min})
   (L_{min} =180nm, W_{min} = 660nm in 0.18μm CMOS process).
2) Simulate the circuit with all possible input pattern transitions (16 transitions).
3) Consider the transitions with the highest delay and mark the transistors involved in those transitions.
4) Size one of the transistors in this critical path.
5) Repeat Steps 2, 3 and 4 until the power-delay product for the cell continues to increase.
6) Record the transistor sizes corresponding to the minimum power-delay product.

This method guarantees that only the right transistors (in the critical path) are sized in a proper way. No over-sizing or under-sizing will be incurred, which makes it optimal for power-delay product performance. Although this is a lengthy process, it is guaranteed to give excellent transistor sizing results, especially for small circuits. Following the same method with larger circuits will take much longer.

It should be mentioned that the above transistor sizing method is a time consuming task for the structure such as double pass-transistor. This structure is already out of interest due to high numbers of transistors. Therefore, not much effort has been taken to optimize the size of the transistors for this adder.

Complementary CMOS full adder

Complementary CMOS full adder (CMOS) [15] has 28 transistors and its operation is based on the regular CMOS structure, pull-up & down networks (Fig.3.1). One of the advantages of the complementary CMOS full adder cell is high noise margins and thus, reliable operation at low voltages and arbitrary transistor sizes (ratio-less logic). The layout of CMOS gates is straightforward due to the complementary transistor pairs. An often mentioned, the disadvantage of complementary CMOS full adder cell is the substantial number of large PMOS transistors resulting in high input loads, more power consumption and larger silicon area. This adder uses C_{out} signal to generate Sum, which produces an unwanted additional delay. Another drawback of CMOS is the relatively weak output driving capability due to series transistors of the output stage.

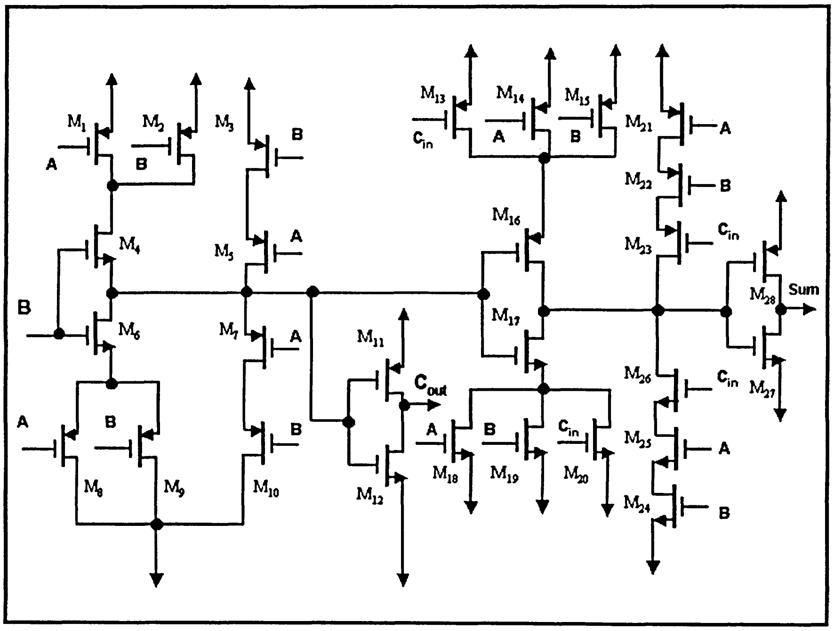


Fig. 3.1 Schematic of complementary CMOS full adder

Table 3.2 Transistor dimension in complementary CMOS full adder

MOST	W(μm)	L(μm)
M_1, M_2, M_3, M_4, M_5, M_{11}, M_{13} M_{14}, M_{15}, M_{16}, M_{21}, M_{22}, M_{23}, M_{28}	2.14	0.18
M_6, M_7, M_8, M_9, M_{10}, M_{12}, M_{17}, M_{18}, M_{19}, M_{20}, M_{24}, M_{25}, M_{26}, M_{27}	1.44	0.18
M_{24}, M_{25}	1.8	0.18

Complementary pass-transistor full adder

Complementary pass-transistor full adder cell has 32 transistors (Fig. 3.2). Using pass-transistor logic with CMOS inverters, this circuit features complementary inputs and outputs. This adder generates many intermediate nodes and their complements in order to generate the final signals (Sum and C_{out}). Having a signal and its complement together produces high rate of switching activities. Therefore, complementary pass-transistor full adder cell is not a suitable option for low power applications. In order to lower the power consumption of complementary pass-transistor, two circuit styles are used. These circuits have output levels restored with cross-coupled inverters [16] and latches [17].

Due to irregular transistor arrangements and high wiring requirement, layout of this full adder cell family is also not straightforward and efficient

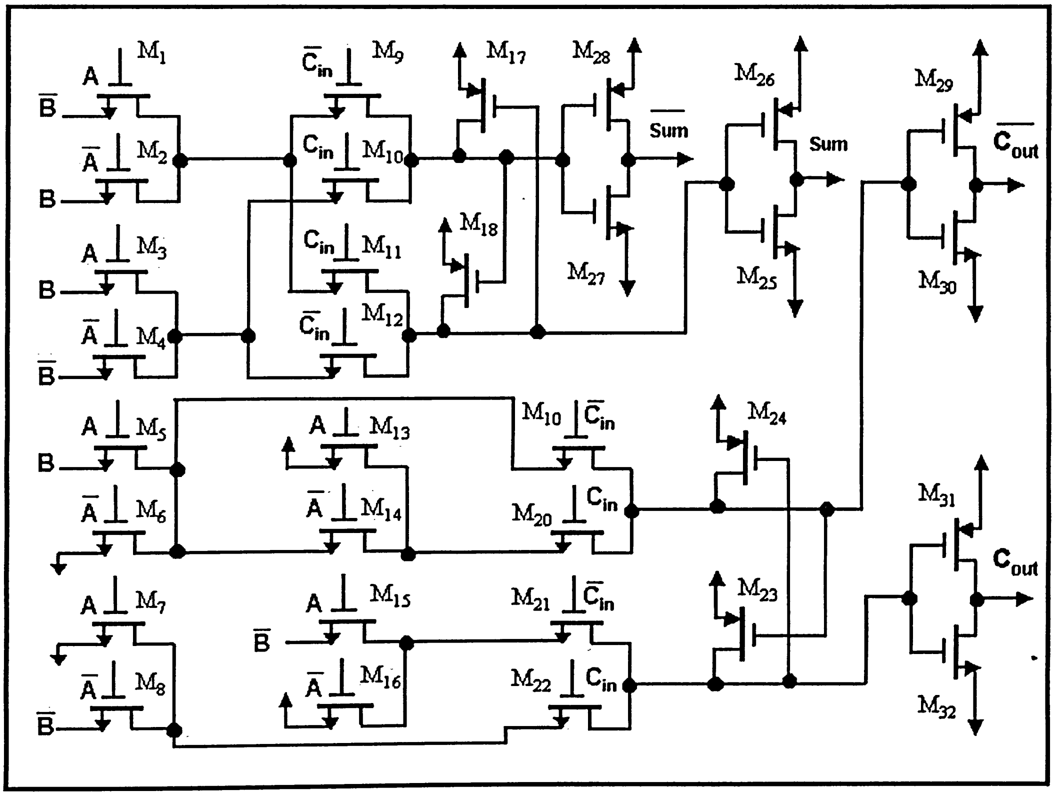


Fig. 3.2 Schematic of complementary pass-transistor full adder

Table 3.3 Transistor dimension in complementary pass-transistor full adder

MOST	W(µm)	L(µm)
M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, M_{10}, M_{11}, M_{12} M_{13}, M_{14}, M_{15}, M_{16}, M_{19}, M_{20}, M_{21}, M_{22}	7.2	1.8
M_{17}, M_{18}, M_{23}, M_{24}	9	1.8
M_{25}, M_{26}, M_{27}, M_{30}, M_{32}	14.4	1.8
M_{28}, M_{29}, M_{31}	18	1.8

Double pass-transistor full adder

Double pass-transistor full adder cell has 48 transistors and its operation is based on the double pass-transistor logic in which both NMOS and PMOS logic networks are used (Fig.3.3.a & b)[18]. The structure of this cell is similar to its complementary pass-transistor counterparts, but it uses complementary transistors to keep full swing operation and reduces the power consumption.

This eliminates the need for restoration circuitry. One disadvantage of this cell is the large area used due to the presence of PMOS transistors.

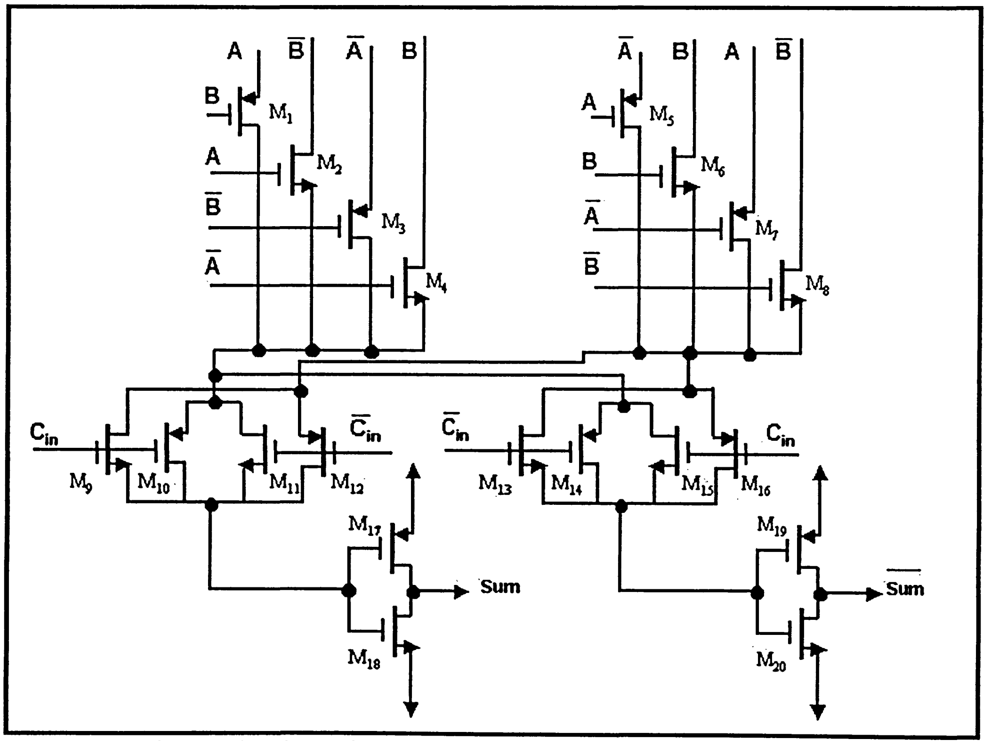


Fig. 3.3(a) Schematic of double pass-transistor full adder (Sum)

Table 3.4 Transistor dimension in double pass-transistor full adder (Sum)

MOST	W(μm)	L(μm)
M_2, M_4, M_6, M_8, M_9, M_{11}, M_{13}, M_{15}, M_{18}, M_{20}	0.77	0.18
M_1, M_3, M_5, M_7, M_{10}, M_{12}, M_{14}, M_{16}, M_{17}, M_{19}	1.08	0.18

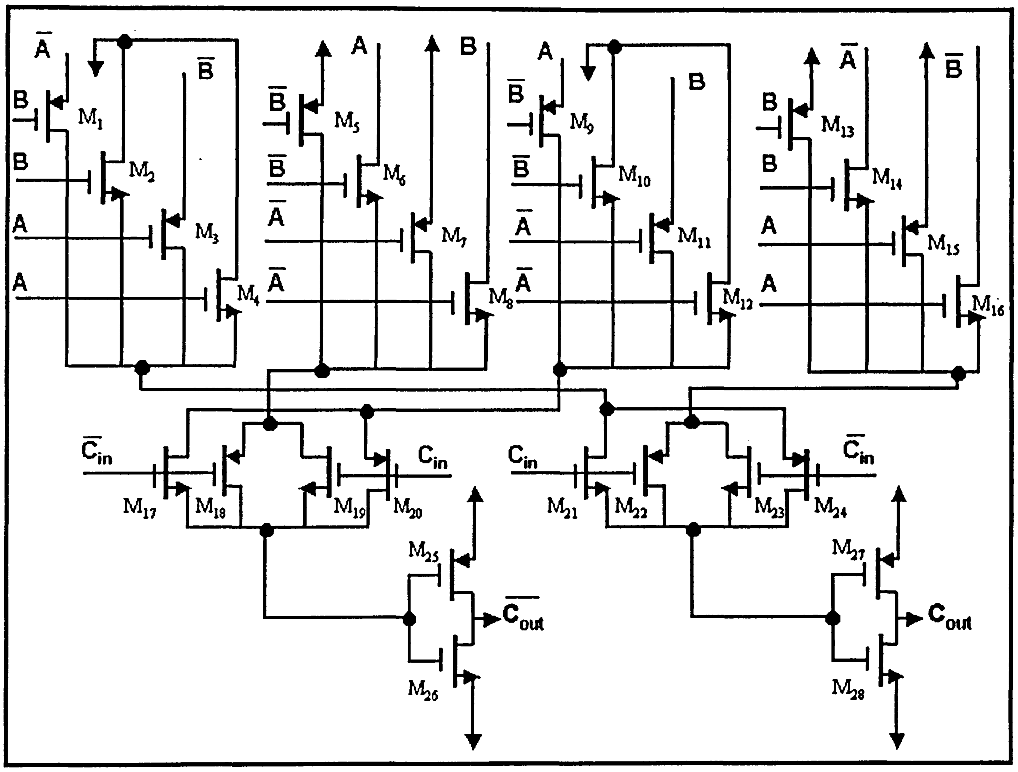


Fig. 3.3(b) Schematic of double pass-transistor full adder (C_{out})

Table 3.5 Transistor dimension in double pass-transistor full adder (C_{out})

MOST	W(μm)	L(μm)
M_2, M_4, M_6, M_8, M_{10}, M_{12}, M_{14}, M_{16}, M_{17}, M_{19}, M_{21} M_{23}, M_{26}, M_{28}	0.77	0.18
M_{18}, M_{20}, M_{22}, M_{24}	0.9	0.18
M_1, M_3, M_5, M_7, M_9, M_{11}, M_{13}, M_{15}, M_{25}, M_{27}	1.08	0.18

Transmission gate CMOS full adder

Transmission gate full adder has 20 transistors (Fig. 3.4). This circuit generates (A+B) and uses this and its complement as selected signals to generate the output signals (Sum & C_{out})[19]. It also requires complementary input signals (A, B, C_{in}) similar to the complementary CMOS full adder. However, it exhibits better speed than CMOS full adder with the same power consumption due to the small transistor stack height [20].

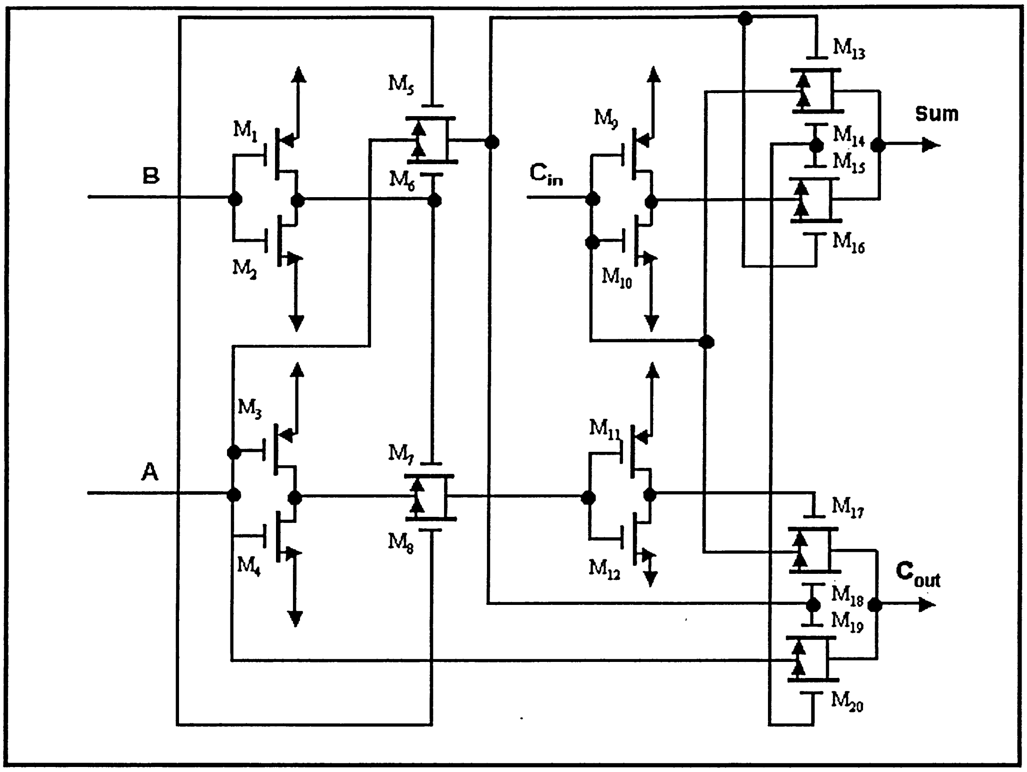


Fig. 3.4 Schematic of transmission gate full adder

Table 3.6 Transistor dimension in transmission gate full adder

MOST	W(μm)	L(μm)
M_2, M_4, M_6, M_8, M_{12}, M_{14}, M_{16}, M_{18}, M_{20}	0.7	0.18
M_5, M_7, M_{13}, M_{15}, M_{17}, M_{19}	0.9	0.18
M_1, M_3, M_{10}, M_{11}	1.44	0.18
M_9	1.8	0.18

Pseudo-NMOS full adder

Pseudo-NMOS full adder operates based on pseudo logic, referred to as ratioed style. This cell uses 14 transistors to realize the negative addition function (Fig. 3.5). The advantage of pseudo-NMOS adder is its higher speed (compared to complementary full adder) and low transistor count. On the negative side is the static power consumption of the pull-up transistor as well as the reduced output voltage swing, which makes this cell more susceptible to noise. In order to increase the output swing two CMOS inverters are added to this circuit, which increases the total transistors of this cell to 18 transistors.

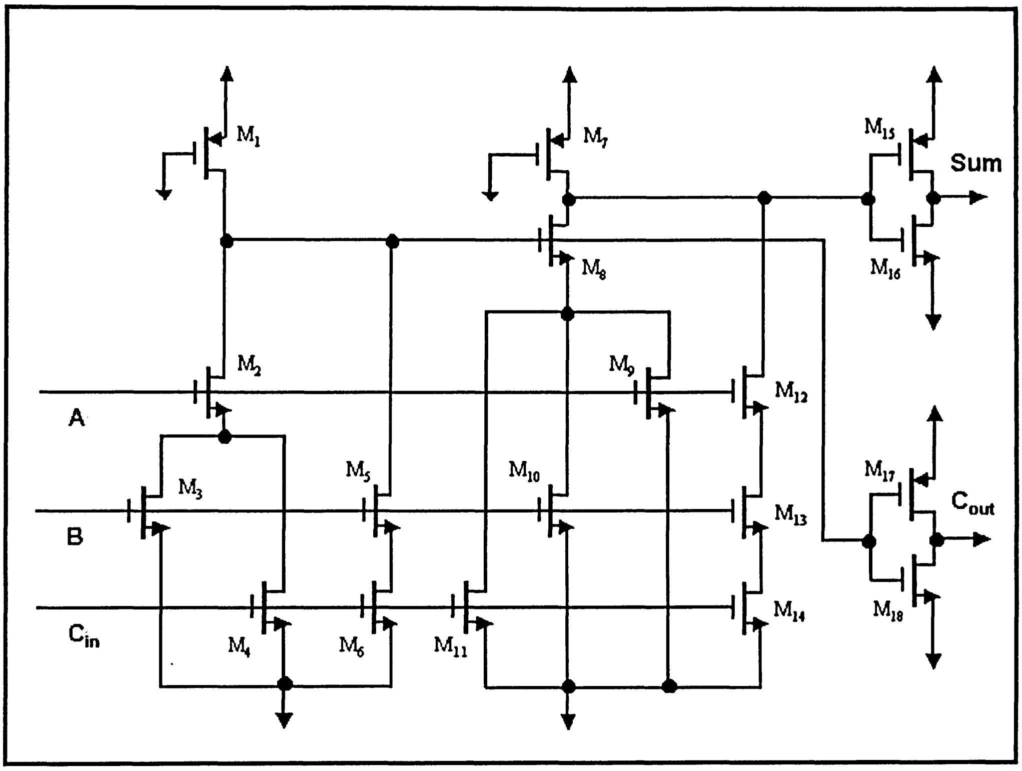


Fig. 3.5 Schematic of pseudo-NMOS full adder

Table 3.7 Transistor dimension in pseudo-NMOS full adder

MOST	W(µm)	L(µm)
$M_7, M_{12}, M_{13}, M_{14}$	0.66	0.18
$M_1, M_2, M_3, M_4, M_5, M_6, M_8$	0.77	0.18
$M_9, M_{10}, M_{11}, M_{16}, M_{18}$	1	0.18
M_{15}, M_{17}	2	0.18

XOR and transmission gate full adder

This adder shown in Fig. 3.6 has been developed based on an XOR gate [21] combined with transmission gate, which requires a total of 14 transistors [22]. XOR gate generates the sum. Using the transmission gate the second half of the circuit produces the carry out. This cell occupies less area compared with complementary CMOS full adder cell. In terms of power consumption this adder has a better performance. This is due to its low activity factor and passing a strong signal in fewer number of pass-logic gates, unlike the other cells where the signal had to go through more number of logic gates. Having discussed the high performance of this novel logic, one should note that the irregularity in layout of transmission gate and large average size of transistors are the considerable drawbacks of this circuit.

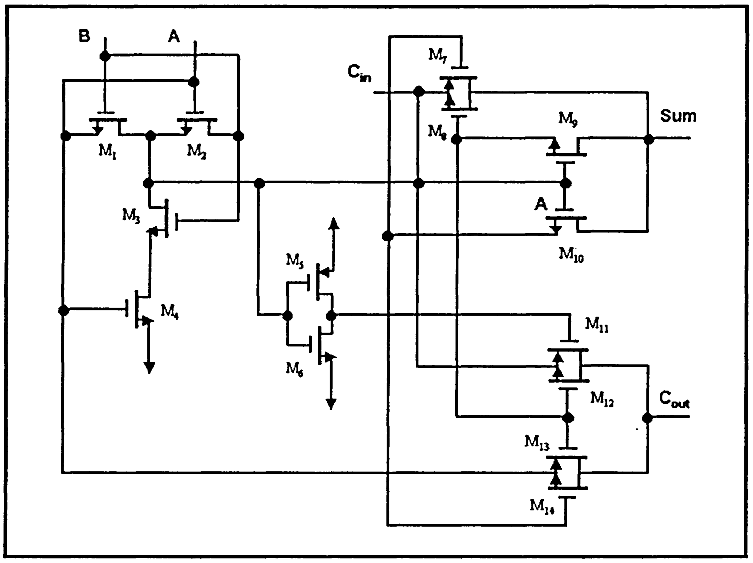


Fig. 3.6 Schematic of XOR & transmission gate full adder

Table 3.8 Transistor dimensions in XOR & transmission gate full adder

MOST	W(µm)	L(µm)
M_6, M_7, M_8, M_{10}	0.7	0.18
M_3, M_4, M_{12}, M_{14}	0.7	0.18
M_{11}, M_{13}	0.9	0.18
M_1, M_5, M_9	1.44	0.18
M_2	1.8	0.18

3.1.1.1.1 Simulation Strategies

In the following the techniques for simulations with regards to input patterns of full adder and output loading are presented.

Input Pattern and Output Loading: In order to compare different adders, input patterns should be in such ways that fairly test all cases. An input pattern which maximizes the power consumption for a given cell, could exhibit less power for another. While another input pattern could have the reversed situation due to different distribution of capacitances in both circuits.

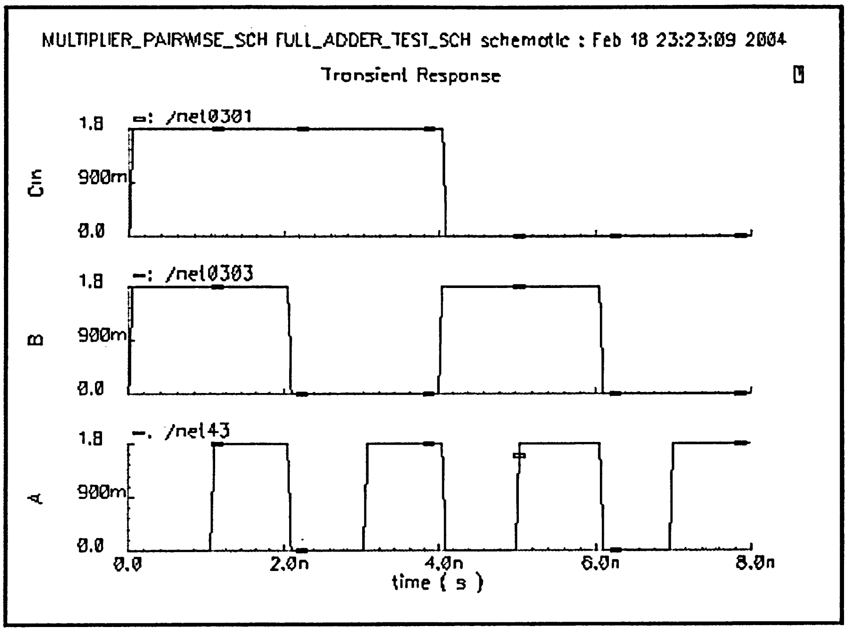


Fig. 3.7 (a) Input patterns used to evaluate the performance of the adders

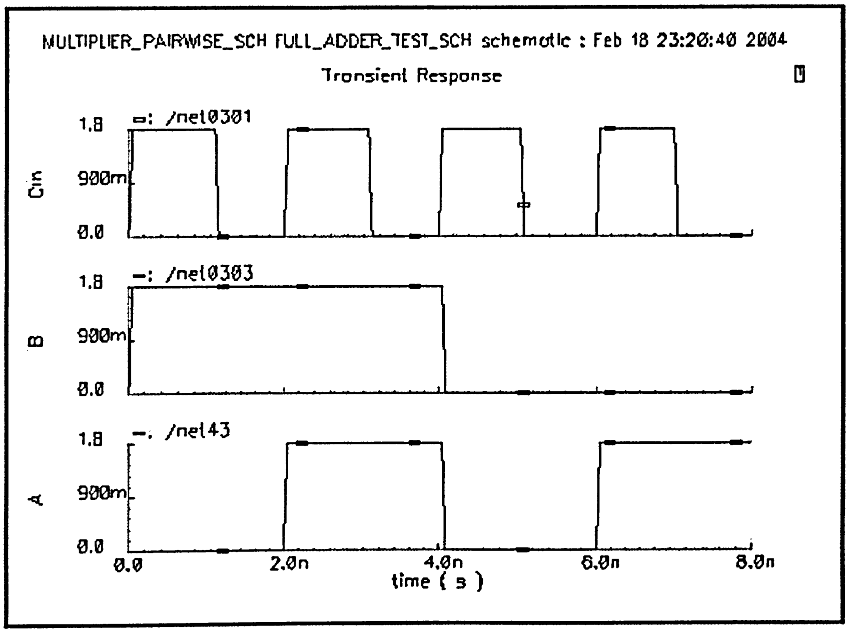


Fig. 3.7 (b) Input patterns used to evaluate the performance of the adders

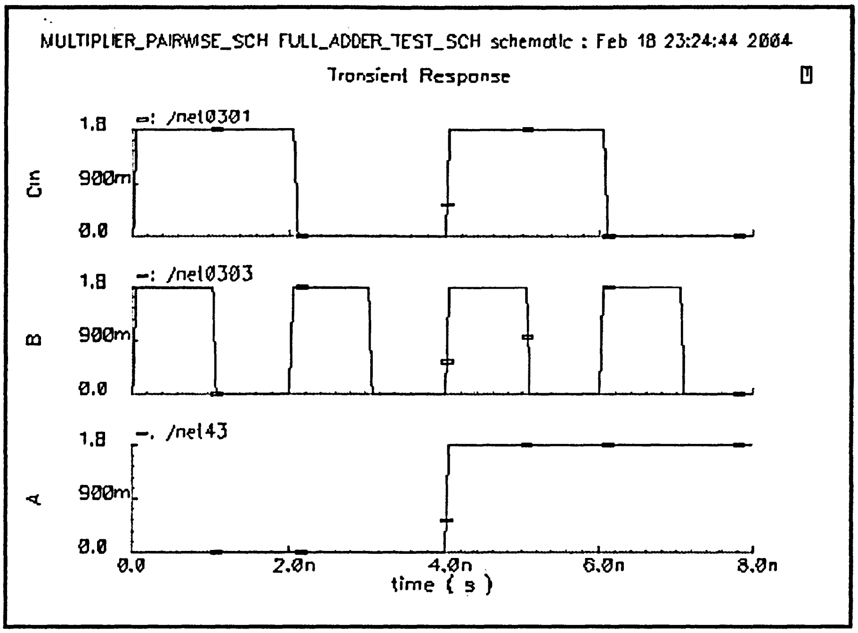


Fig. 3.7 (c) Input patterns used to evaluate the performance of the adders

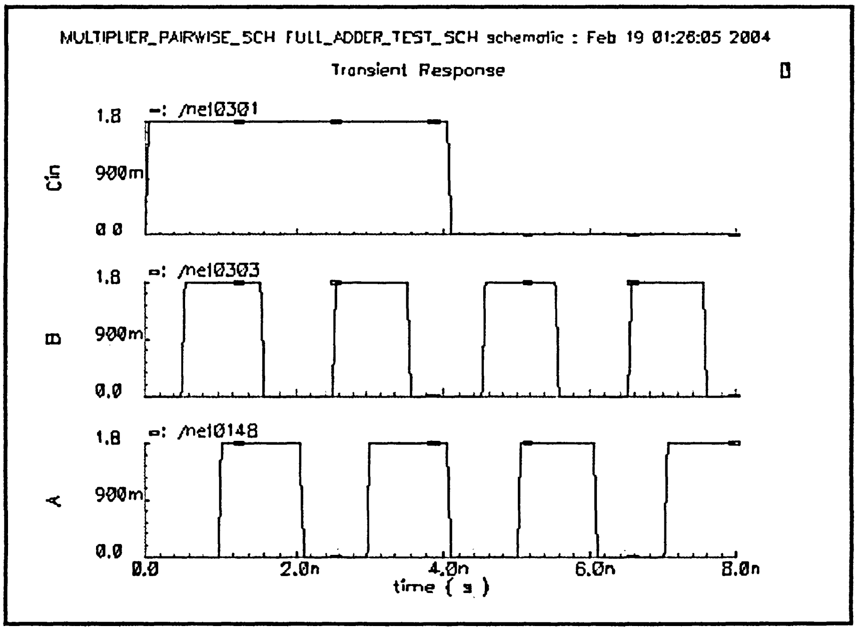


Fig. 3.7 (d) Input patterns used to evaluate the performance of the adders

A good input pattern for power consumption leading to a fair comparison of adder cells should alternate the high frequency at the input and intermediate nodes. A good example is the concatenation of the four patterns shown in Fig 3.7 (a, b, c, d).

Table 3.9 Characteristic of the input signals

Patterns	a		b		c		d	
Inputs	T(ns)	P.W. (ns)	T(ns)	P.W.(ns)	T(ns)	P.W.(ns)	T(ns)	P.W.(ns)
A	2	1	4	2	8	4	4	2
B	4	2	8	4	2	1	4	2
C_{in}	8	4	2	1	4	2	8	4

P.W. = Pulse width, T = Period, Rise time = 50ps, Fall time =50ps

As for speed, the input patterns should have all the required input-pattern-to-input-pattern transitions. The delay of the cell should be measured for each transition. The input pattern used for the simulation process is a concatenation of the four-input patterns shown in Fig. 3.7 (a, b, c, d).

The test bench used for simulating the adder cell is shown in Fig. 4.1 of Chapter 4, where the simulation result of the selected adder cell is discussed. The inputs are applied through buffers (two cascade inverters), which load adder cells with more realistic inputs in terms of slope and driving strength. Outputs are also applied to another adder to evaluate the driving capability of each cell.

3.1.1.1.2 Power Consumption Performance

Results of the comparison among adders, sorted by power consumption are shown in Table 3.10. The power performance of the second and third adder cells (Fig. 4.1) in the cascade configuration seems to be more realistic because in such a case, the high driving capability of the adder is a must in order to provide the next cell with the clean inputs. Therefore, the power values of either second or third full adder can be considered as the basis for our comparison. These results show that XOR and transmission gate full adder exhibit the lowest power consumption and transmission gate CMOS pseudo-NMOS, complementary CMOS, double pass-transistor and complementary pass-transistor are ranked respectively after it.

One can see that ranking is not necessarily related to the transistor count. It should be also pointed out that this evaluation corresponds to a 1.8 V power supply, and this point has slightly rearranged the previously reported adder ranking. The impact of supply reduction is an incomplete voltage swing at some internal nodes leading to a constant current drain. This, in turn, results in higher power consumption in circuits such as complementary pass-transistor and double pass-transistor.

Table 3.10 Simulation results for the full adders sorted by power consumption

Adder Cell (1.8V)	Power (mW)
XOR and transmission gate	0.0203
Transmission gate CMOS	0.0305
Pseudo-NMOS	0.0341
Complementary CMOS	0.0504
Double pass-transistor	0.0861
Complementary pass-transistor	0.0967

3.1.1.1.3 Delay Performance

The experimental results of the comparison among adders sorted by speed are presented in Table 3.11. The delay values are measured from the moment A, B and C_{in} signals reach the adder inputs till the last of the Sum and C_{out} signals reach the next adder cell inputs. The cell with the lowest-delay values is Complementary pass-transistor.

Table 3.11 Simulation results for the full adders sorted by propagation delay

Adder Cell	Delay (ns)
Complementary pass-transistor	0.057
XOR and transmission gate	0.066
Transmission gate CMOS	0.074
Pseudo-NMOS	0.080
Double pass-transistor	0.091
Complementary CMOS	0.140

Fig. 3.8 shows the delay of an adder. This measurement is based on the definition of the propagation delay of digital cells, explained at the beginning of this chapter. The inputs

signals are as $A = 1$, $B = 1$, and $C_{in} = 1$, therefore, the adder response will be as $Sum = 1$ and $C_{out} = 1$. Then, the delay between the earliest input signal (C_{in}) and Sum has been measured. The delay is also measured between C_{in} and C_{out}. This measurement has been performed at 50% transition point of the signals (which is 0.9 V in our case of $V_{dd} = 1.8$ V). The delay values of pseudo-NMOS adder are shown in Fig. 3.8. It can be seen that delay of *Sum* and C_{out} is very close in this cell, which avoids any data hazard, and race effects that may occur later in the proposed architecture.

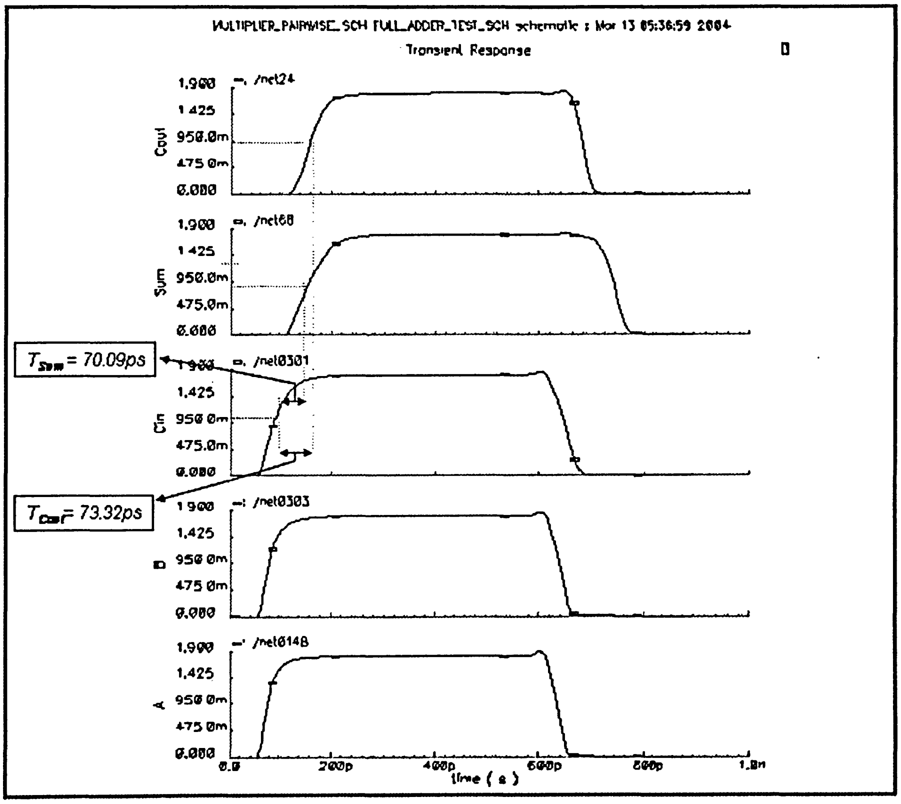


Fig. 3.8 Propagation delay measurement

3.1.1.1.4 Performance Comparisons

The following criteria have been considered in performing the comparison amongst different adder:

Power-delay product: The power-delay product is defined as a compromise between speed and power consumption. The values of the power-delay are presented in Table 3.12. The measurements are performed in identical conditions as it is recorded in Tables 3.10 and 3.11.

Area: The transistor count, showing area efficiency and layout productivity must be taken into account for choosing the best adder.

Table 3.12 Simulation results for the full adder cells sorted by power-delay product

Adder Cell	Power-delay product x10^{-12}	Transistor No.
XOR and transmission gate	0.00133	14
Transmission gate CMOS	0.00222	20
Pseudo-NMOS	0.00272	14
Complementary pass-transistor	0.00551	32
Complementary CMOS	0.00702	28
Double pass-transistor	0.00783	48

The measurement shows that pseudo-NMOS full adder has average values in both power consumption and delay, while providing a sum signal in good logic level. This leads to average of value in power-delay products.

Pseudo-NMOS adder also has small area occupancy not only due to the number of transistors but also because of the size of PMOSs, which are the main issue when it turns to layout extraction level. These properties make the pseudo-NMOS circuit amenable to use of a lower supply voltage to further reduce the power and at the same time maintaining a specific speed of the multiplication operation.

It is timely to mention that the comparison of the performance of the adder cells based on different logic is a very broad area of study and it is impossible to appreciate fully in a small section. Here, identical conditions such as uniform input pattern, capacitive load and constant V_{dd} have been used during simulation in order to achieve a fair comparison.

However, other factors such as selecting different geometry and physical designs and process variations could be considered as well.

3.1.1.2 Carry Lookahead Adder

The carry lookahead adder is a viable candidate to resolve the propagation delay problem by calculating the carry signal in advance based on the input signals. It relies on the fact that a carry signal will be generated in two cases:

a) when both input bits (A_i, B_i) are "1",

b) when one of the two bits is "1" and the C_{in} (carry-in of the previous stage) is "1".

Thus, one can write

$$C_{out} = C_{i+1} = A_i.B_i + (A_i \oplus B_i).C_i \ . \tag{3.5}$$

The above expression can be rewritten as

$$C_{i+1} = Gi + P_i.C_i, \tag{3.6}$$

in which

$$G_i = A_i.B_i, \tag{3.7}$$

$$P_i = (A_i \oplus B_i). \tag{3.8}$$

G_i and P_i are called generate and propagate terms, respectively [23].

Note that propagate and generate terms only depend on the input bits. If one uses the above expression to calculate the carry signal, s/he does not need to wait for the carry to ripple through all the previous stages to find its proper value. Thus, comes the main advantage of the carry lookahead adder: reducing the propagation delay.

In the following the generate and propagate terms are derived for a 4-bit adder.

$$C_1 = G_0 + P_0.C_0 \tag{3.9}$$

$$C_2 = G_1 + P_1.C_1 = G_1 + P_1.G_0 + P_1.P_0.C_0 \tag{3.10}$$

$$C_3 = G_2 + P_2.G_1 + P_2.P_1.G_0 + P_2.P_1.P_0.C_0 \tag{3.11}$$

$$C_4 = G_3 + P_3.G_2 + P_3.P_2.G_1 + P_3.P_2.C_1.G_0 + P_3.P_2.P_1.P_0.C_0 \tag{3.12}$$

Note that C_{out} bit and C_{i+1} of the last stage will be available after four delays (two gate delays to calculate propagate signal and two delays due to AND and OR gates). The sum signal (S_i) can be calculated as follows;

$$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus G_i. \tag{3.13}$$

Thus, the sum bit will be available after two additional gate delays (due to the XOR gate) or total of six gate delays after the input signals A_i and B_i have been applied. The advantage is that these delays will be the same and independent of the number of bits one needs to add, as opposed to the case of ripple counter.

The carry lookahead adder can be broken up in two modules;

1) The partial full adder, PFA, which generates G_i, P_i and S_i as defined by Equations 3.7, 3.8 and 3.13.

2) The carry lookahead logic, which generates C_{out} bits according to Equations 3.9 to 3.12. The 4-bit adder can then be built by using four PFAs and the carry lookahead logic block.

The disadvantage of carry lookahead adder is that the carry logic tends to get quite complicated for more than 4 bits. Therefore, carry lookahead adders are usually implemented as 4-bit modules and are used in a hierarchical structure to realize adders that have multiples of 4-bits. High fan-in OR gate is an unavoidable problem in designing a 16-bit carry lookahead adder. This is shown in Equation 3.12 when C_4 is calculated. Using high fan-in in logic gate would not only increase the propagation delay, but also contributes to additional power consumption. In order to resolve these issues the cascade of four 4-bit carry lookahead adders have been employed in design of 16-bit carry lookahead adder. The propagation delay of 16-bit carry lookahead adder in this architecture is approximately equal to that of the 4-bit ripple carry adder. This is because of C_{out} signals that have to ripple from one module to the next one. This is repeated four times until the final C_{out} arrives at the output. Despite the amount of delay, this approach is more power-efficient.

In the following the overview of the sub-cells of the 4-bit carry lookahead adder are described. Figure 3.9 shows the block diagram of 4-bit carry lookahead adder.

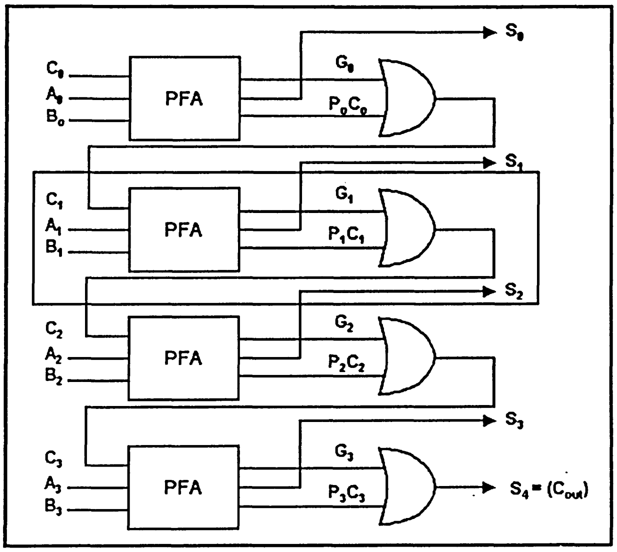


Fig. 3.9 Block diagram of 4-bit carry lookahead adder

As seen in Fig. 3.9 partial full adder (PFA) is the first block where inputs are fed. As it is mentioned earlier, this block generates, propagate, generate and sum signals. Fig. 3.10 shows the gate-level implementation of PFA. Sum signal is also generated in this block according to Equation 3.13. In order to generate C_{out} signal another XOR gate is needed.

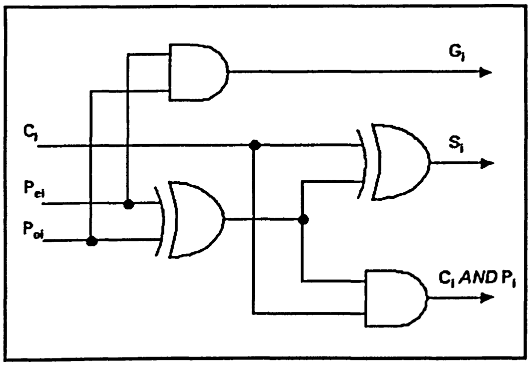


Fig. 3.10 Gate-level implementation of partial full adder (PFA)

The delays of signals in the highlighted block of carry lookahead adder (Fig. 3.9) are measured and shown in Table 3.13.

Table 3.13 Delay of the generate, propagate and sum signals of PFA

Output	Delay (ns)
G_i	0.0552
S_i	0.0385
C_iP_i	0.08

The block diagram of the 16-bit carry lookahead adder is shown in Fig. 3.11. Four 4-bit carry lookahead modules have been used to implement the final stage of the pair-wise multiplier. The labels on this diagram are based on the outputs of the previous stages.

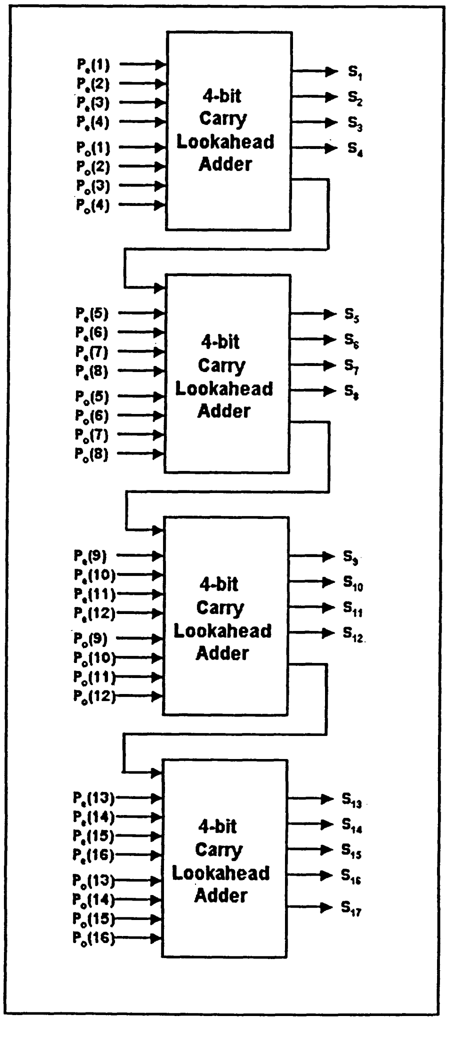


Fig. 3.11 Block diagram of the 16-bit carry lookahead adder implemented by cascading four 4-bit carry lookahead modules

3.1.1.3 AND, NAND, OR and XOR Gates

AND, NAND, OR and XOR are the fundamental logic gates, used in most logic circuits to realize the arithmetic operations. The Boolean expressions for two-input AND, NAND, OR and XOR gates, followed by their truth tables are shown in Table 3.14.

$$A.B, \tag{3.5}$$

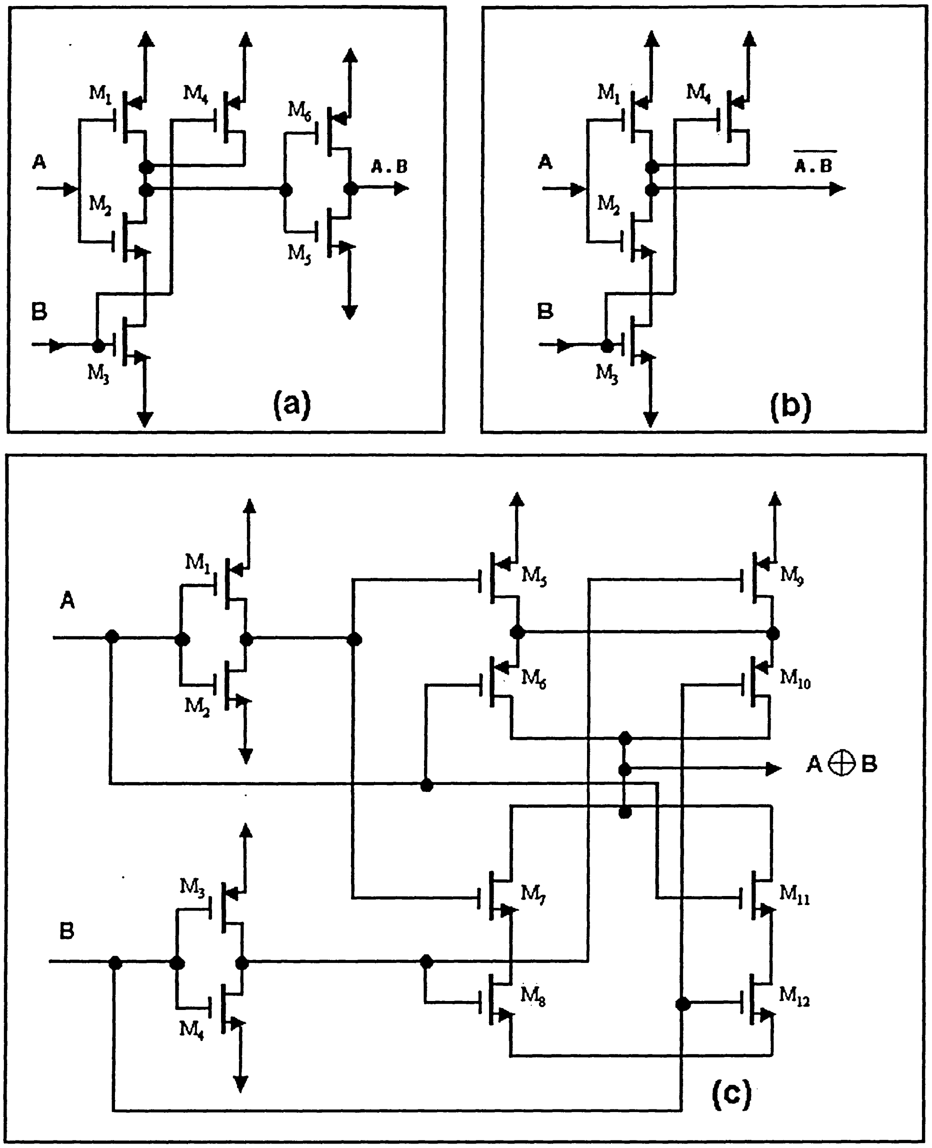
$$A + B, \tag{3.6}$$

$$A \oplus B. \tag{3.7}$$

Table 3.14 Truth table of AND, NAND, OR and XOR

A	B	$A.B$	$\overline{A.B}$	$A+B$	$A \oplus B$
1	1	1	0	1	0
1	0	0	1	1	1
0	1	0	1	1	1
0	0	0	1	0	0

The Boolean expressions representing AND/NAND/OR/XOR operation can be realized by different circuitries. However, the varieties of these structures are not as many as adder circuits. Therefore, very common configurations have been used to implement the required logic tasks. Figure 3.12(a, b) shows the schematics of AND, NAND gates that have been optimized for the required speed in the proposed multiplier [22]. NAND gate is composed of two NMOSs and PMOSs. An inverter is added to the circuit to generate the AND function. Several designs of OR and XOR gates have been reported. Each has its own advantages such as less delay and drawbacks such as poor response to some particular inputs [20]. Figure 3.12 (c, d) shows schematics of the OR and XOR circuits Dimension of NMOS and PMOS transistors have been modified for the required rise and fall times in the pair-wise multiplier.



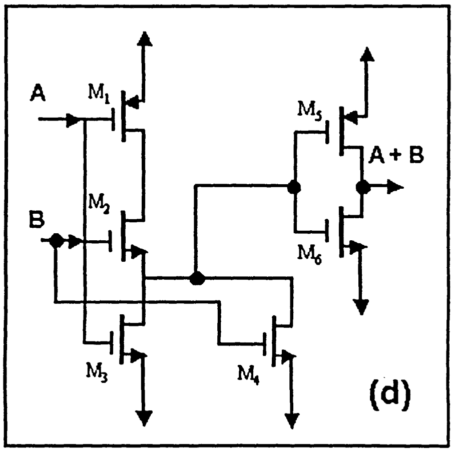


Fig. 3.12 Schematic of (a) AND (b) NAND (c) XOR (d) OR gates

Table 3.15 Transistor dimensions of AND, NAND, XOR and OR gates

AND Gate Fig 3.10 (a)		
MOST	**W (µm)**	**L (µm)**
M_1, M_4, M_2, M_3	1.4	0.18
M_5, M_6	1.8	0.18
NAND Gate Fig 3.10 (b)		
MOST	**W (µm)**	**L (µm)**
M_1, M_4, M_2, M_3	0.7	0.18
XOR Gate Fig 3.10 (c)		
MOST	**W (µm)**	**L (µm)**
M_1, M_3	3.5	0.18
M_2, M_4, M_{12}	1.5	0.18
$M_6, M_7, M_8, M_{10}, M_{11},$	2	0.18
M_5, M_9	3	0.18
OR Gate Fig 3.10 (d)		
MOST	**W (µm)**	**L (µm)**
M_1, M_2	2.2	0.18
M_3, M_4	0.7	0.18
M_5	2	0.18
M_6	0.75	0.18

3.1.2 Cell Design

In order to increase the productivity in ASIC design, cell design techniques are highly critical. In cell deign, a basic concept is to design uniform circuits that can perform the same task. In the following the top-level and the circuit-level of the required cells in each stage are described.

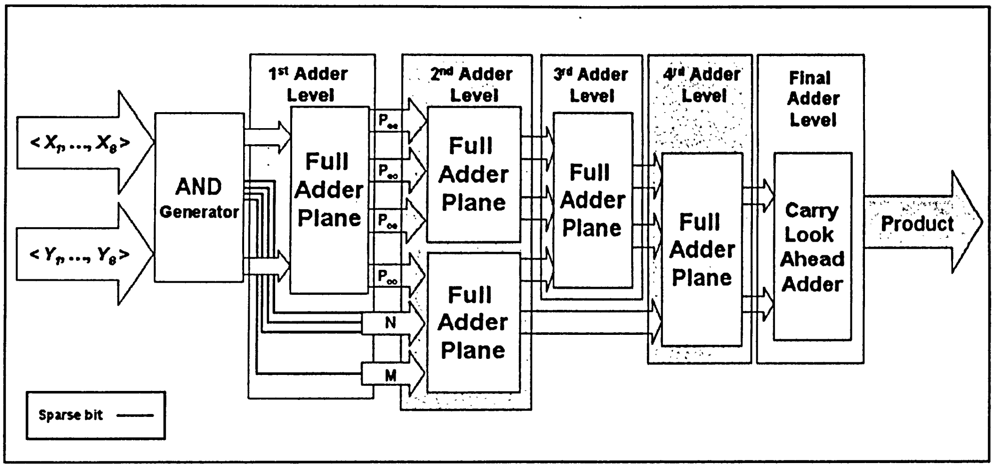


Fig. 3.13 Block diagram of the proposed 8 x 8-bit multiplier showing detail of the required cells

AND Generator: As seen in the block diagram of the pair-wise 8 x 8-bit multiplier (Fig. 3.13) the first stage of this architecture is an AND generator. In order to execute the first step of the pair-wise algorithm discussed in Section 2.3.5 AND combinations of all odd and even positions of two 8-bit multiplicand and multiplier are required. This task is performed by the AND generator. The block diagram of the AND generator is shown in Fig. 3.14. This stage consists of four AND planes known as:

X_eY_e: generating AND combination of all even bits of the both multiplicand and multiplier. The results are: X_2Y_2, X_2Y_4, X_2Y_6, X_2Y_8, X_4Y_2, X_4Y_4, X_4Y_6, X_4Y_8, X_6Y_2, X_6Y_4, X_6Y_6, X_6Y_8, X_8Y_2, X_8Y_4, X_8Y_6, X_8Y_8,.

X_eY_o: generating AND combination of even bits of the multiplicand and odd bits of the multiplier. The results are: X_2Y_1, X_2Y_3, X_2Y_5, X_2Y_7, X_4Y_1, X_4Y_3, X_4Y_5, X_4Y_7, X_6Y_1, X_6Y_3, X_6Y_5,

$X_6Y_7, X_8Y_1, X_8Y_3, X_8Y_5, X_8Y_7$.

X_oY_e: generating AND combination of odd bits of the multiplicand and even bits of the multiplier. The results are: X_1Y_2, X_1Y_4, X_1Y_6, X_1Y_8, X_3Y_2, X_3Y_4, X_3Y_6, X_3Y_8, X_5Y_2, X_5Y_4, X_5Y_6, X_5Y_8, X_7Y_2, X_7Y_4, X_7Y_6, X_7Y_8.

X_oY_o: generating AND combination of all odd bits of the both multiplicand and multiplier. The results are: X_1Y_1, X_1Y_3, X_1Y_5, X_1Y_7, X_3Y_1, X_3Y_3, X_3Y_5, X_3Y_7, X_5Y_1, X_5Y_3, X_5Y_5, X_5Y_7, X_7Y_1, X_7Y_3, X_7Y_5, X_7Y_7.

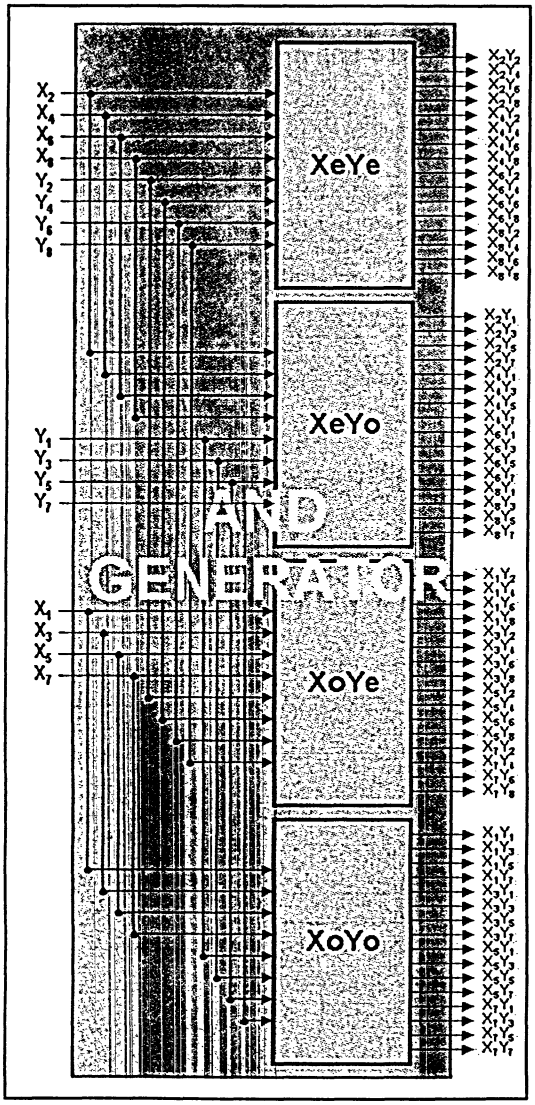


Fig. 3.14 Block diagram of AND generator

Fig 3.15 shows the gate-level implementation of the AND plane. Combination of four planes consequently constructs AND generation stage. The AND circuit discussed in Section 3.1.1.2 (Fig. 3.12a) is used in the circuit level.

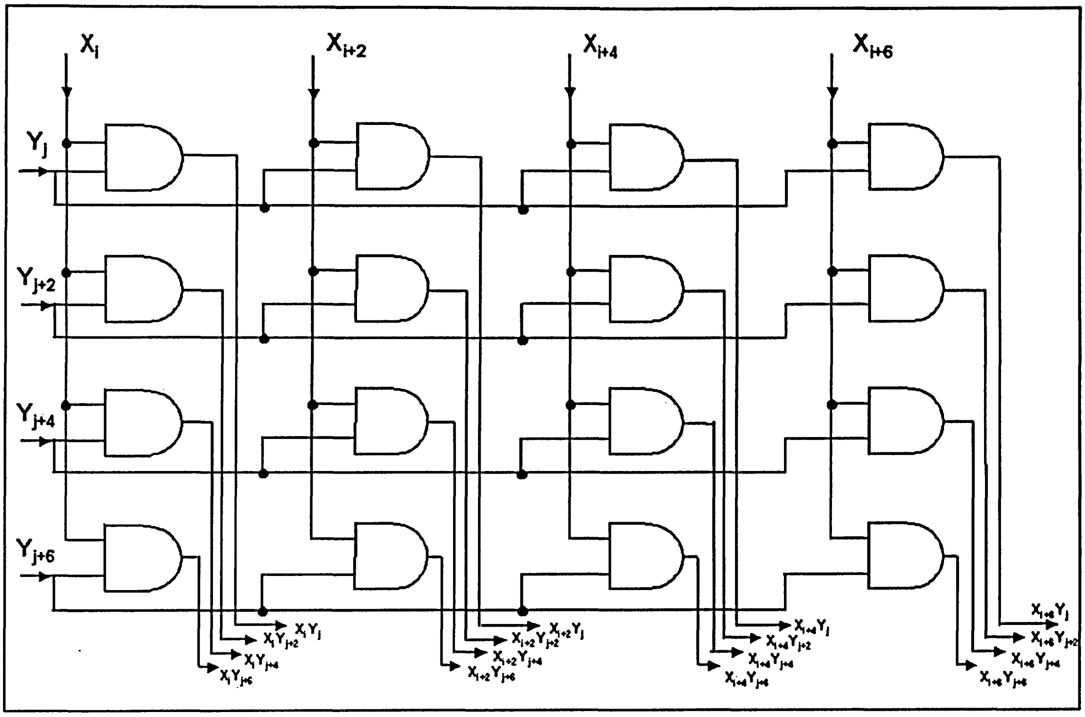


Fig. 3.15 Gate level of the AND plane (X_iY_j Cell)

First Adder Plane: The second stage of the multiplier is the first adder plane where partial products (P_{ee}, P_{eo}, P_{oe}, P_{oo}) are generated. Equations 2.20, 2.21, 2.22 and 2.23 show the different AND combinations of multiplicand and multipliers' bits required for generating each of partial products. Fig 3.16 shows the block diagram of this stage.

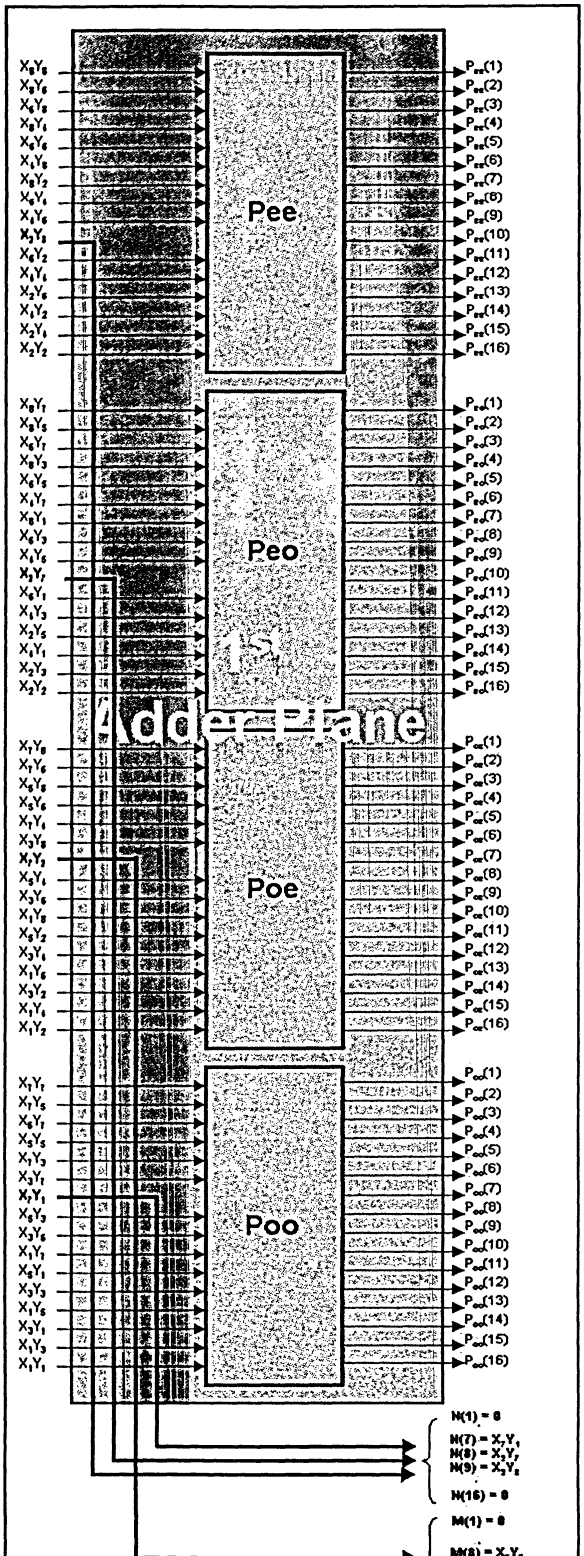


Fig. 3.16 Block diagram of partial products generator

This stage consists of four blocks of the partial product generators known as:

P_{ee}: generating partial products resulted by multiplication of the even bits of both multiplicand and multiplier. P_{ee} is a 15-bit number shown by bit number in parentheses as follows:

$P_{ee}(1) = 0$ $P_{ee}(2) = 0$ $P_{ee}(3) = X_2Y_2$ $P_{ee}(4) = 0$

$P_{ee}(5) = \text{Sum}\ [X_4Y_2 + X_2Y_4]$ $P_{ee}(6) = \text{C}_{\text{out}}\ [X_4Y_2 + X_2Y_4]$

$P_{ee}(7) = \text{Sum}\ [X_6Y_2 + X_4Y_4 + X_2Y_6]$ $P_{ee}(8) = \text{C}_{\text{out}}\ [X_6Y_2 + X_4Y_4 + X_2Y_6]$

$P_{ee}(9) = \text{Sum}\ [X_8Y_2 + X_6Y_4 + X_4Y_6 + \boldsymbol{X_2Y_8}]$ $P_{ee}(10) = \text{C}_{\text{out}}\ [X_8Y_2 + X_6Y_4 + X_4Y_6 + \boldsymbol{X_2Y_8}]$

$P_{ee}(11) = \text{Sum}\ [X_8Y_4 + X_6Y_6 + X_4Y_8]$ $P_{ee}(12) = \text{C}_{\text{out}}\ [X_8Y_4 + X_6Y_6 + X_4Y_8]$

$P_{ee}(13) = \text{Sum}\ [X_8Y_6 + X_6Y_8]$ $P_{ee}(14) = \text{C}_{\text{out}}\ [X_8Y_6 + X_6Y_8]$ $P_{ee}(15) = X_8Y_8$

P_{eo}: generating partial products resulted by multiplication of even bits of the multiplicand and odd bits of the multiplier. P_{eo} is a 15-bit number shown by bit number in parentheses as follows:

$P_{eo}(1) = 0$ $P_{eo}(2) = X_2Y_1$ $P_{eo}(3) = 0$ $P_{eo}(4) = \text{Sum}\ [X_4Y_1 + X_2Y_3]$

$P_{eo}(5) = \text{C}_{\text{out}}\ [X_4Y_1 + X_2Y_3]$ $P_{eo}(6) = \text{Sum}\ [X_6Y_1 + X_4Y_3 + X_2Y_5]$

$P_{eo}(7) = \text{C}_{\text{out}}\ [X_6Y_1 + X_4Y_3 + X_2Y_5]$ $P_{eo}(8) = \text{Sum}\ [X_8Y_1 + X_6Y_3 + X_4Y_5]$

$P_{eo}(9) = \text{C}_{\text{out}}\ [X_8Y_1 + X_6Y_3 + X_4Y_5 + \boldsymbol{X_2Y_7}]$ $P_{eo}(10) = \text{Sum}\ [X_8Y_3 + X_6Y_5 + X_4Y_7 + \boldsymbol{X_2Y_7}]$

$P_{eo}(11) = \text{C}_{\text{out}}\ [X_8Y_3 + X_6Y_5 + X_4Y_7]$ $P_{eo}(12) = \text{Sum}\ [X_8Y_5 + X_6Y_7]$

$P_{eo}(13) = \text{C}_{\text{out}}\ [X_8Y_5 + X_6Y_7]$ $P_{eo}(14) = X_8Y_7$

$P_{eo}(15) = 0$

P_{oe}: generating partial products resulted by multiplication of odd bits of the multiplicand and even bits of the multiplier. P_{oe} is a 15-bit number shown by bit number in parentheses as follows:

$P_{oe}(1) = 0$ $P_{oe}(2) = X_1Y_2$ $P_{oe}(3) = 0$ $P_{oe}(4) = \text{Sum}\ [X_1Y_4 + X_3Y_2]$

$P_{oe}(5) = \text{C}_{\text{out}}\ [X_1Y_4 + X_3Y_2]$ $P_{oe}(6) = \text{Sum}\ [X_1Y_6 + X_3Y_4 + X_5Y_2]$

$P_{oe}(7) = \text{C}_{\text{out}}\ [X_1Y_6 + X_3Y_4 + X_5Y_2]$ $P_{oe}(8) = \text{Sum}\ [X_1Y_8 + X_3Y_6 + X_5Y_4 + \boldsymbol{X_7Y_2}]$

$P_{oe}(9) = \mathrm{C_{out}}\ [X_1Y_8 + X_3Y_6 + X_5Y_4 + X_7Y_2]$ $P_{oe}(10) = \mathrm{Sum}\ [X_3Y_8 + X_5Y_6 + X_7Y_4]$

$P_{oe}(11) = \mathrm{C_{out}}\ [X_3Y_8 + X_5Y_6 + X_7Y_4]$ $P_{oe}(12) = \mathrm{Sum}\ [X_5Y_8 + X_7Y_6]$

$P_{oe}(13) = \mathrm{C_{out}}\ [X_5Y_8 + X_7Y_6]$ $P_{oe}(14) = X_7Y_8$ $P_{oe}(15) = 0$

P_{oo}: generating partial products resulted by multiplication of the odd bits of both multiplicand and multiplier. P_{oo} is a 15-bit number shown by bit number in parentheses as follow;

$P_{oo}(1) = X_1Y_1$ $P_{oo}(2) = 0$ $P_{oo}(3) = \mathrm{Sum}\ [X_1Y_3 + X_3Y_1]$

$P_{oo}(4) = \mathrm{C_{out}}\ [X_1Y_3 + X_3Y_1]$ $P_{oo}(5) = \mathrm{Sum}\ [X_1Y_5 + X_3Y_3 + X_5Y_1]$

$P_{oo}(6) = \mathrm{C_{out}}\ [X_1Y_5 + X_3Y_3 + X_5Y_1]$ $P_{oo}(7) = \mathrm{Sum}\ [X_1Y_7 + X_3Y_5 + X_5Y_3 + X_7Y_2]$

$P_{oo}(8) = \mathrm{C_{out}}\ [X_1Y_7 + X_3Y_5 + X_5Y_3 + X_7Y_2]$ $P_{oo}(9) = \mathrm{Sum}\ [X_3Y_7 + X_6Y_5 + X_7Y_3]$

$P_{oo}(10) = \mathrm{C_{out}}\ [X_3Y_7 + X_6Y_5 + X_7Y_3]$ $P_{oo}(11) = \mathrm{Sum}\ [X_5Y_7 + X_7Y_5]$

$P_{oo}(12) = \mathrm{C_{out}}\ [X_5Y_7 + X_7Y_5]$ $P_{oo}(13) = X_7Y_7$

$P_{oo}(14) = 0$ $P_{oo}(15) = 0$

All P_{ee}, P_{eo}, P_{oe}, P_{oo} blocks perform a similar task and have the same number of inputs and outputs. This makes it possible to employ the same cell for all four partial products (P_{ee}, P_{eo}, P_{oe}, P_{oo}) generators. This cell has been constructed by two half adders and five adders.

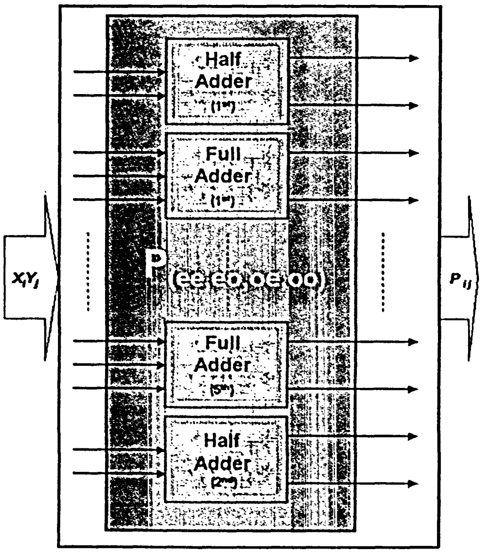


Fig. 3.17 Gate-level of one partial product generator (PP Cell)

Fig 3.17 shows the gate-level diagram of the partial product generator cell. In circuit-level pseudo-NMOS adder has been used to realize these cells. Using adder to convert three k-bit numbers to two (k + 1) numbers avoids the carry propagation delay in body of the multiplier. The following is description of this technique.

In order to use 3-to-2 adding technique it is necessary that not more than three inputs be used for generating any elements of partial product (P_{ij}). This condition is not met when the partial product elements are generated by four terms as it happens in $P_{ee}(9)$, $P_{ee}(10)$, $P_{eo}(9)$, $P_{eo}(10)$, $P_{oe}(8)$, $P_{oe}(9)$, $P_{oo}(7)$, $P_{oo}(8)$ (the fourth terms are highlighted in the relevant equation). To deal with these extra terms called spares terms they are taken out of the equations and collected together to form two distinct numbers which are called N and M. N(i) is a 15-bit number with zero in all even and odd positions except for the seventh [N(7)], eighth [N(8)] and ninth [N(9)] positions ($0,0,0,0,0,X_7Y_1,X_2Y_7,X_2Y_8,0,0,0,0,0,0$). M(i) is the second 15-bit number with zero in all even and odd positions except for the eighth position [M(8)] ($0,0,0,0,0,0,0, X_7Y_2,0,0,0,0,0,0,0$). These two numbers are shown in the block diagram of partial products generator (Fig. 3.16).

Now outputs of the first adder plane are six 15-bit numbers called (P_{ee}, P_{eo}, P_{oe}, P_{oo}, N, M).

Second & Third Adder Planes: In order to generate the final product of multiplication (P) of 8-bit X (multiplicand) and 8-bit Y (multiplier) all the individual partial products (P_{ee}, P_{eo}, P_{oe}, P_{oo}) generated from summation of even and odd bits of the multiplicand and multiplier and two distinct numbers generated by sparse bits (M, N) in the previous stage should be added together.

$$P_{ee} + P_{eo} + P_{oe} + P_{oo} + M + N = P \qquad (3.6)$$

This task requires the second and third adder planes. This addition operation has to be performed bit-by-bit resulting in carry out propagation. In order to postpone the carry propagation delay to the last stage of the multiplier a 3-to-2 adding technique has been used. To facilitate this technique adding of four partial products (P_{ee}, P_{eo}, P_{oe}, P_{oo}) and two extra numbers (M, N) is broken up into two steps as shown in Equation 3.7. These six numbers are divided to two batches of three numbers.

$$P_{ee} + P_{eo} + P_{oe} + P_{oo} + M + N = (P_{ee} + P_{eo} + P_{oe}) + (P_{oo} + M + N) \quad (3.7)$$

At this stage three 15-bit numbers (P_{ee},P_{eo},P_{oe}) are converted to two16-bit numbers (P_{eoeo}, P_{eoee}) and so are the fourth partial product (P_{oo}) and two distinct numbers (M, N) which generate ($P_{oo}S_e$, $P_{oo}S_o$).

This task can be performed by using a similar structure shown in Fig. 3.17 with a total of 14 adders. Note that due to the power and area constraints of the entire architecture using half adder is preferred whenever only two inputs signals need to be added (i.e. no C_{in} signal exits). The result of this 3-to-2 adding is shown as follows;

P_{eoeo}: result of adding all the odd positions of P_{ee}, P_{eo}, P_{oe}.

$P_{eoeo}(1) = 0 \qquad P_{eoeo}(2) = 0 \qquad P_{eoeo}(3) = P_{ee}(3) \qquad P_{eoeo}(4) = 0$

$P_{eoeo}(5) = \text{Sum}\,[P_{ee}(5) + P_{eo}(5) + P_{oe}(5)] \qquad P_{eoeo}(6) = \text{C}_{\text{out}}\,[P_{ee}(5) + P_{eo}(5) + P_{oe}(5)]$

$P_{eoeo}(7) = \text{Sum}\,[P_{ee}(7) + P_{eo}(7) + P_{oe}(7)] \qquad P_{eoeo}(8) = \text{C}_{\text{out}}\,[P_{ee}(7) + P_{eo}(7) + P_{oe}(7)]$

$P_{eoeo}(9) = \text{Sum}\,[P_{ee}(9) + P_{eo}(9) + P_{oe}(9)] \qquad P_{eoeo}(10) = \text{C}_{\text{out}}\,[P_{ee}(9) + P_{eo}(9) + P_{oe}(9)]$

$P_{eoeo}(11) = \text{Sum}\,[P_{ee}(11) + P_{eo}(11) + P_{oe}(11)] \qquad P_{eoeo}(12) = \text{C}_{\text{out}}[P_{ee}(11) + P_{eo}(11) + P_{oe}(11)]$

$P_{eoeo}(13) = \text{Sum}\,[P_{ee}(13) + P_{eo}(13) + P_{oe}(13)] \qquad P_{eoeo}(14) = \text{C}_{\text{out}}[P_{ee}(13) + P_{eo}(13) + P_{oe}(13)]$

$P_{eoeo}(15) = P_{ee}(15)$

P_{eoee}: result of adding all the even positions of P_{ee},, P_{eo} and P_{oe}.

$P_{eoee}(1) = 0$

$P_{eoee}(2) = \text{Sum}\,[P_{eo}(2) + P_{oe}(2)] \qquad P_{eoee}(3) = \text{C}_{\text{out}}\,[P_{eo}(2) + P_{oe}(2)]$

$P_{eoee}(4) = \text{Sum}\,[P_{eo}(4) + P_{oe}(4)] \qquad P_{eoee}(5) = \text{C}_{\text{out}}\,[P_{eo}(4) + P_{oe}(4)]$

$P_{eoee}(6) = \text{Sum}\,[P_{ee}(6) + P_{eo}(6) + P_{oe}(6)] \qquad P_{eoee}(7) = \text{C}_{\text{out}}\,[P_{ee}(6) + P_{eo}(6) + P_{oe}(6)]$

$P_{eoee}(8) = \text{Sum}\,[P_{ee}(8) + P_{eo}(8) + P_{oe}(8)] \qquad P_{eoee}(9) = \text{C}_{\text{out}}\,[P_{ee}(8) + P_{eo}(8) + P_{oe}(8)]$

$P_{eoee}(10) = \text{Sum}\ [P_{ee}(10) + P_{eo}(10) + P_{oe}(10)]$ $P_{eoee}(11) = \text{C}_{\text{out}}[P_{ee}(10)+P_{eo}(10)+P_{oe}(10)]$

$P_{eoee}(12) = \text{Sum}\ [P_{ee}(12) + P_{eo}(12) + P_{oe}(12)]$ $P_{eoee}(13) = \text{C}_{\text{out}}[P_{ee}(12)+P_{eo}(12)+P_{oe}(12)]$

$P_{oo}S_e$: result of adding all the even positions of P_{oo} , M and N.

$P_{oo}S_e(1) = 0$ $P_{oo}S_e(2) = 0$ $P_{oo}S_e(3) = 0$

$P_{oo}S_e(4) = P_{oo}(4)$ $P_{oo}S_e(5) = 0$ $P_{oo}S_e(6) = P_{oo}(6)$

$P_{oo}S_e(7) = 0$ $P_{oo}S_e(8) = \text{Sum}\ [P_{oo}(8) + X_2Y_7 + X_7Y_2]$

$P_{oo}S_e(9) = \text{C}_{\text{out}}\ [P_{oo}(8) + X_2Y_7 + X_7Y_2]$ $P_{oo}S_e(10) = P_{oo}(10)$

$P_{oo}S_e(11) = 0$ $P_{oo}S_e(12) = P_{oo}(12)$ $P_{oo}S_e(13) = 0$

$P_{oo}S_e(14) = P_{oo}(14)$ $P_{oo}S_e(15) = 0$

$P_{oo}S_o$: result of adding all the odd positions of P_{oo} , M and N.

$P_{oo}S_o(1) = P_{oo}(1)$ $P_{oo}S_o(2) = 0$ $P_{oo}S_o(3) = P_{oo}(3)$

$P_{oo}S_o(4) = 0$ $P_{oo}S_o(5) = P_{oo}(5)$ $P_{oo}S_o(6) = 0$

$P_{oo}S_o(7) = \text{Sum}\ [P_{oo}(7) + X_7Y_1]$ $P_{oo}S_o(8) = \text{C}_{\text{out}}\ [P_{oo}(7) + X_7Y_1]$

$P_{oo}S_o(9) = \text{Sum}\ [P_{oo}(9) + X_2Y_8]$ $P_{oo}S_o(10) = \text{C}_{\text{out}}\ [P_{oo}(10) + X_2Y_8]$

$P_{oo}S_o(11) = P_{oo}(11)$ $P_{oo}S_o(12) = 0$ $P_{oo}S_o(13) = P_{oo}(13)$

$P_{oo}S_o(14) = 0$ $P_{oo}S_o(15) = P_{oo}(15)$

Addition process is completed at this stage and four 16-bit numbers (P_{eoee}, P_{eoeo}, $P_{oo}S_e$, $P_{oo}S_o$) result of 3-to-2 addition of P_{ee},P_{eo},P_{oe}, M and N are the outputs of this level.

Equation 3.7 is rewritten as:

$$P_{ee} + P_{eo} + P_{oe} + P_{oo} + M + N = P_{eoee}, + P_{eoeo} + P_{oo}S_e + P_{oo}S_o. \qquad (3.8)$$

At the next stage addition of the four numbers is broken to two steps as shown in Equation 3.9.

$$P_{eoee}, + P_{eoeo} + P_{oo}S_e + P_{oo}S_o = (\ P_{eoee}, + P_{eoeo} + P_{oo}S_e) + P_{oo}S_o. \qquad (3.9)$$

The same technique as the previous stage is used two more times to convert the three numbers (P_{eoee},, P_{eoeo} , $P_{oo}S_e$) to two numbers ($PS1_e$, PSI_o) as following:

$PS1_e$: result of adding all the even positions of P_{eoee},, P_{eoeo} and $P_{oo}S_e$.

$PS1_e(1) = 0$

$PS1_e(2) = \text{Sum}\,[P_{eoee}(2) + P_{eoeo}(2) + P_{oo}S_e(2)]$

$PS1_e(3) = C_{\text{out}}\,[P_{eoee}(2) + P_{eoeo}(2) + P_{oo}S_e(2)]$

$PS1_e(4) = \text{Sum}\,[P_{eoee}(4) + P_{eoeo}(4) + P_{oo}S_e(4)]$

$PS1_e(5) = C_{\text{out}}\,[P_{eoee}(4) + P_{eoeo}(4) + P_{oo}S_e(4)]$

$PS1_e(6) = \text{Sum}\,[P_{eoee}(6) + P_{eoeo}(6) + P_{oo}S_e(6)]$

$PS1_e(7) = C_{\text{out}}\,[P_{eoee}(6) + P_{eoeo}(6) + P_{oo}S_e(6)]$

$PS1_e(8) = \text{Sum}\,[P_{eoee}(8) + P_{eoeo}(8) + P_{oo}S_e(8)]$

$PS1_e(9) = C_{\text{out}}\,[P_{eoee}(8) + P_{eoeo}(8) + P_{oo}S_e(8)]$

$PS1_e(10) = \text{Sum}\,[P_{eoee}(10) + P_{eoeo}(10) + P_{oo}S_e(10)]$

$PS1_e(11) = C_{\text{out}}\,[P_{eoee}(10) + P_{eoeo}(10) + P_{oo}S_e(10)]$

$PS1_e(12) = \text{Sum}\,[P_{eoee}(12) + P_{eoeo}(12) + P_{oo}S_e(12)]$

$PS1_e(13) = C_{\text{out}}\,[P_{eoee}(12) + P_{eoeo}(12) + P_{oo}S_e(12)]$

$PS1_e(14) = \text{Sum}\,[P_{eoee}(14) + P_{eoeo}(14) + P_{oo}S_e(14)]$

$PS1_e(15) = C_{\text{out}}\,[P_{eoee}(14) + P_{eoeo}(14) + P_{oo}S_e(14)]$

$\mathrm{PS1_o}$: result of adding all the odd positions of $\mathrm{P_{eoee.}}$, $\mathrm{P_{eoeo}}$ and $\mathrm{P_{oo}S_e}$

$PS1_o(1) = \text{Sum}\,[P_{eoee}(1) + P_{eoeo}(1) + P_{oo}S_e(1)]$

$PS1_o(2) = C_{\text{out}}\,[P_{eoee}(1) + P_{eoeo}(1) + P_{oo}S_e(1)]$

$PS1_o(3) = \text{Sum}\,[P_{eoee}(3) + P_{eoeo}(3) + P_{oo}S_e(3)]$

$PS1_o(4) = C_{\text{out}}\,[P_{eoee}(3) + P_{eoeo}(3) + P_{oo}S_e(3)]$

$PS1_o(5) = \text{Sum}\,[P_{eoee}(5) + P_{eoeo}(5) + P_{oo}S_e(5)]$

$PS1_o(6) = C_{\text{out}}\,[P_{eoee}(5) + P_{eoeo}(5) + P_{oo}S_e(5)]$

$PS1_o(7) = \text{Sum}\,[P_{eoee}(7) + P_{eoeo}(7) + P_{oo}S_e(7)]$

$PS1_o(8) = C_{\text{out}}\,[P_{eoee}(7) + P_{eoeo}(7) + P_{oo}S_e(7)]$

$PS1_o(9) = \text{Sum}\,[P_{eoee}(9) + P_{eoeo}(9) + P_{oo}S_e(9)]$

$PS1_o(10) = C_{\text{out}}\,[P_{eoee}(9) + P_{eoeo}(9) + P_{oo}S_e(9)]$

$PS1_o(11) = \text{Sum}\ [P_{eoee}(11) + P_{eoeo}(11) + P_{oo}S_e(11)]$

$PS1_o(12) = \text{C}_{\text{out}}\ [P_{eoee}(11) + P_{eoeo}(11) + P_{oo}S_e(11)]$

$PS1_o(13) = \text{Sum}\ [P_{eoee}(13) + P_{eoeo}(13) + P_{oo}S_e(13)]$

$PS1_o(14) = \text{C}_{\text{out}}\ [P_{eoee}(13) + P_{eoeo}(13) + P_{oo}S_e(13)]$

$PS1_o(15) = \text{Sum}\ [P_{eoee}(15) + P_{eoeo}(15) + P_{oo}S_e(15)]$

$PS1_o(16) = \text{C}_{\text{out}}\ [P_{eoee}(15) + P_{eoeo}(15) + P_{oo}S_e(15)]$

At the next parallel adder plane the two new words from previous adder plane ($PS1_c$, $PS1_o$) are added to ($P_{oo}S_o$) via another 3-to-2 adder stage to complete the Equation 3.9. This addition process is carried out similar to the one in the previous level. The three input numbers at this level are converted to two new 15-bit numbers called P_e and P_o. Arithmetic

P_e: result of adding all the even positions of $P_{oo}S_o$, PSI_e, PSI_o

$P_e(2) = \text{Sum}\ [P_{oo}S_o(2) + P_{eoeo}(2) + P_{eoee}(2)]$

$P_e(3) = \text{C}_{\text{out}}\ [P_{oo}S_o(2) + P_{eoeo}(2) + P_{eoee}(2)]$

$P_e(4) = \text{Sum}\ [P_{oo}S_o(4) + PSI_e(4) + PSI_o(4)]$

$P_e(5) = \text{C}_{\text{out}}\ [P_{oo}S_o(4) + PSI_e(4) + PSI_o(4)]$

$P_e(6) = \text{Sum}\ [P_{oo}S_o(6) + PSI_e(6) + PSI_o(6)]$

$P_e(7) = \text{C}_{\text{out}}\ [P_{oo}S_o(6) + PSI_e(6) + PSI_o(6)]$

$P_e(8) = \text{Sum}\ [P_{oo}S_o(8) + PSI_e(8) + PSI_o(8)]$

$P_e(9) = \text{C}_{\text{out}}\ [P_{oo}S_o(8) + PSI_e(8) + PSI_o(8)]$

$P_e(10) = \text{Sum}\ [P_{oo}S_o(10) + PSI_e(10) + PSI_o(10)]$

$P_e(11) = \text{C}_{\text{out}}[P_{oo}S_o(10) + PSI_e(10) + PSI_o(10)]$

$P_e(12) = \text{Sum}\ [P_{oo}S_o(12) + PSI_e(12) + PSI_o(12)]$

$P_e(13) = \text{C}_{\text{out}}[P_{oo}S_o(12) + PSI_e(12) + PSI_o(12)]$

$P_e(14) = \text{Sum}\ [P_{oo}S_o(14) + PSI_e(14) + PSI_o(14)]$

$P_e(15) = \text{C}_{\text{out}}[P_{oo}S_o(14) + PSI_e(14) + PSI_o(14)]$

P_o: result of adding all the odd positions of $P_{oo}S_o$, PSI_e, PSI_o

$P_o(1) = \text{Sum}\,[P_{oo}S_o(1) + PSI_e(1) + PSI_o(1)]$

$P_o(2) = C_{out}\,[P_{oo}S_o(1) + PSI_e(1) + PSI_o(1)]$

$P_o(3) = \text{Sum}\,[P_{oo}S_o(3) + PSI_e(3) + PSI_o(3)]$

$P_o(4) = C_{out}\,[P_{oo}S_o(3) + PSI_e(3) + PSI_o(3)]$

$P_o(5) = \text{Sum}\,[P_{oo}S_o(5) + PSI_e(5) + PSI_o(5)]$

$P_o(6) = C_{out}\,[P_{oo}S_o(5) + PSI_e(5) + PSI_o(5)]$

$P_o(7) = \text{Sum}\,[P_{oo}S_o(7) + PSI_e(7) + PSI_o(7)]$

$P_o(8) = C_{out}\,[P_{oo}S_o(7) + PSI_e(7) + PSI_o(7)]$

$P_o(9) = \text{Sum}\,[P_{oo}S_o(9) + PSI_e(9) + PSI_o(9)]$

$P_o(10) = C_{out}\,[P_{oo}S_o(9) + PSI_e(9) + PSI_o(9)]$

$P_o(11) = \text{Sum}\,[P_{oo}S_o(11) + PSI_e(11) + PSI_o(11)]$

$P_o(12) = C_{out}[P_{oo}S_o(11) + PSI_e(11) + PSI_o(11)]$

$P_o(13) = \text{Sum}\,[P_{oo}S_o(13) + PSI_e(13) + PSI_o(13)]$

$P_o(14) = C_{out}[P_{oo}S_o(13) + PSI_e(13) + PSI_o(13)]$

$P_o(15) = \text{Sum}\,[P_{oo}S_o(15) + PSI_e(15) + PSI_o(15)]$

$P_o(16) = C_{out}[P_{oo}S_o(15) + PSI_e(15) + PSI_o(15)]$

In the last stage of multiplication process, these two final numbers (P_e,P_o) need to be summed as it is shown in Equation 3.10. This equation is summarized form of Equation 3.6.

$$P_e + P_o = P \tag{3.10}$$

At the last step final two numbers (P_e, P_o) are simply added to generate the final product. This addition needs to be performed fast. Therefore, carry lookahead structure, known as a fast adder, has been used to speed up the multiplication.

In the next Chapter the simulation of the major block as well as the final simulation results are presented.

Chapter 4

Simulation Results & Layout Considerations

This Chapter presents the simulation results for the major designed cells and circuits. The simulation results of the final stage of the proposed multiplier for certain given inputs are further discussed. Layout considerations are explained later in this Chapter.

All circuits including individual cells and entire design have been simulated in Cadence environment.

4.1 Simulation Results of the Individual Circuits

Before presenting the simulation results of the individual circuit and designed cells, we need to introduce the circuit structure that have been used for simulation purposes. Arranging the proper test circuits has significant impact in increasing the ASIC productivity.

Simulation Circuit Structure: In regular multipliers such as the proposed architecture that uses full-adder cells as the building block, a cascade of full adders is usually utilized. In such cases, the high driving capability of adder is a must for providing the next cell with input signal with proper logic level. Having this point in mind, the circuit structure used to simulate the adder is illustrated in Fig. 4.1. A cascade of four full adder cells is utilized; the inputs are fed from buffers (two cascaded inverters) to give more realistic signals and outputs are loaded with buffers to give proper loading conditions [28].

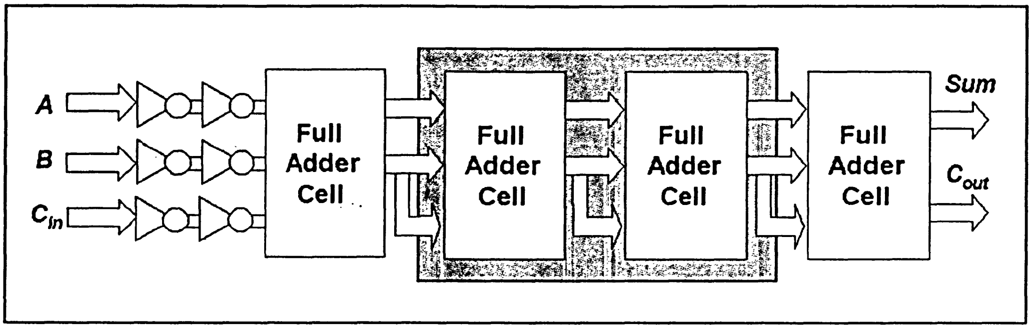


Fig. 4.1 Circuit structure used for simulation of full adder cell

The parasitic effects are, therefore, included in the simulation results. The same structure has been used to compare the adder cells discussed in topology selection.

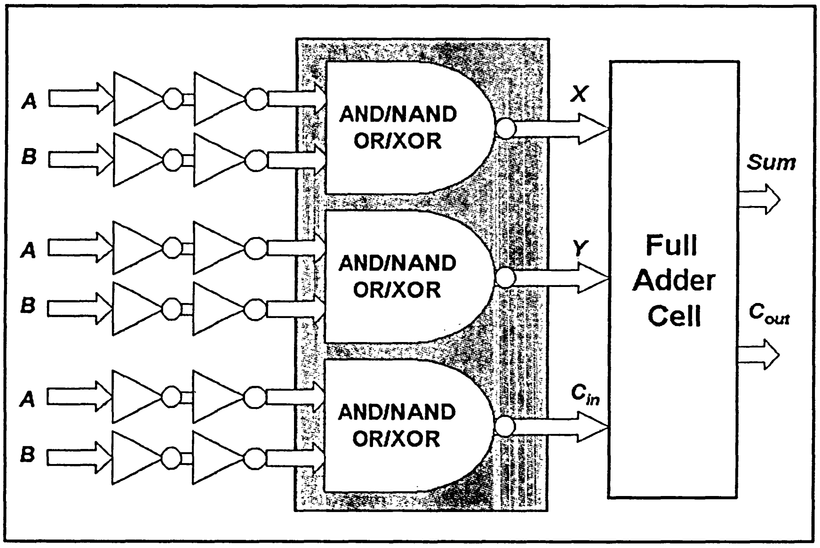


Fig. 4.2 Circuit structure used for simulation of AND/NAND/OR/XOR gates

Full Adder Simulation Results

Here are the simulation results for pseudo-NMOS full adder using the test circuit structure of Fig. 4.1 corresponding to four different input patterns. The input patterns were already introduced in Chapter 3 when describing simulation strategy of adder cells. These patterns fairly cover most of the possible input combinations.

Fig. 4.3 shows *Sum* and C_{out} signals of pseudo-NMOS full adder to input pattern shown in Fig. 3.7 (a). This pattern covers 6 transitions of the input signals (*A*, *B*, C_{in}). These transitions are also shown in Table 4.1 corresponding to those in Fig. 4.3.

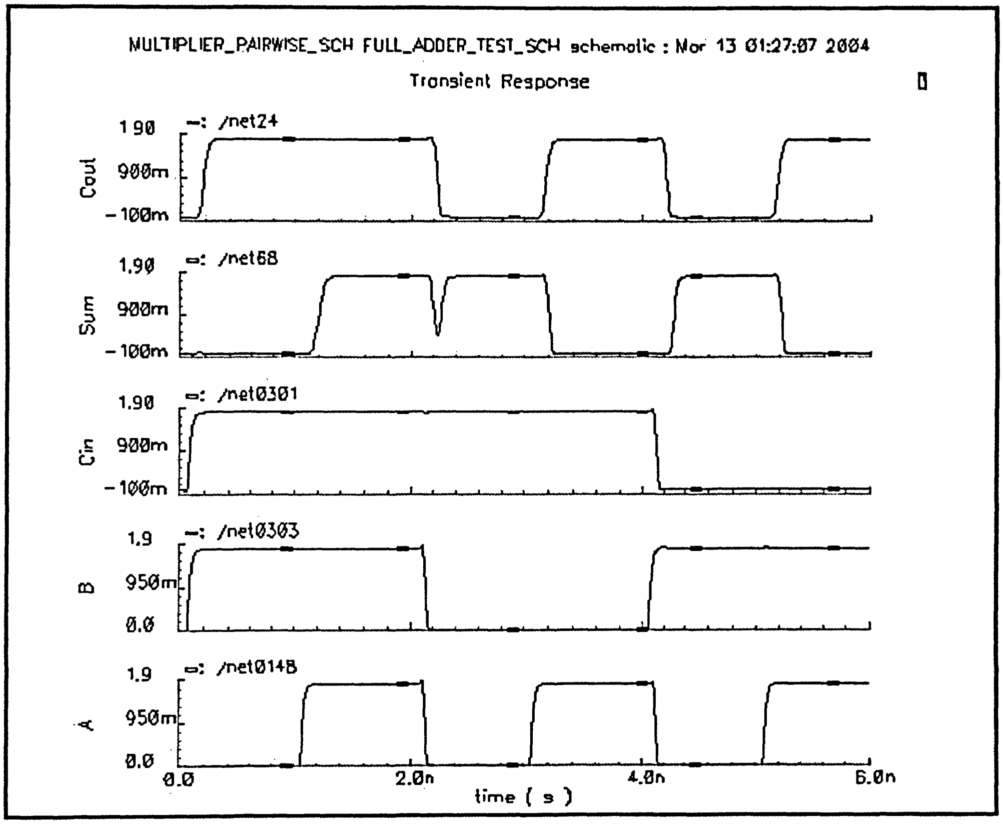


Fig. 4.3 The simulation waveforms showing respond of the pseudo full adder to the input pattern (a)

Table 4.1 Transitions covered by input pattern (a)

Inputs			**Outputs**	
A	*B*	C_{in}	*Sum*	C_{out}
0	1	1	0	1
1	1	1	1	1
0	0	1	1	0
1	0	1	0	1
0	1	0	1	0
1	1	0	0	1

Fig. 4.4 shows *Sum* and C_{out} signals of pseudo-NMOS full adder to input pattern shown in Fig 3.7 (b). This pattern covers 6 transitions of the input signals (A, B, C_{in}). These transitions are shown in Table 4.2 corresponding to those in Fig. 4.4.

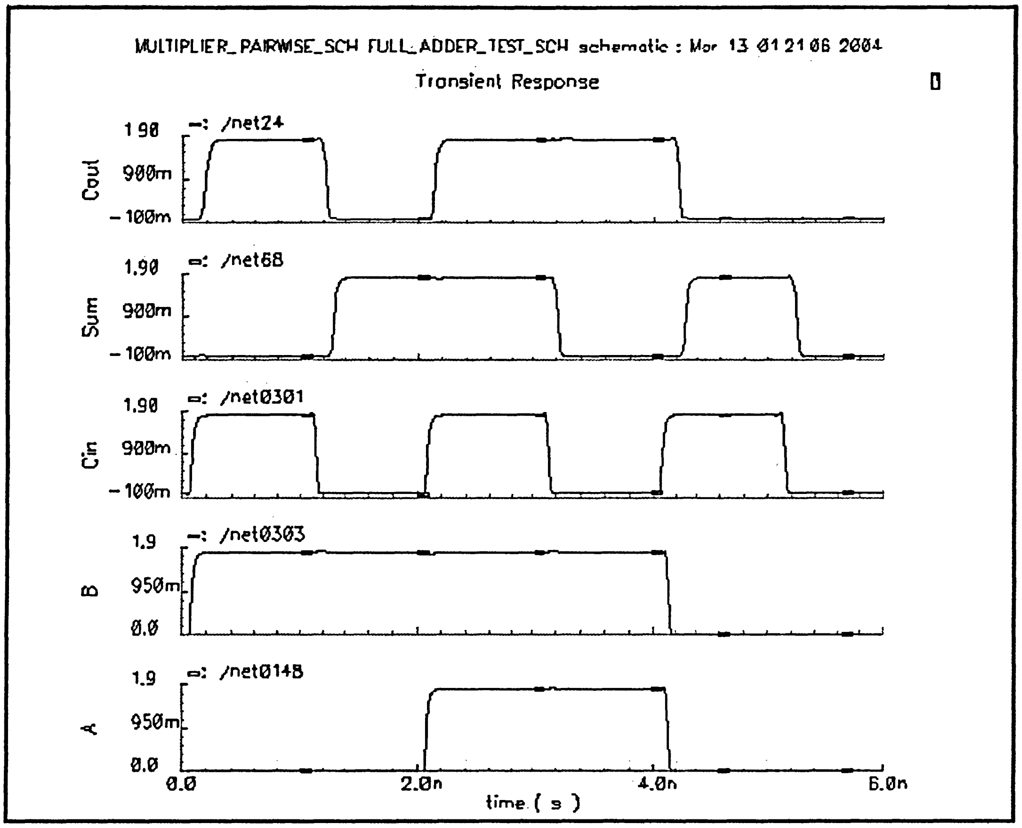


Fig. 4.4 The simulation waveforms showing respond of the pseudo full adder to the input pattern (b)

Table 4.2 Transitions covered by input pattern (b)

Inputs			**Outputs**	
A	*B*	C_{in}	*Sum*	C_{out}
0	1	1	0	1
0	1	0	1	0
1	1	1	1	1
1	1	0	0	1
0	0	1	1	0
0	0	0	0	0

Fig. 4.5 shows *Sum* and C_{out} signals of pseudo-NMOS full adder to input pattern shown in Fig. 3.7 (c). This pattern covers 6 transitions of the input signals (A, B, C_{in}). These transitions are shown in Table 4.3 corresponding to those in Fig. 4.5.

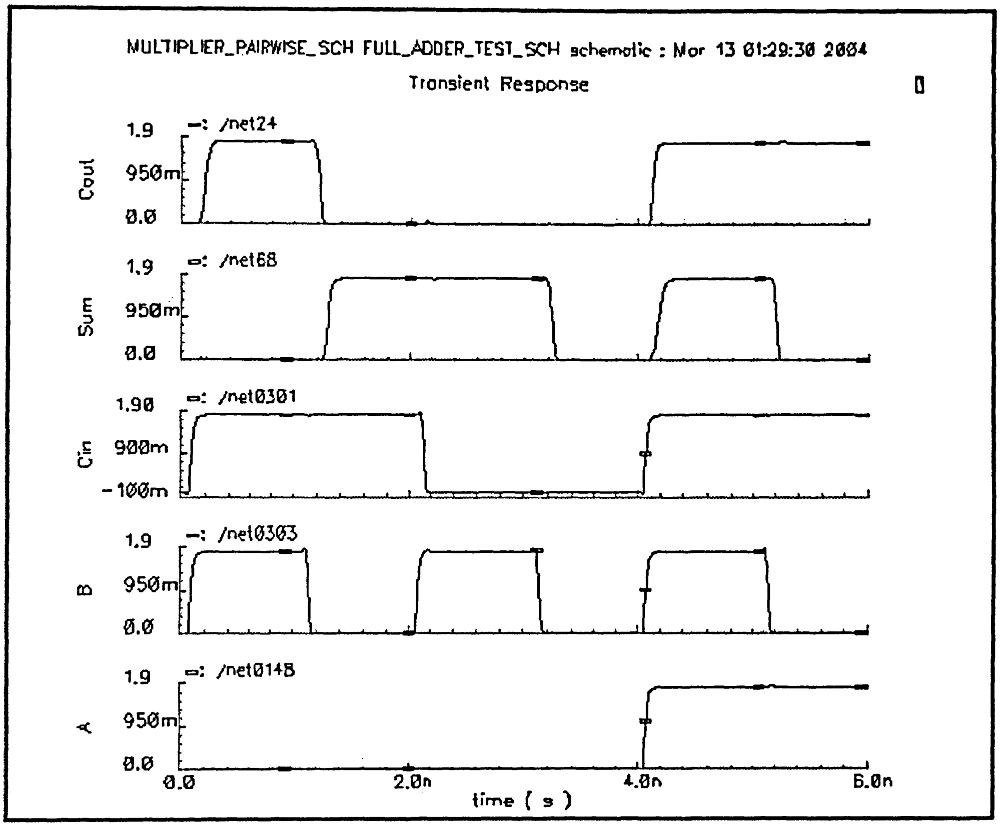


Fig. 4.5 The simulation waveforms showing respond of the pseudo full adder to the input pattern (c)

Table 4.3 Transitions covered by input pattern (c)

Inputs			Outputs	
A	B	C_{in}	*Sum*	C_{out}
0	1	1	0	1
0	0	1	1	0
0	1	0	1	0
0	0	0	0	0
1	1	1	1	1
1	0	1	0	1

Fig. 4.6 shows the *Sum* and C_{out} signals of pseudo-NMOS full adder to input pattern discussed in Fig. 3.7 (d). This pattern covers 6 transitions of the input signals (*A, B,* C_{in}). These transitions are shown in Table 4.4 respectively as it is seen in Fig. 4.6.

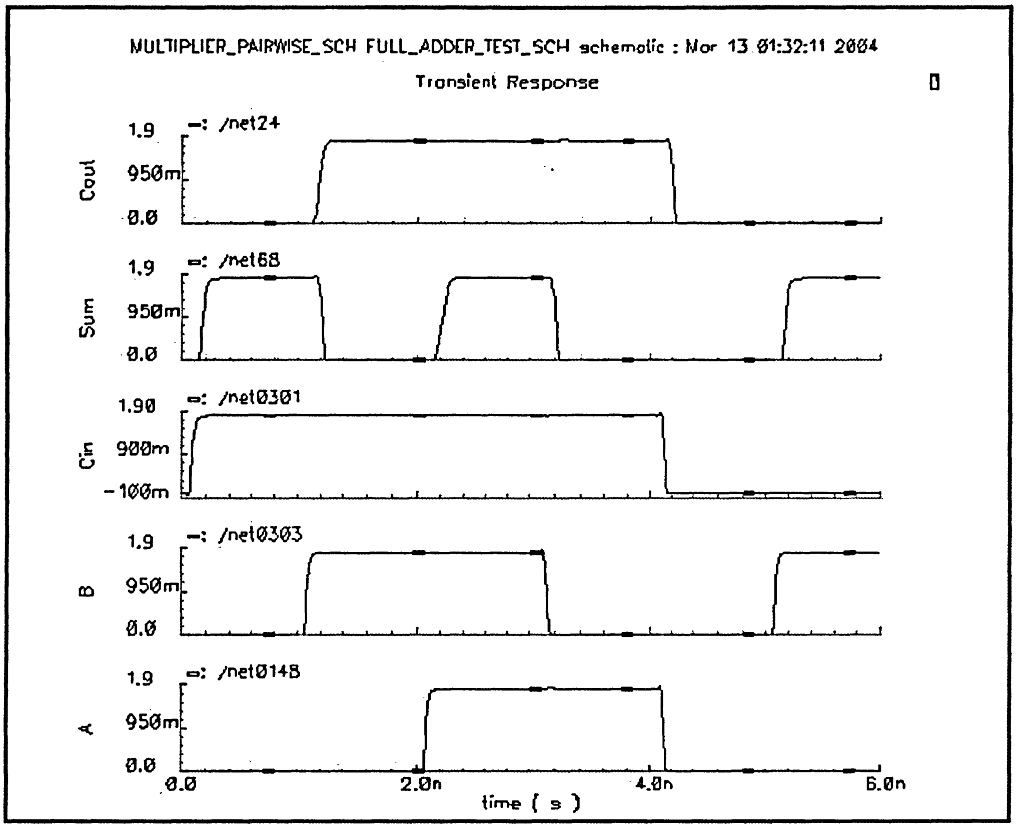


Fig. 4.6 The simulation waveforms showing respond of the pseudo full adder to the input pattern (d)

Table 4.4 Transitions covered by input pattern (d)

Inputs			Outputs	
A	*B*	C_{in}	*Sum*	C_{out}
0	0	1	1	0
0	1	1	0	1
1	1	1	1	1
1	0	1	0	1
0	0	0	0	0
0	1	0	1	0

AND/NAND/OR/XOR Gates Simulation Results

AND/NAND/OR/XOR gates have been described in more details in Chapter 3. Schematics are shown in Fig. 3.15. The test structure used for the simulation is shown in Fig. 4.2. The input signals have 50% duty cycles with period of 2ns. Figures 4.7 to 4.9 show the results of the simulation for these gates.

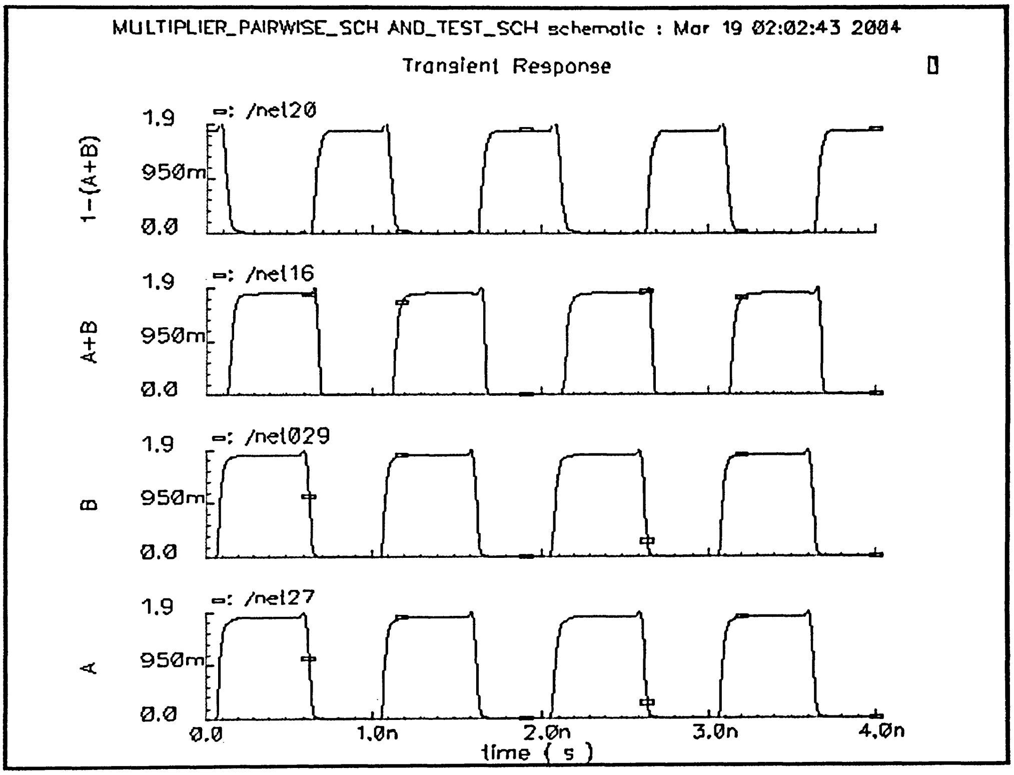


Fig. 4.7 The simulation waveforms showing respond of the AND/NAND gate

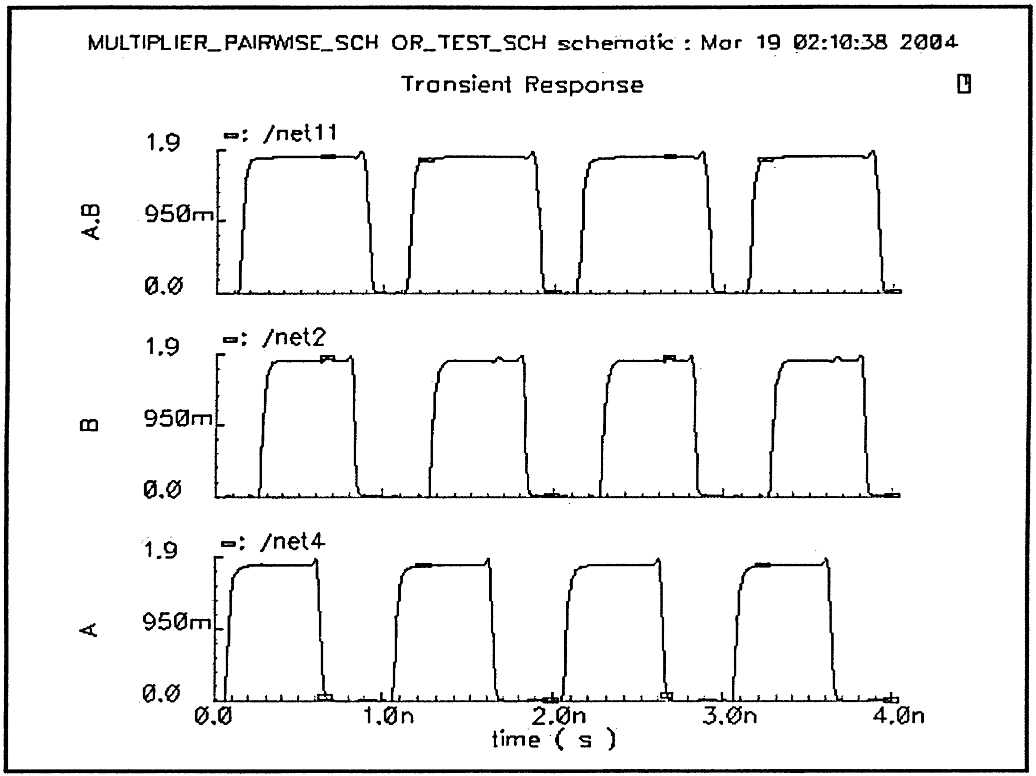


Fig. 4.8 The simulation waveforms showing respond of OR gate

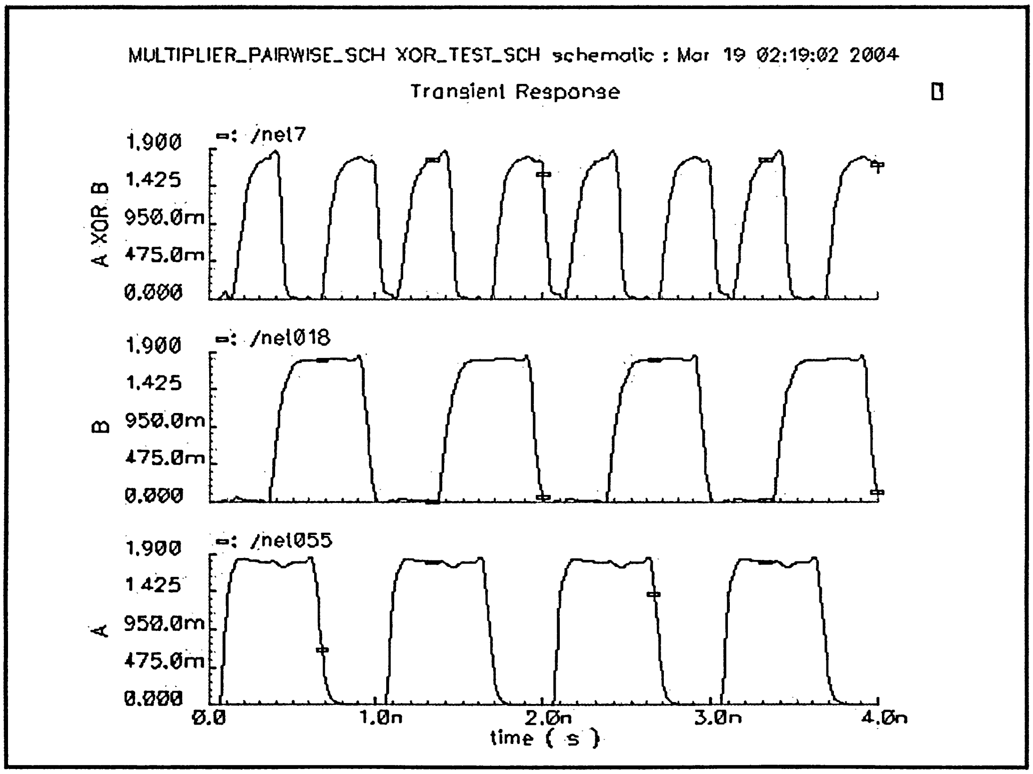


Fig. 4.9 The simulation waveforms showing respond of XOR gate

4.2 Final Simulation Results

In order to evaluate the performance of the proposed multiplier three dimensions have to be measured. These dimensions are speed, power consumption and area. In this section speed and power consumption are estimated.

Speed: Speed of the multiplier is translated to the minimum interval (frequency) between two sequential multiplication operations (8-bit x 8-bit) for which the results of multiplication are successful. To determine the frequency of multiplication, worst-case (maximum) delay of the entire design should be measured. By having the worst-case delay the minimum operating frequency of multiplier can be calculated according to Equation 4.1.

$$f_{min} = 1/\tau_{max}, \qquad (4.1)$$

where f_{min} is minimum operating frequency of multiplication, τ_{max} is the worst-case delay of the multiplier.

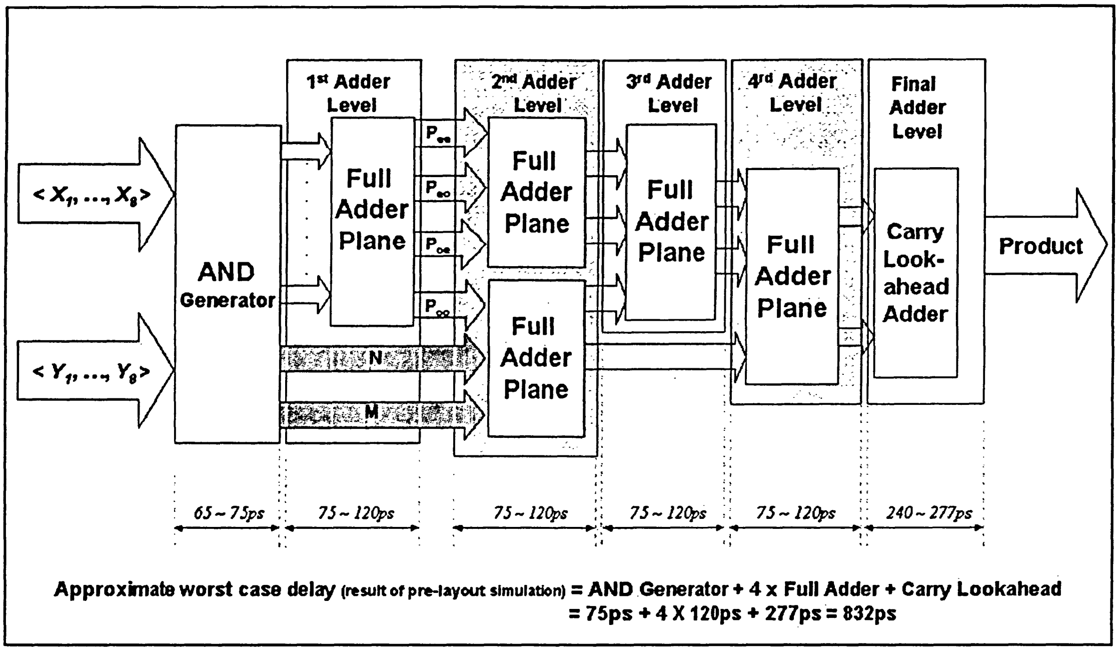
As shown in Fig 4.10 the operation of the proposed multiplier can be divided to 6 stages as:

1) AND generation 2) 1st Adder level 3) 2nd Adder level

4) 3rd Adder level 5) 4th Adder level 6) Final adder level (Carry lookahead)

Due to parallel operation (AND and addition) in stages 1 to 5 the delay of one AND gate can represent the delay of the first stage (AND generation) and so does the delay of one pseudo-NMOS adder for each of stages 2 to 5 (Adder levels). Delay of carry lookahead adder is separately measured.

In order to evaluate the worst-case delay of the entire design first the worst-case delay of each stage has been measured and, then, the final worst-case delay of the proposed multiplier can be calculated by Equation 4.2:



Approximate worst case delay (result of pre-layout simulation) = AND Generator + 4 x Full Adder + Carry Lookahead
= 75ps + 4 X 120ps + 277ps = 832ps

Fig. 4.10 The critical path of the proposed multiplier

$$\tau_{Total} = \tau_{AND\ Generator} + 4 \times \tau_{Adder\ level} + \tau_{Final\ adder\ stage}, \tag{4.2}$$

where τ_{Total} is the worst-case delay of the multiplier, $\tau_{AND\ Generator}$ is the worst-case delay of the AND generator stage which is equal to the delay of an AND gate, $\tau_{Adder\ level}$ is worst-case delay of one adder stage which is equal to the worst-case delay of one pseudo-NMOS full adder cell, and $\tau_{Final\ adder\ stage}$ is worst-case delay of the final adder stage which is equal to the worst-case delay of the 16-bit carry lookahead. Delay of AND gate can be simply measured according to propagation delay definition. Fig 4.11 shows the delay of AND gate.

The worst-case delay has a better meaning for pseudo-NMOS full adder due to possibility of different input combinations. The delay of pseudo-NMOS adder has been measured with all input combination. The worst-case delay has been occurred when $A = 1$, $B = 1$ and $C_{in} = 1$ as shown in Fig 4.12.

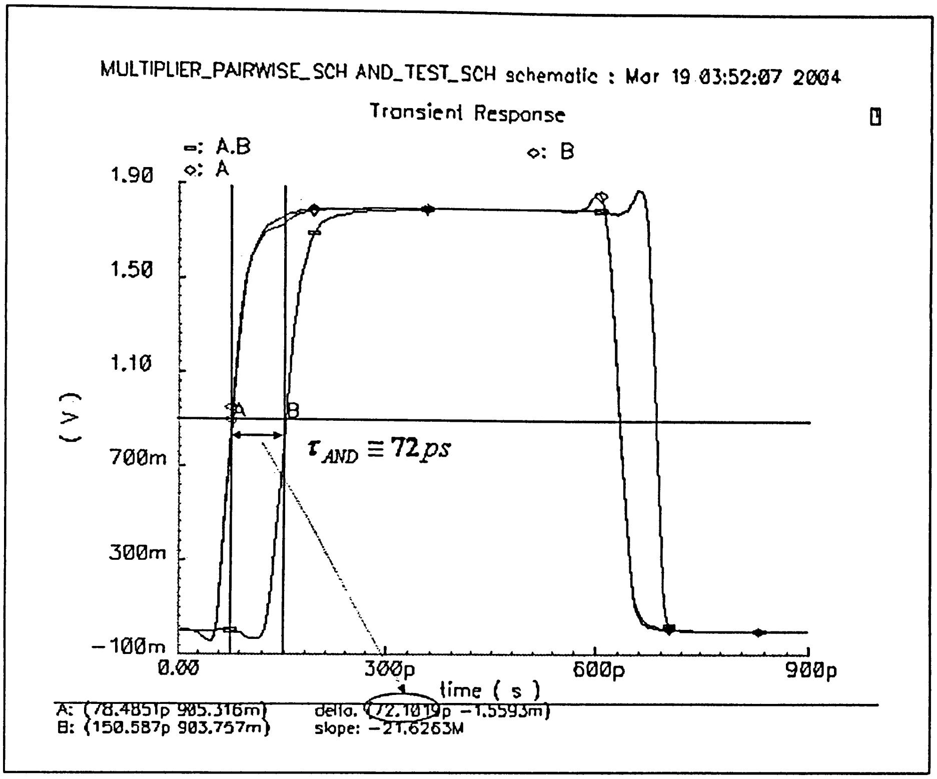


Fig. 4.11 Delay of AND gate

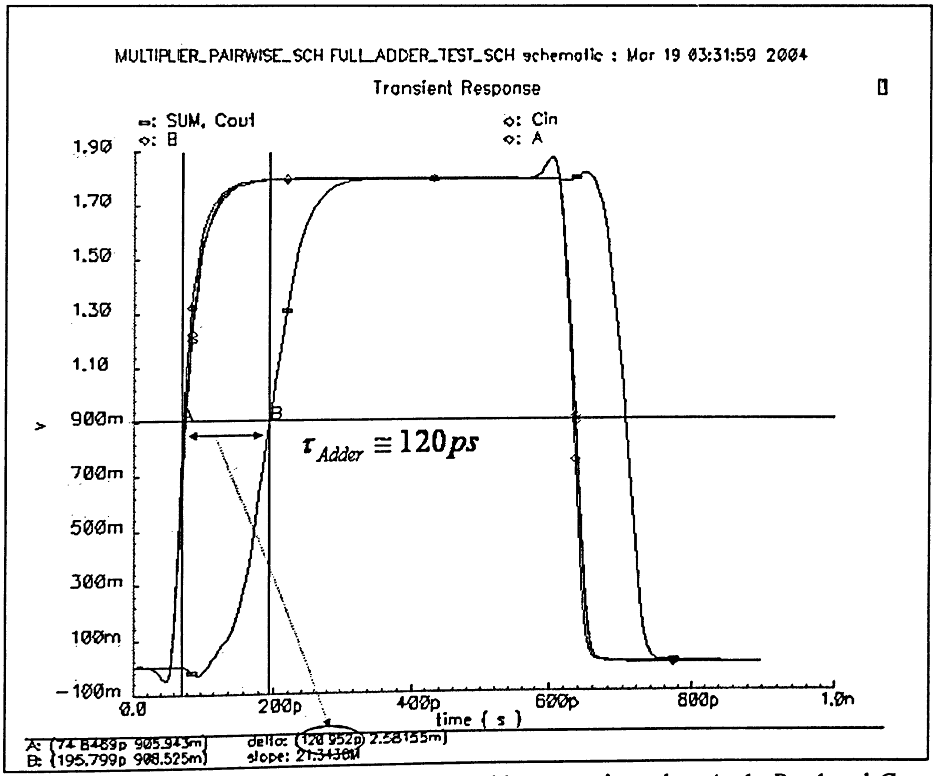


Fig. 4.12 The worst-case delay of pseudo-NMOS adder occurring when A =1, B = 1 and C_{in} = 1

In order to measure the worst-case delay of 16-bit carry lookahead adder, the same method has been taken. Different input transitions have been applied and delay has been measured between the input and the last output signals at 50% of transition point. The worst-case delay has been seen when "1111111111111111" and "1111111111111111" are added as it was expected due to rippling C_{out} signal between every 4-bit carry lookahead adder modules (remember that the 16-bit carry lookahead adder constructed by four 4-bit carry lookahead adder modules).

Fig. 4.13 shows the input and output signals in composite format. The delay occurring between input and output signals is clearly seen in this figure.

Table 4.5 shows the values of worst-case delay of AND gate, pseudo-NMOS full adder and 16-bit carry lookahead adder.

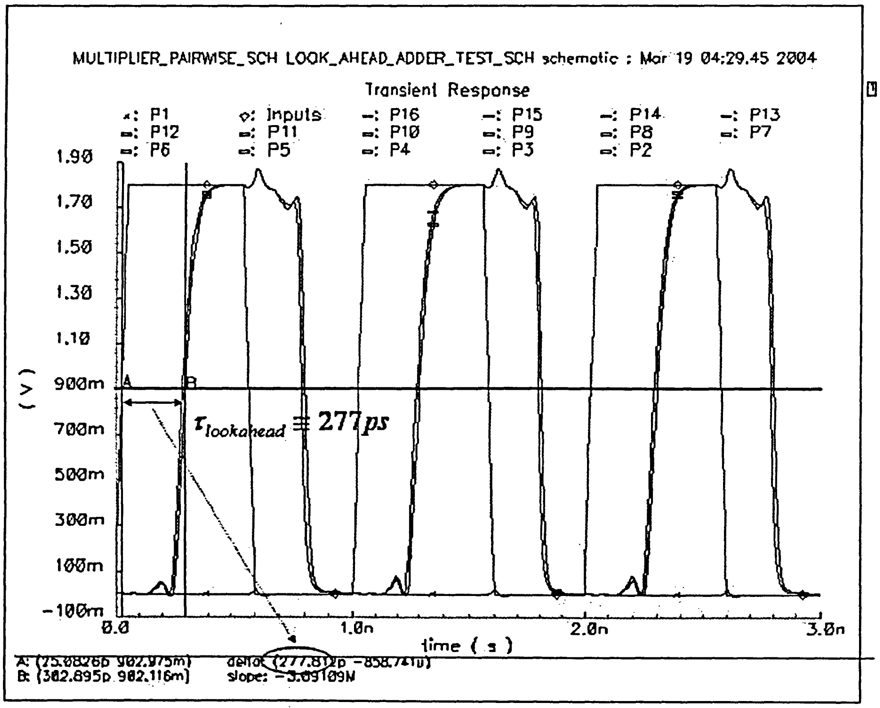


Fig. 4.13 The worst-case delay of 16-bit carry lookahead adder

Table 4.5 The results of the worst-case delay measurement

Multiplier Stage	**The worst-case delay (ps)**
AND generation (One AND gate)	75
Adder stage (One pseudo-NMOS adder)	120
16-bit carry lookahead adder	277
$\tau_{Total} = 832\text{ps}$	

It should be pointed out that he worst-case delay that has been measured and shown in Table 4.5 is the results of examining each blocks (AND, Adder plane and Final adder stage) separately. It gives an estimation of the worst-case delay of the entire design but as one may notice applying the pattern causing the worst case delay is under control only for the first two multiplier stages which are "AND Generator" and "First Adder Level". By applying pattern "11111111"as X and "11111111"as Y, AND generator creating "1" at all of its outputs. Therefore, all the input of the next stage which is the first adder level are "1". It means the worst-case delay definitely occurs in this stage. So it can be guaranteed that the first two stages operate with their slowest speed (worst-case delay) but this is not necessarily the case in the other stages. However, it is not expected that the entire multiplier's delay exceeds the total worst-case delay (τ_{Total}) that have been measured. This has been examined by applying X = "11111111" and Y = "11111111" to the proposed multiplier. Fig 4.14 shows this input signals applied to the proposed multiplier. The first curve from the top is the current drawn from the V_{dd} node. Fig 4.15 shows post-layout simulation result corresponding to the applied input (Fig.4.15).

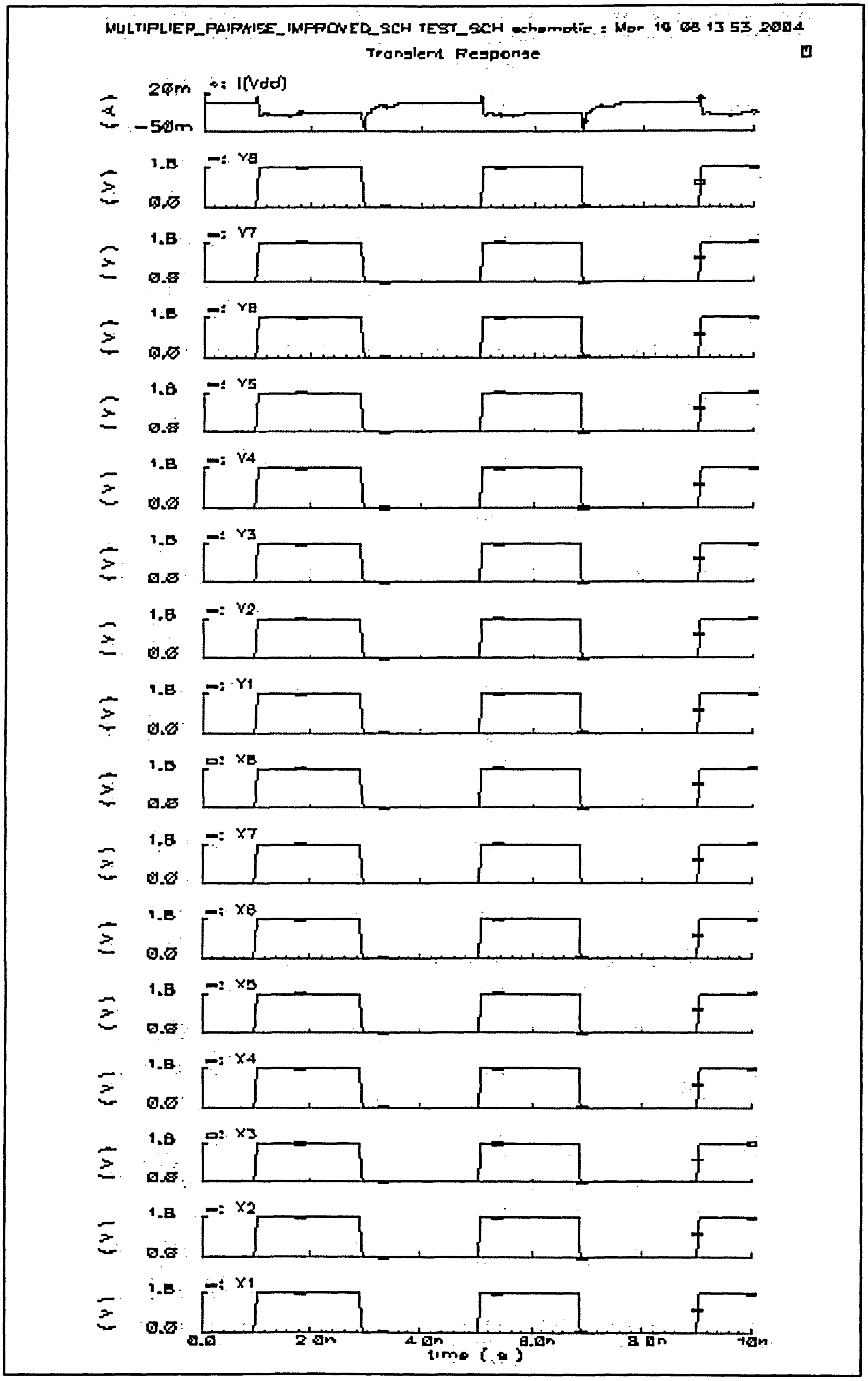


Fig. 4.14 Input waveforms of X = 11111111 and Y = 11111111

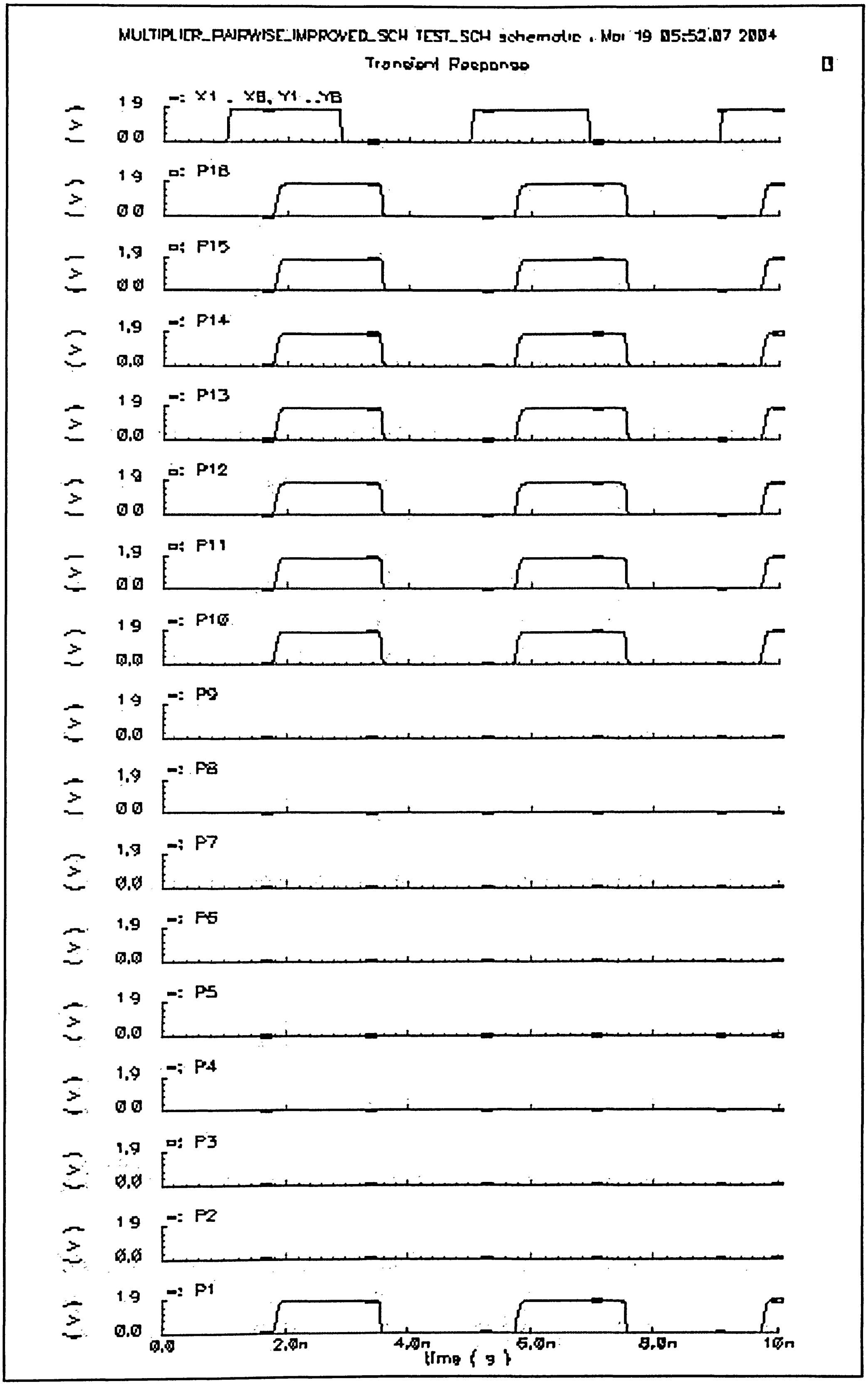


Fig. 4.15 The post layout simulation waveforms showing results of "11111111"x"11111111"

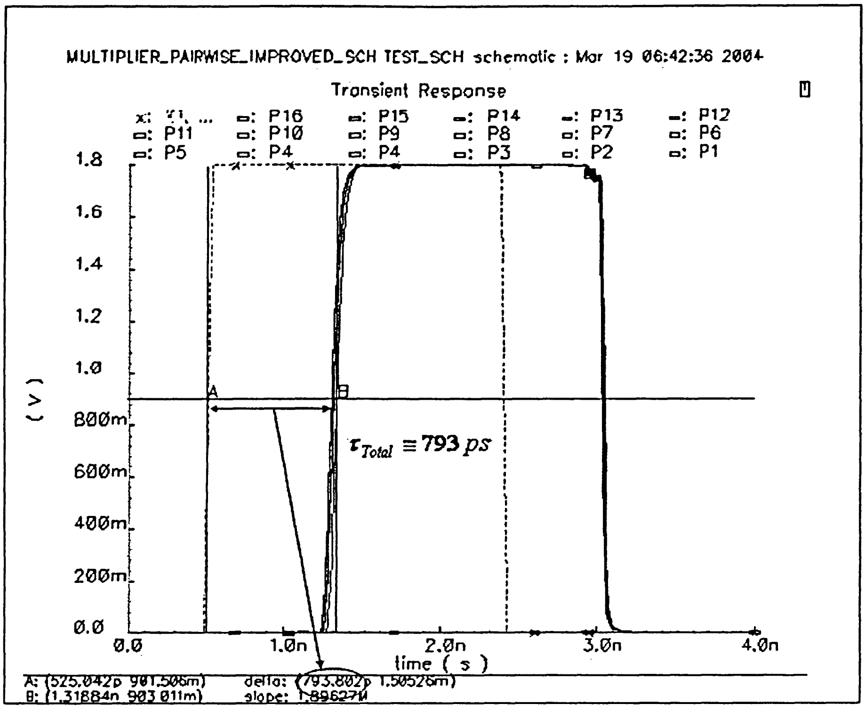


Fig. 4.16 The measured delay of the proposed multiplier corresponding to "11111111" x "11111111"

In order to measure the delay occurring during "11111111" x "11111111" the composite simulation of the input and output waveforms are used. Then, the delay has been measured in the same manner as previously defined. Fig 4.16 shows the input and output waveforms. The delay (τ_{Total}) is also shown in Fig 4.16. The measured delay is 793ps and it is less than 835ps as it was expected. So now it is fair to assume that the delay of the proposed 8 x 8-bit multiplier in worst case is less than summation of worst-case delay of its individual blocks. This worst-case delay may never occur, but it is used to set a point for maximum delay of the multiplier. It is translated to the minimum operating frequency of the proposed multiplier, which is determined as 1.19GHz. That is, the frequency of the proposed design is guaranteed at 1.19GHz.

Power: the estimation of power consumed by large digital circuit is a complex task. Measuring the power consumption is critical for low-power design as it permits the designer to optimize power, to meet requirements, and to know the power distribution through the chip. Several heuristic algorithms, statistical, and probabilistic methods have been introduced [24,25,26]. These methods become less accurate when the size of the circuits increases. It is better to decompose the large circuit into smaller modules and use this method to estimate the power consumption of each module. These methods are also very helpful approaches to optimize the performance of the decomposed modules. One of these approaches has been employed in the circuit-level design of this work in order to meet the power efficiency as one of the objective. The practical aspect of the method is explained more in detail in the topology selection and transistor sizing of full adder circuitry in Chapter 3.

Nevertheless, in case of complex systems it is wise to use CAD tools for accurate power consumption measurement.

Generally power estimation refers to the techniques of estimating the average power dissipation of circuits. There are several power analysis techniques and tools at the circuit, gate, architectural, and behavioural level of abstraction. The most straightforward method of power estimation is done through circuit simulation; i.e., performing a circuit simulation of the design and measuring the average current drawn from the supply. Therefore, the average power can be estimated which is the average of summation of the three major components as shown in Equation 4.3:

$$P_{total} = P_s + P_d + P_{sc}, \tag{4.3}$$

where P_s is static power consumption and it is the power consumed due to leakage and static currents, P_{sc} is short-circuit power consumed because of the current flowing from power supply to ground during transistors switching and P_d is referred to as dynamic power consumption which constitutes the majority of the power consumed in CMOS VLSI circuits.

The method used by CAD tools to measure the power consumption is strongly dependent on the input patterns (pattern-dependent technique). The technique is also called dynamic power simulation, which should not be confused with dynamic power. Equation (4.4) shows the dynamic portion of the total power consumption of the digital circuit. This equation is very similar to the algorithm that has been driven to compute the power consumed by digital circuits in CAD tools such as Spice [27]:

$$P_{dynamic} = \sum (C_{i_{load}} . V_{i_{swing}} . \alpha_i) . f_{clk} . V_{dd} + \sum (K_i . \alpha_i)(V_{dd} - 2V_t) . 3 f_{clk} \,, \qquad (4.4)$$

where $K_i = \beta_t / 12$, C_i is load capacitance at node i, V_i is the voltage swing at node i, α_i is known as switching activity factor at node i, f_{clk} is the system clock frequency, V_{dd} is the power supply voltage, V_t is transistor threshold voltage, β is the gain factor of the transistor. The summation is over all the nodes of the circuit, which makes the power estimation a very complex calculation. Changing any of the components in Equation 4.4 would result different power consumption values. Some of the components of Equation 4.4 are process-dependent such as V_t and V_{dd}. Other components such as C_{iload} , V_{iswing} are predetermined by to the design requirements.

Two components in Equation 4.4, which depend only on input pattern, are clock frequency (f_{clk}) and the switching factor (α_i). The input frequency of the entire system has been limited at lower level by the delay of the critical path. It means that by taking into account the approximate 800ps delay of the critical path that has already been measured, the characteristics of the input signals are determined. The required period for input signal is 800ps at minimum value. By considering 50% duty cycle as a standard for the input signals the lower input pulse width of 400ps is required. Thus, the frequency of the input signals is set at the minimum value of 1.1GHz. This brings the first condition for power consumption evaluation.

Switching factor (α_i) is the underlying factor of transistors switching. For N periods of $0 \rightarrow V_{dd}$ and $V_{dd} \rightarrow 0$ transitions, the switching activity α_i determines how many $0 \rightarrow V_{dd}^2$ transitions occur at the capacitive nodes. In the other words, the α_i represents the probability that a transition $0 \rightarrow V_{dd}$ will occur during the period $T = 1/f$, where f is the period of the input signals at the node. Considering all internal nodes' transition is a complex task, which is out of the scope of present discussion. However, it is clear at this point that choosing the pattern that makes the high number of transitions in one period of multiplier is a contributing factor to power consumption value. Hence, this brings another condition for the input patterns.

Therefore, due to using Cadence to measure power consumption of the proposed multiplier the two following conditions are considered to govern the power performance;

1) Applying the input signals with the operating frequency of approximate1.2GHz.
2) Applying the input pattern causing the maximum switching activity in entire design.

The power consumed by the entire system has been measured by changing the inputs from X_1 = 00000000 $\rightarrow X_2$ = 11111111 and Y_1 = 00000000 $\rightarrow Y_2$ = 11111111. The transition occurring by these input patterns guarantees of charging the load capacitances at all nodes of the circuits to maximum (Fig. 4.17). So one can expect to observe the maximum power consumed by the multiplier by applying this pattern. This pattern is shown in Fig. 4.14. The waveform of the current drawn by V_{dd} node during applying this pattern is shown in Fig 4.17. The average of this current computed by Cadence is 10.5 mA and the power consumption is measured as 18.09 mW. Many different patterns have been applied to the proposed multiplier and delay and power consumption have been recorded (Table 4.6 and 4.7). The maximum power consumption observed belongs to the pattern multiplication of "11111111" x "11111111", which is expected according to the switching activity definition in complex digital system.

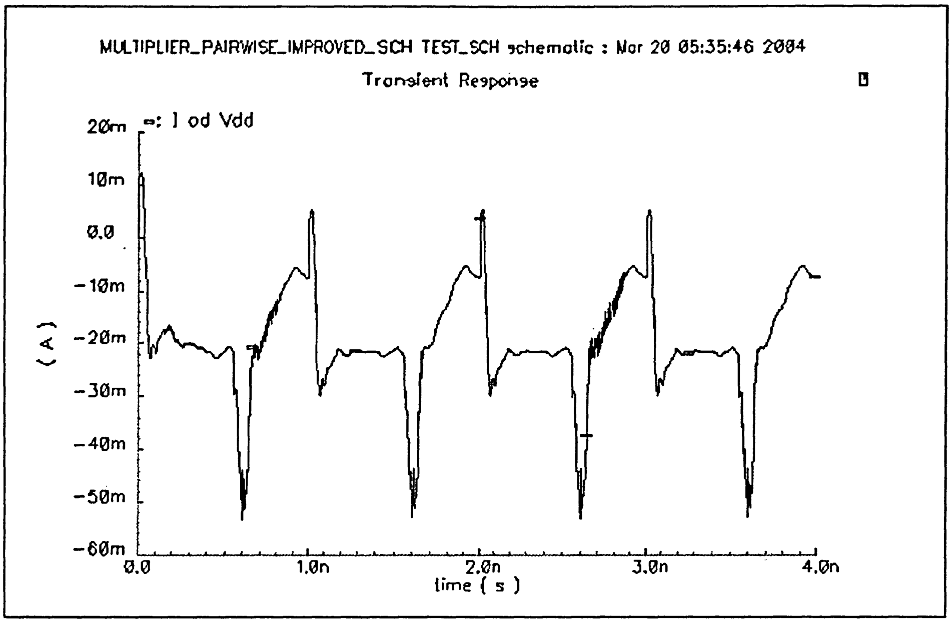


Fig. 4.17 The waveform of current drawn by V_{dd} node ("11111111" x "11111111")

In order to further examine the effect of switching activity in a complex system such as the proposed design another random pattern has been chosen. The power consumed by the entire system has been measured by applying a pattern causing transitions of $X_1 = 00000000 \rightarrow X_2 = 10101010$ and $Y_1 = 00000000 \rightarrow Y_2 = 01010101$. It is expected that the value of the power consumed by the multiplier under this pattern is almost the mean value of the power consumed by the system when all inputs are set to "0" which is called the "power down" or "minimum power consumption" value and the maximum power consumption of the system which occurs by applying "11111111" x "11111111". This is shown in Equation (4.5).

$$P_{average} = \frac{P_{\min} + P_{\max}}{2}, \tag{4.5}$$

where $P_{\min}$ is the power down value measured as 19.314 nW when no inputs applied and $P_{\max}$ is the power consumed by applying pattern "11111111" x "11111111" which is measured as 10.5 mW. The reason for such an assumption is equal random density of "0" and "1" in the pattern "101010101" and "01010101" which makes possible to assume that capacitance at 50% of all the nodes in entire system will be charged. So the assumed value for the power consumption by applying this pattern from Equation 4.5 is 9.054 mW. The actual power consumption measured by Cadence during applying this pattern is 10.347 mW. The difference about 12% has been seen between the assumed power consumption and the actual power consumption, which is measured by Cadence. This difference is mainly due to power consumed by the interconnections and routing paths.

Fig 4.18 shows the current drawn by V_{dd} node during applying pattern "10101010" x "01010101". The average of this current is computed by Cadence as 5.74 mA.

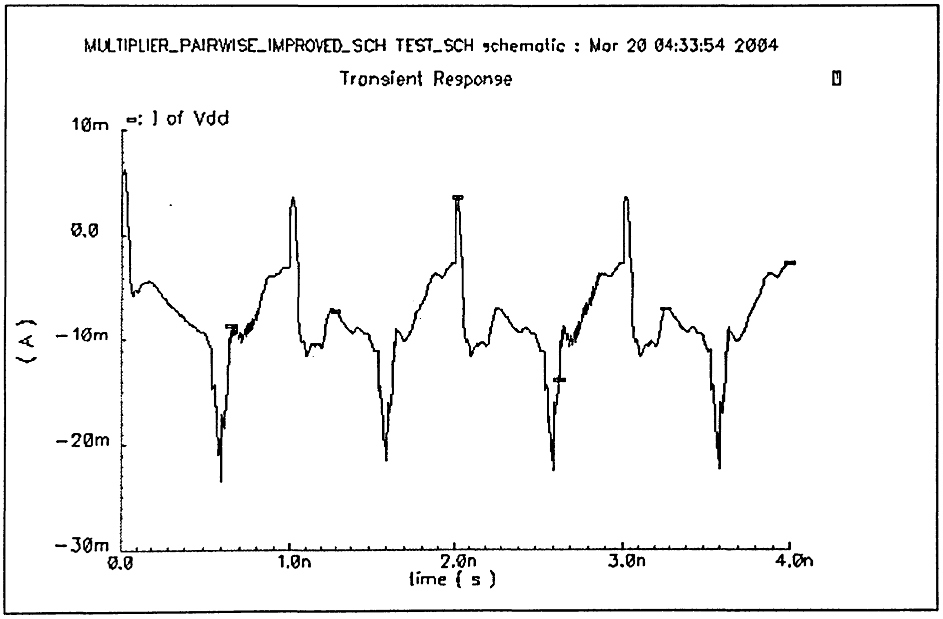


Fig. 4.18 The waveform of current drawn by V_{dd} node ("10101010" x "01010101")

Fig 4.19 shows the simulation waveform of the input patterns by assumption of 50% switching activity compared to the pattern "11111111" x "11111111". Fig 4.20 shows the multiplication product of the input patterns of Fig 4.19.

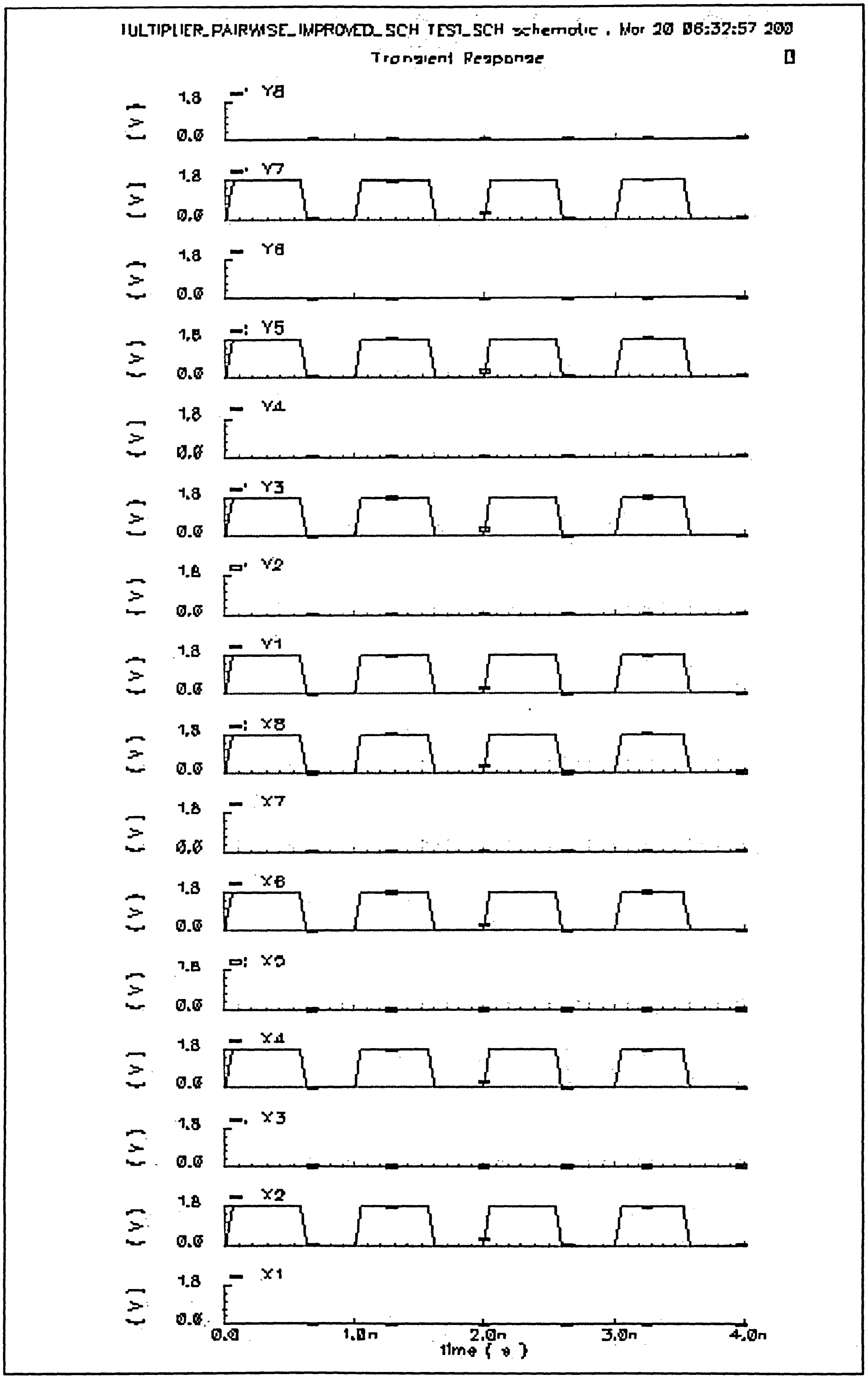


Fig. 4.19 waveform of the input patterns “10101010” x “01010101”

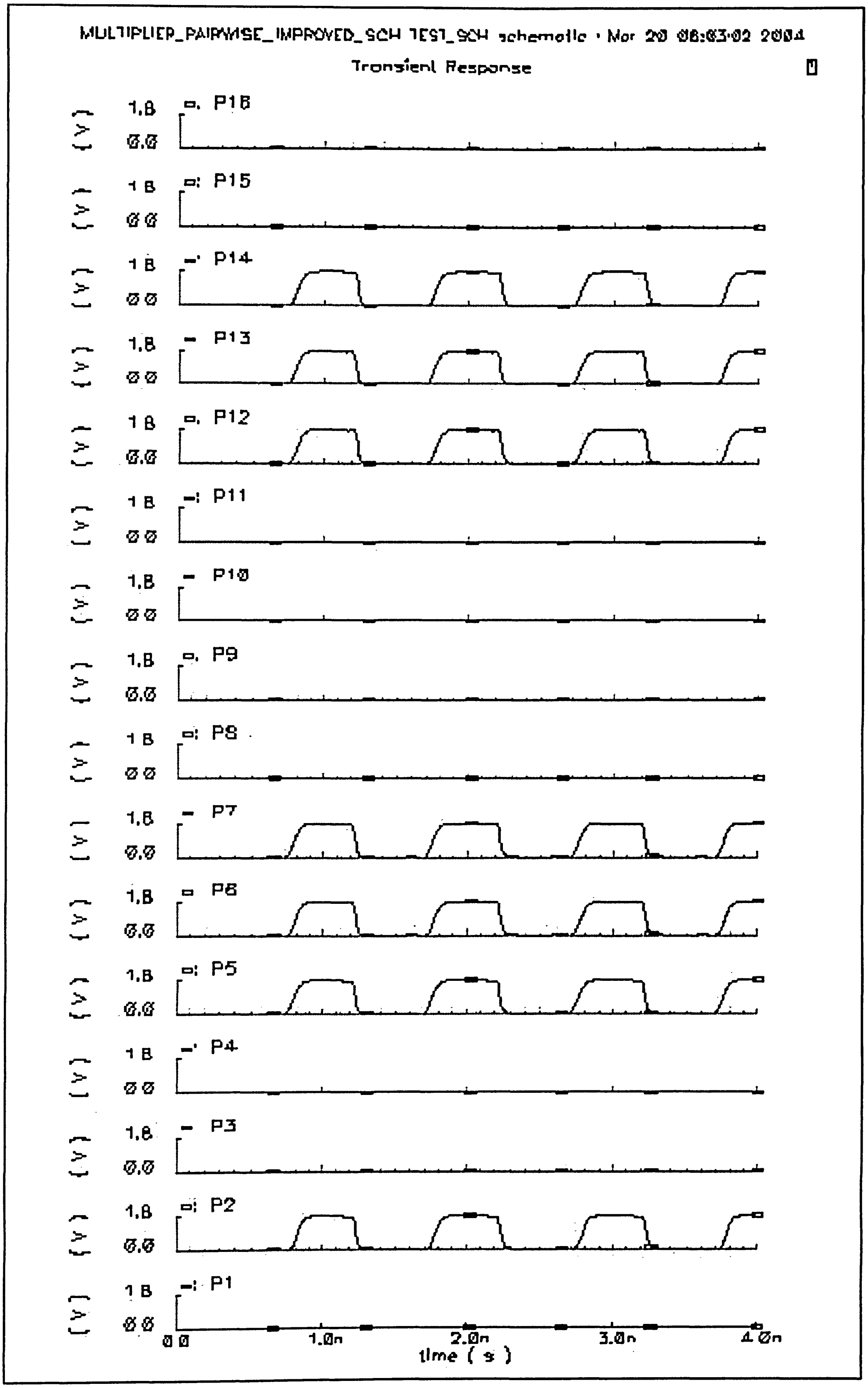


Fig 4.20 The simulation waveforms of multiplication product of the input patterns shown in Fig 4.19

A total number 100 patterns have been examined as inputs to validate the operation of the proposed design. These patterns included 80 random patterns and the 20 intentional patterns propagate the carry from 0th bit to 16th bit.

Tables 4.6 and 4.7 present the delay and power consumption of the proposed multiplier as results of the applying some of the random and intentional patterns.

Table 4.6 The numerical results of several intentional multiplications

$X_7...X_2X_1$	**Dec.**	**$Y_7...Y_2Y_1$**	**Dec.**	**$P_{17}...P_5P_4P_3P_2P_1$**	**Dec.**	**Delay (ns)**	**Power Consumption (mW)**
"00000000"	0	"00000000"	0	"0000000000000000"	0	0	0.9504
"00000001"	1	"11111111"	255	"0000000011111111"	255	0.78	7.542
"00000011"	3	"11111111"	255	"0000001011111101"	765	0.749	10.71
"00000111"	7	"11111111"	255	"0000011011111001"	1785	0.791	10.962
"00001111"	15	"11111111"	255	"0000111011110001"	3825	0.666	12.456
"00011111"	31	"11111111"	255	"0001111011100001"	7905	0.78	16.362
"00111111"	63	"11111111"	255	"0011111011000001"	16065	0.78	16.936
"01111111"	127	"11111111"	255	"0111111010000001"	32385	0.78	17.744
"11111111"	255	"11111111"	255	"1111111000000001"	65025	0.78	18.09

Table 4.7 The numerical results of several random multiplications sorted by delay

$X_7...X_2X_1$	**Dec.**	**$Y_7...Y_2Y_1$**	**Dec.**	**$P_{17}...P_5P_4P_3P_2P_1$**	**Dec.**	**Delay (ns)**	**Power Consumption (mW)**
"10110100"	180	"00101000"	40	"0001110000100000"	7200	0.662	2.54
"10011101"	157	"00101100"	44	"0001101011111100"	6908	0.662	5.17
"10100101"	165	"00001100"	12	"0000011110111100"	1980	0.662	3.91
"10101101"	173	"00110100"	52	"0010001100100100"	8996	0.670	4.42
"00111101"	61	"00101100"	44	"0000101001111100"	2684	0.689	6.93
"10111101"	189	"00111100"	60	"0010110001001100"	11340	0.703	5.92
"10111001"	185	"00111100"	60	"0010101101011100"	11100	0.703	5.99
"10110101"	181	"00111000"	56	"0010011110011000"	10136	0.704	4.50
"00001100"	12	"00011000"	24	"0000000100100000"	288	0.711	7.23
"00111101"	61	"00111100"	60	"0000111001001100"	3660	0.711	9.83
"00110000"	48	"00010000"	16	"0000001100000000"	768	0.711	9.46
"00011001"	25	"00110100"	52	"0000010100010100"	1300	0.712	7.16
"10100101"	165	"00001100"	12	"0000011110111100"	1980	0.712	3.90
"00111101"	61	"00011100"	28	"0000011010101100"	1708	0.714	4.55
"00100100"	36	"00001000"	8	"0000000100100000"	288	0.716	7.13
"10110001"	177	"00110000"	48	"0010000100110000"	8496	0.729	6.84

X_7 ... X_2X_1 = Binary representation of multiplier, Y_7 ... Y_2Y_1 = Binary representation of multiplicand, P_{17}...P_2P_1 = Binary representation of the multiplication product, Dec.= Decimal representation of the multiplicand and multiplier

In regard to design robustness, effects of noise have to be evaluated. Noise is the main factors determining the stability of the system. In the following the main sources of the noise have been described and the performance of the proposed design has been examined by considering the noise effects.

Noise: One of the main degrading factors in performance of high-speed VLSI system is noise, which comes from different sources. Noise can be induced through supply and ground of the system during switching transitions. This noise is known as Simultaneous Switching Noise (SSN). Another type of noise is thermal noise. However, this noise is not dominant source, it is inevitable [23].

Following by definition of the SSN and thermal noise the robustness of the proposed multiplier has been examined by considering the effects of these noises on the performance of the entire design.

One of the main sources of the noise in digital system is Simultaneous Switching Noise (SSN). The effects of SSN is getting more attention as a result of the continuous increase in integration level on a single chip and the operating speed. This noise is caused by the large instantaneous current, due to the switching of multiple drivers and switches, through the parasitic inductance at the ground and power node. SSN can have dramatic impacts on the system by:

a) generating glitches on the ground and power supply interconnections

b) decreasing the effective driving strength of the circuit

c) generating output signal distortion

d) reducing the overall noise margin of the system.

The quantify ground bounce or noise is given by Equation 4.6 as:

$$V_n = L_{Vss} \frac{dI}{dt}, \qquad (4.6)$$

where , V_n is ground bounce, L_{vss} is bond wire parasitic inductance and I is the current flow in the bond wire inductor.

Equation 4.6 shows that SSN can be lowered by reducing parasitic inductance. In order to reduce parasitic inductance a multiple pads and pins for power supply (V_{dd}) and ground (V_{ss}) are needed. Allocating the extra pins to V_{dd} and V_{ss} reduces L_{vss} to $L_{vss\ effective}$ due to the parallel configuration of parasitic inductors as shown in Fig 4.21.

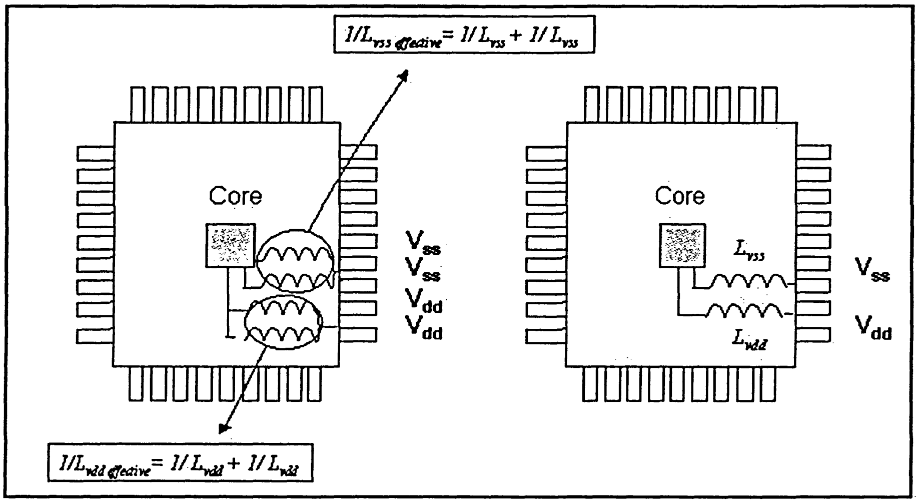

1/L_vss effective = 1/L_vss + 1/L_vss
Core
Vss
Vss
Vdd
Vdd
1/L_vdd effective = 1/L_vdd + 1/L_vdd
Core
L_vss
L_vdd
Vss
Vdd

Fig. 4.21 Adding two extra pins to V_{dd} and V_{ss} reducing the parasitic inductance

The standard package No. 68PGA offered by CMC has 36 pins which allows us to assign 32 pins to the inputs (two 8-bit multiplicand and multiplier) and the outputs (16-bit multiplication product). Therefore, the 2 extra pins are specified to V_{dd} and V_{ss} (one extra for each) and this has been done at no extra cost. These multiple pads reduce the parasitic inductance to half.

Not having glitches also strong driving capability of the output signals validate our approach to reduce the impact of SSN. This proves the robustness of the proposed design against the switching noise.

Thermal noise is another source of noise, which is generated by thermal agitation of electrons in conductors. Equation 4.7 shows the power of this noise as:

$$P = \frac{kT}{\Delta f}, \tag{4.7}$$

where P is noise power, k is Boltzmann'a constant, T is the conductor temperature and Δf is the bandwidth. It is often preferred to represent this noise in noise voltage as shown in Eqaution 4.8.

$$V_{n(Thermal)} = \sqrt{\frac{kTR}{\Delta f}}, \tag{4.8}$$

where R is the parasitic resistance. Thermal noise power, per Hz, is equal throughout the frequency spectrum, depending only on k and T. So to simply examine the effect of this noise on performance of the entire system the voltage of the final outputs could be simulated within operating temperature ranged from -40C to 135C. The voltage variation within this temperature range with capacitive load (C_l) of 5pf is 0.15 %. This shows the system is quite robust against the temperature variation. Fig 4.22 shows the simulation results of the final outputs against temperature variation.

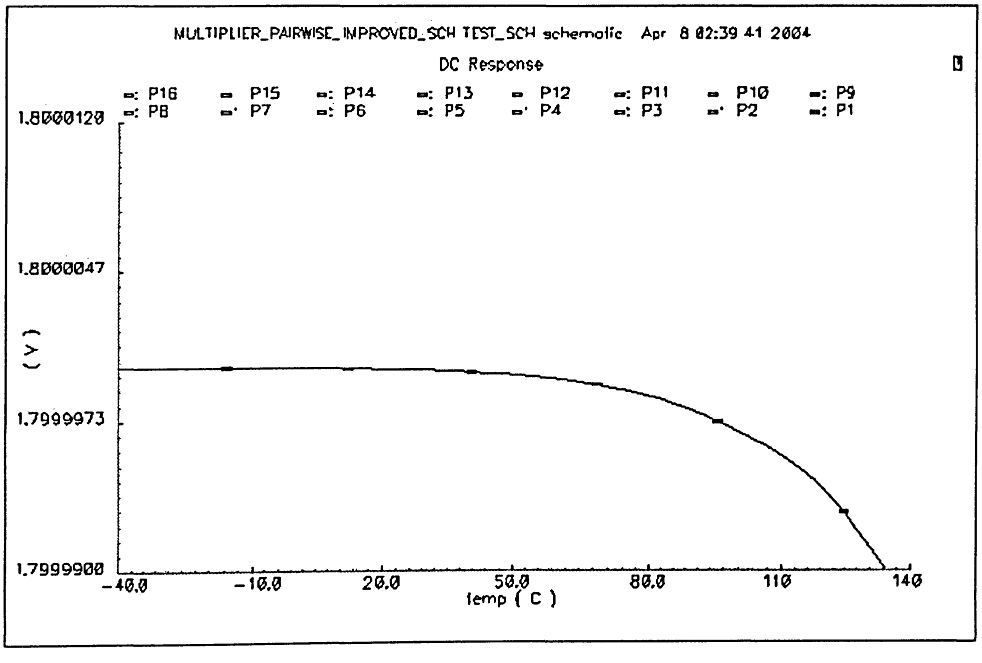


Fig. 4.22 The simulation results of the final outputs against temperature variation

4.3 Layout Considerations

The proposed 8 x 8-bit multiplier has been laid out in 0.18μm CMOS technology. In the following the layout issues such as floor planning, routing, pads and packaging of the adder cell are explained.

4.3.1 Layout Strategies

Considering transistor chaining, grouping, and signal sequencing in our proposed adder layout, has been shown to bring substantial power saving and speed improvement at no area penalty [9].

The following measures have been taken to reduce parasitic effects:

1) Minimizing the use of diffusion as a routing layer to reduce the overall parasitic by using metal II layer.

2) Placing the transistors switched by C_{in} signal close to the output.

3) Minimizing the capacitive load on C_{out} signal by minimizing the size of those transistors in Sum gate whose gate signals are connected to C_{out}.

4) Using matching transistor structure to reduce area.

5) Making the transistor connecting to C_{in} closer to the input of the circuit, therefore, reducing input capacitance.

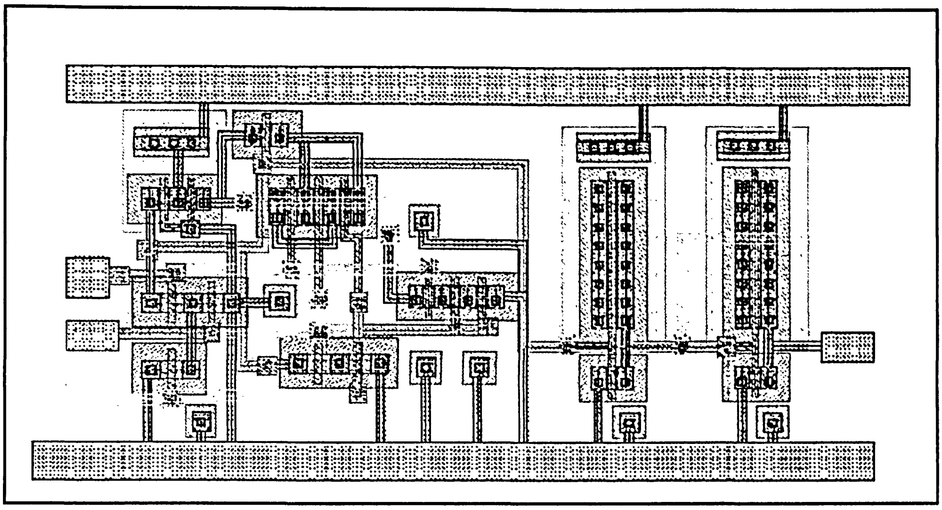


Fig. 4.23 Layout of the pseudo-NMOS full adder (die size of 22 x 8.5 μm^2)

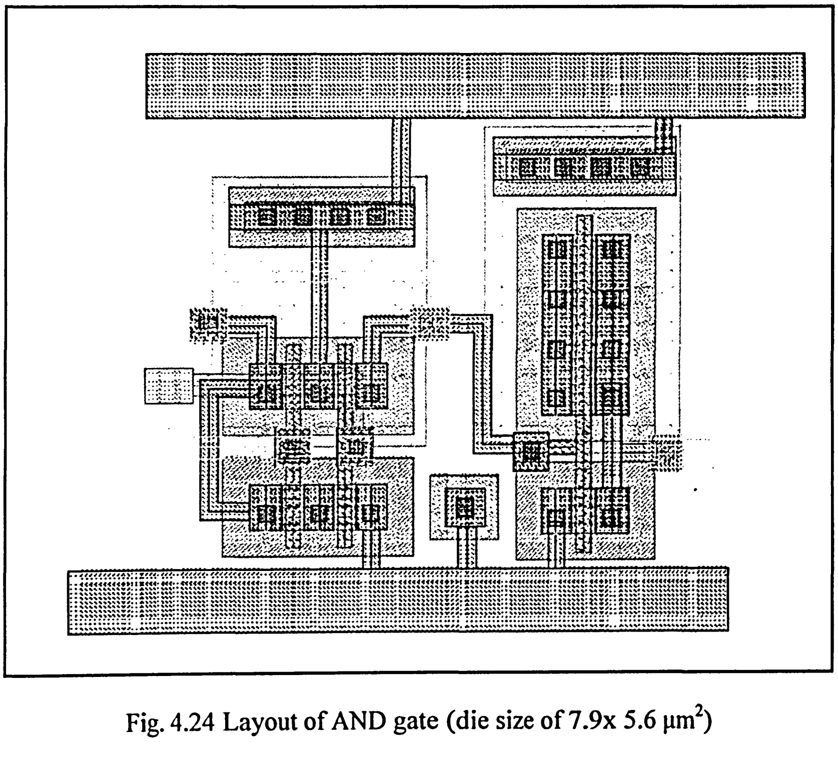


Fig. 4.24 Layout of AND gate (die size of 7.9x 5.6 μm^2)

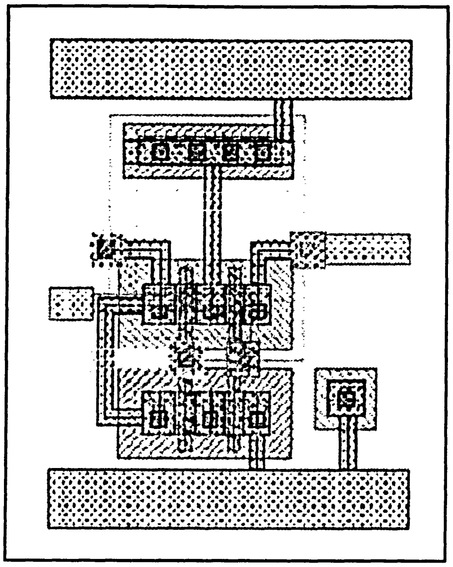


Fig. 4.25 Layout of NAND gate (die size of 5.4 x 5.6 μm^2)

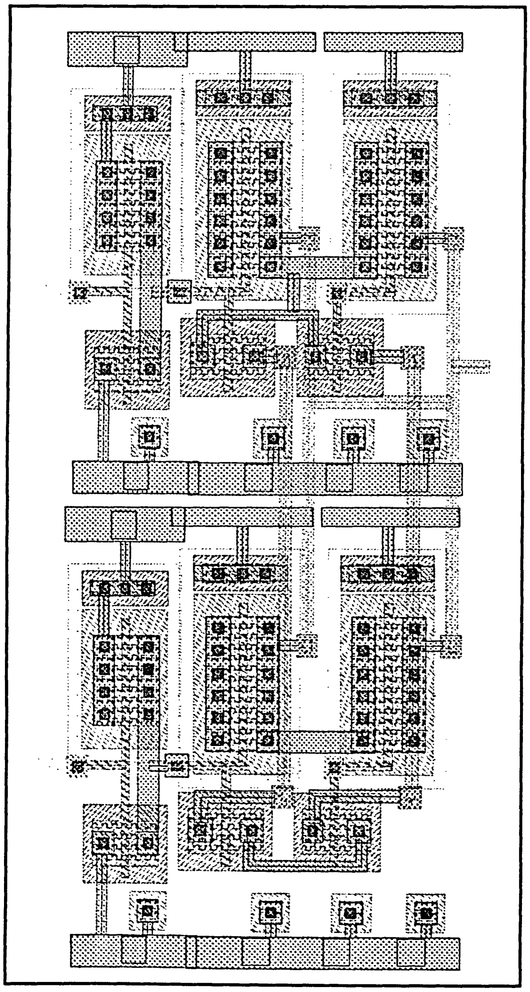


Fig. 4.26 Layout of XOR gate (die size of 10.2 x 20.7 μm^2)

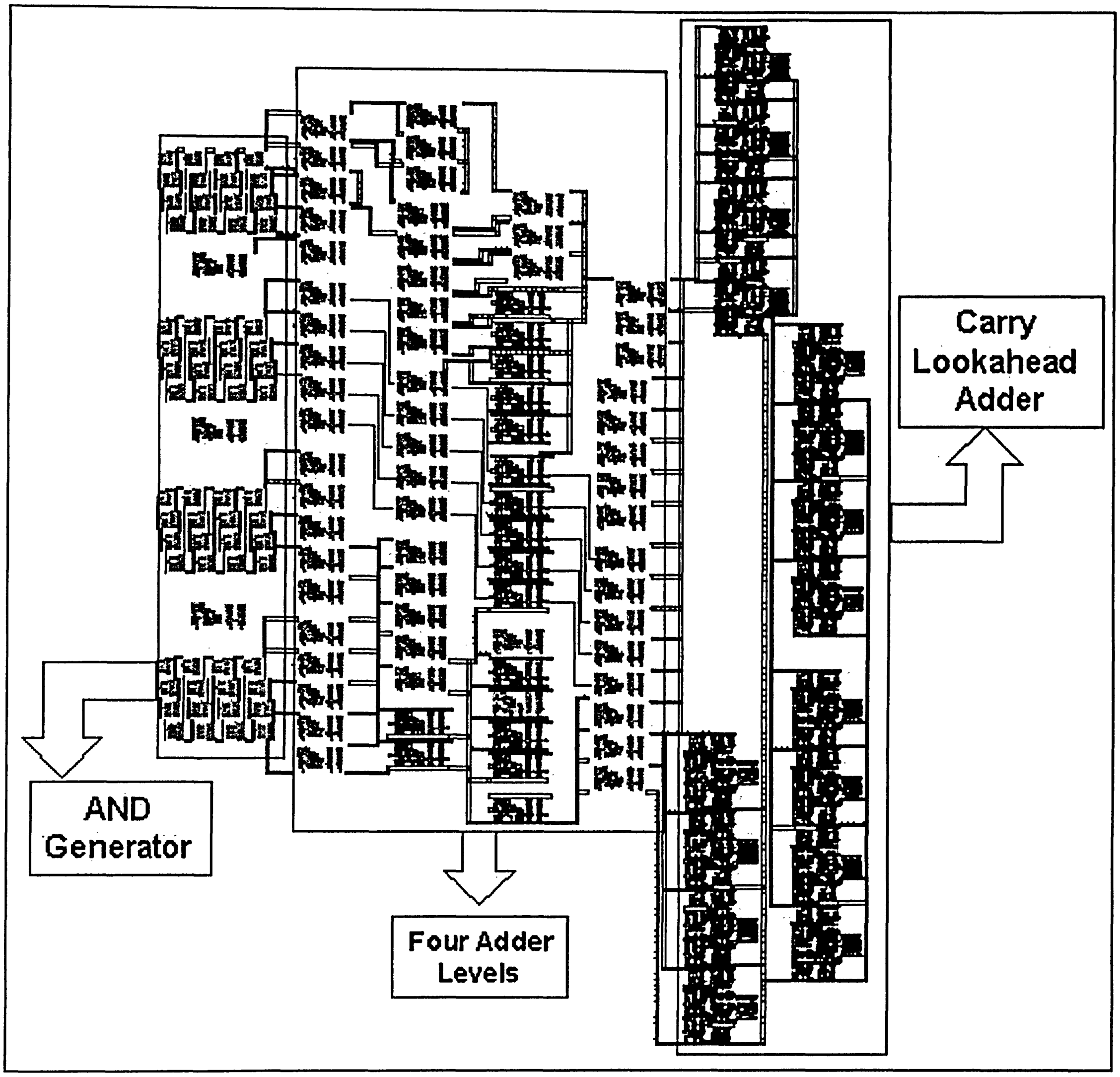


Fig. 4.27 Layout of core of the 8 x 8-bit proposed multiplier core (die size of 0.275 x 0.38 mm^2)

4.3.2 Pads, Package and Chip Size

Following are some details on the routing, pad, package and chip size of the proposed multiplier.

Routings: To reduce parasitic capacitances, local interconnections use metal # 2, metal # 3 and metal # 4. The input and output signals go through metal # 5. Avoiding long overlap between neighbouring metal layers will reduce the coupling capacitances.

Pad: I/O digital pads of TSMC library "tpz973q" from cell "PDIDGZ" have been used for connecting the core to the output. Dummy layers are also added to satisfy the density requirement.

Package: Package 68PGA is used for the chip. This package provides the core chip with 36 pins (9 pins in each side). The total area including the area occupied by pads is 1.395 x 1.37 mm^2. The core area of the chip is 0.275 x 0.38 mm^2. Fig 4.28 shows the layout the entire chip.

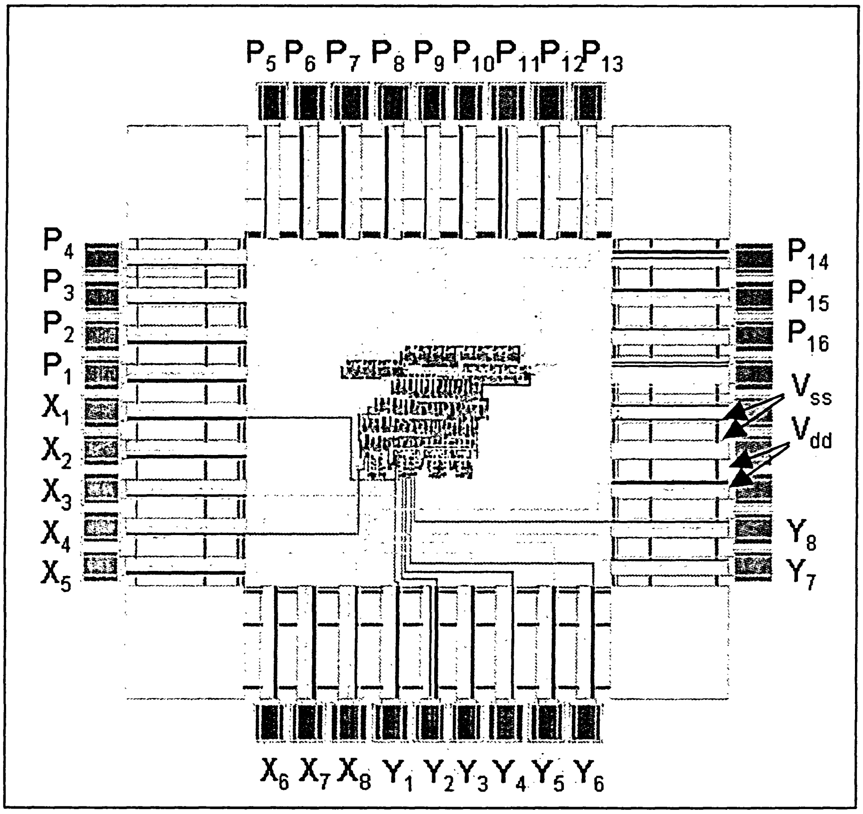

P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13}
P_4
P_3
P_2
P_1
X_1
X_2
X_3
X_4
X_5
P_{14}
P_{15}
P_{16}
V_{ss}
V_{dd}
Y_8
Y_7
X_6 X_7 X_8 Y_1 Y_2 Y_3 Y_4 Y_5 Y_6

Fig. 4.28 The proposed 8 x 8-bit multiplier chip

Chapter 5

Conclusion

Digital multipliers are one of the crucial blocks of real-time Digital Signal Processing (DSP) application ranging from digital filtering to image processing. However, speed of the operation is not the only considerations; low power dissipation and small chip area are also needed because of the dense packing of transistors in today's system on-chip (SoC) applications.

This thesis focuses on an application specific integrated circuit (ASIC) implementation of a digital multiplier with speed of operation over 1GHz. The three main considerations for the design are high multiplication speed, low power consumption, and a small rectangular chip area.

5.1 Features of the Designed Multiplier

The performance of the proposed multiplier has met the objectives of this work. The high-performance of the proposed multiplier has been achieved by an efficient design strategy as follow;

- Several multiplication algorithms have been reviewed. Considering speed as the priority in system-level, pair-wise algorithm has been chosen.
- The critical building blocks of pair-wise algorithm have been identified and ranked by their impacts on speed, power and area on the entire design. This emphasizes the full adder importance.

- An extensive study on performance of the main six static full adders has been performed in order to select the most power-speed efficient full adder topology. Six full adders haven been re-designed through an iterative approach in sake of proper transistor sizing (this approach has been used in design of the other required elements to avoid under-sizing and over-sizing transistors). This approach guaranteed power reduction in the circuit design level. The full adders have been examined under equal conditions by a realistic circuit structure.
- Speed and power are treated as same importance during topology selection by using power-delay product as a measuring factor. Area and driving capability are also taken into account. The comparison results in choosing pseudo-NMOS full adder.

Table 5.1 Performance of the proposed multiplier

Device Characteristics	
Process	Five-Metal 0.18μm Digital CMOS
Power Supply (V_{dd})	1.8V
Chip Characteristics	
Multiplier & Multiplicand	8 bits
Product	16 bits
Multiplication delay	666 ~ 793 ps
Power Consumption (power down)	19.314 nW
Power Consumption @ Input Frequency 1.1GHz	18.09 mW
Average Power Consumption	10.347 mW
Core size	0.1045 mm^2
Operating Temperature	-40C to +135C

The designed multiplier is suitable for high speed and low power applications, which provides multiplication product of two 8-bit numbers in approximate 800psec. Accurate functioning in supply ranged from 1.8 V to 0.09V has proved the suitability of the proposed architecture. The power consumption is 18.09 mW for 1.1 GHz. The design is implemented in TSMC 0.18μm CMOS technology and analyzed using Cadence's Spectre with BSIM3v3 device models.

The proposed 8 x 8-bit multiplier is laid out in 0.18μ CMOS technology and was verified for design rules and matched with schematics. The total area is 1.395 x 1.37 mm^2. The results of post-layout simulations are in reasonable consistency with those found in the design process.

5.2 Comparison Results

In this section the summarized results of investigation in the recent works on digital multipliers are provided in Table 5.2. This selection has been made based on the novelty of the works. Data are extracted form IEEE Journal of the Solid-State circuits and the results are based on measurement of the actual chips.

As it is seen in this Table, the reported multipliers are implemented in different CMOS technologies with different bit words. This can have dramatic impacts on criteria of comparison such as power, speed and silicon area.

As it is seen, different approaches in designing multipliers are taken based on different design dimensions such as area, power consumption, and speed throughput. However, it will not be fair if the results of the conducted survey on digital multiplier are directly compared, as the technology, bit width, target frequency, and simulation methodologies vary widely. The following discussion provides some indications of multiplier results and this leads us to evaluate the performance of the proposed multiplier.

When speed is the main concern Booth encoding scheme and Wallace tree reduction show their abilities for large throughput multiplier. However, combination of these methodologies with GaAs device results in high-speed multiplier, which is not feasible for CMOS device to reach. From this point of view the proposed multiplier shows its superiority for the medium bit width (4 to 8 bits) applications in speed and power trade off.

Table 5.2 Summary of the performance of the recent publications on digital multiplier

Author, Year/Ref.	Multiplier Structure	Bit-Width	Technology (μm)	Power Supply (V)	Speed (MHz)	Core Area (mm^2)	Power Consumption (mW)
N. Itoh (2001)[28]	Rectangular-Styled Wallace Tree	54	0.18	1.8	600	0.98	-
J. Butas (2001)[29]	Asynchronous Cross-Pipelined	16	0.6	1.5~5	59~251	-	40.59
S. Kim (2001)[30]	True Single-Phase Adiabatic	8	0.5	2.7	220	0.47	-
J.Lim (2000)[31]	NRERL Serial	8	0.6	2.5	0.1~1	2.37	-
J.S. Wang (2000)[32]	TSPC Flip-Flops	8	0.6	3.3	300	0.3	52.4
A. Smith (1997)[33]	GaAs	16	0.6	0.9	416	1.98	1700
J. Mori (1991)[34]	4-2 Compressor	54	0.5	3.3	100	12.4	870
K. Yano (1990)[35]	Pass-Transistor	54	0.25	2.5	227	12.7	-
M. Hatamian (1986)[36]	Parallel Pipelined	8	2.5	2.5	75	-	250

In terms of power consumption, asynchronous circuitry and adiabatic logic are viable approaches for applications where speed is not the prime concern. Nevertheless, pass-transistor logic has properties of both higher speed and lower power consumption (Table 3.12). Also NMOS reversible energy recovery logic, which is a new reversible adiabatic logic circuit, is employed in ultra-low-power applications. The serial multiplier, which has been implemented by this logic, is suitable for the applications where energy consumption is the top priority [30]. The proposed multiplier still stands ahead in power consumption compared to other designs. However, this structure could be more power efficient specifically for large bit words if pipelining technique is employed.

Where area is the prime concern, the recent progress in use of Deep Sub-micron Devices can help to overcome this constraint. It is also possible to reduce the silicon area by tighter layout style such as rectangular Wallace tree [28].

Therefore, it has been recommended that in multiplier performance and area tradeoffs, combinations of several parameters feature size, encoding scheme should be well considered. Encoding scheme has significant effect in the area of implementation. In design of the proposed multiplier the area is considered as one of the criteria in choosing the building blocks. Pseudo-NMOS shows significant area saving due to having only 14 transistors with maximum transistor width size of 2μm.

5.3 Future Work

To further improve the performance of pair-wise multiplier one needs to consider a way to reduce the critical path delay of the multiplier for longer bit width with better trade off between speed and power consumption.

A well-known technique to reduce the critical path in digital architectures is to place pipeline latches at appropriate places so that the functionality of the circuit remains unchanged and no appreciable reduction in the throughput occurs, however it takes a very accurate time scheduling for pipeline tasks.

This methodology can be considered as an alternative design of pair-wise due to the absence of any feedback loop in this architecture (Fig 2.9). The advantages of this methodology are many-fold. Since the proposed architecture permits pipelining, the operation speed can be considerably increased. This increased speed can be traded for reduction in supply voltage to achieve a considerable reduction in power consumption.

This approach makes the pair-wise architecture qualitatively a viable configuration for constant data stream in DSP applications, however, extensive quantitative evaluation based on the proper simulation arrangements is required to show the speed and power trade off.

References

[1] C. R. Baugh and B. A. Wooley, "A Two's Complement Parallel Array Multiplication Algorithm," *IEEE Trans. Computers*, vol. C-22, no. 12, pp.1045-1047, Dec 1973.

[2] J. S. Wang, "A New True-Single-Phase-Clocked Double-Edge-Triggered Flip-Flop for Low Power VLSI Designs," in *Proc. IEEE ISCAS 1997*, pp.1896-1899.

[3] A.D. Booth, "A Signed Binary Multiplication Technique," *Quarterly J. Mechanics and Applied Mathematics*, vol. 4, no. 2, pp. 236-240, 1951.

[4] P.Y. Lu, *et al.*, "A 30-MFLOP 32b CMOS Floating-Point Processor," *IEEE Solid-State Circuit Conf. Proceedings*, vol. XXXI, pp. 28-29, February 1988.

[5] W. McAllister and D. Zuras, "An NMOS 64b Floating-Point Chip Set," *IEEE Int. Solid-State Circuits Conf.*, pp. 34-35, February 1986.

[6] B. J. Benschneider, *et al.*, "A 50MHz Uniformly Pipelined 64b Floating-Point Arithmetic Processor," *IEEE Int. Solid-State Circuits Conf.*, pp.50-51, February 1989.

[7] M. Hatamian and G. L. Cash, "High Speed Signal Processing Pipelining, and VLSI," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pp.1173-1176, April 1986.

[8] A. P. Chandrakasan, S. Sheng and W. Brodersen, "Low-Power CMOS Digital Design," *IEEE J. Solid State Circuits*, vol. 27, No. 4, April 1992.

[9] A. Khatibzadeh and K. Raahemifar, "A 14-Transistor Low Power High-Speed Full Adder Cell," in *Proc. 2003 Canadian Conf. on Electrical and Computer Engineering (CCECE2003)*, Montreal, May 2003.

[10] A. Khatibzadeh and K. Raahemifar, "A Study & Comparison of Full Adder Cells Based on the Standard Static CMOS Logics," in *Proc. 2004 Canadian Conf. on Electrical and Computer Engineering (CCECE2004)*, Niagara, May 2004.

[11] J. Yuan and C. Svensson, "High-Speed CMOS Circuit Technique," *IEEE J. Solid State Circuits*, vol. 24, No. 1, February 1989.

[12] K. Hwang, "Computer Arithmetic: Principles, Architecture, and Design," John Wiley and Sons, 1979.

[13] J. J. F. Cavanagh, "Computer Science Series: Digital Computer Arithmetic," McGraw-Hill Book Co., 1984.

[14] K. Raahemifar and M. Ahmadi, "A Fast 32-bit Digital Multiplier," in *Proc. of the 8th IEEE International Conf. on Electronics, Circuits and Systems (ICECAS)*, Malta, Sept. 2001, pp.1413-1416.

[15] R. Zimmermann and W. Fichtner, "Low-power Logic Styles: CMOS versus Pass-Transistor Logic," *IEEE J. Solid State Circuits*, vol. 32, pp.1079-90, July 1997.

[16] A. Parameswar *et al.*, "A High Speed, Low Power, Swing Restored Pass-Transistor Logic Based on Multiply and Accumulate Circuit for Multimedia Applications," *IEEE CICC*, May 1994, pp. 278-281.

[17] Y. Sasaki *et al.*, "Pass-Transistor Based on Gate Array Architectures," in 1995 *Symp. VLSI Circuits, Dig. Tech. Papers*, June 1995, pp. 123-124.

[18] M. Suzuki *et al.*, "A 1.5-ns 32-b CMOS ALU in Double Pass-Transistor Logic," *IEEE J. Solid State Circuits*, vol. 28, no. 11, pp. 1145-1151, November 1993.

[19] N. Weste and K. Eshraghian, Principles of CMOS VLSI Design, A System Perspective, MA: Addison-Wesley, 1993.

[20] A. Shams and M. Bayoumi, "A Modular Approach for Designing Low Power Adders", *Proc. ASILOMAR*, June 1997.

[21] J. M. Wang *et al.*, "New Efficient Design for XOR & XNOR Functions on the Transistor Level," *IEEE J. Solid State Circuits*, vol. 29, no. 7, pp. 780-786, July 1994.

[22] E. Abu-Shama and M. Bayoumi, "A New Cell for Low-Power Adder," in *Proc. Int. Midwest Symp. Circuits Syst.*, 1995.

[23] J.M. Rabaey, "Digital Integrated Circuits," Prentice Hall, 1996.

[24] A. Bellauar and M. Elmasry, "Low-Power Digital VLSI Design Circuit and System," Kluwer academic Publishers, 1995.

[25] S. M. Kang, "Accurate Simulation of Power Dissipation in VLSI Circuits," *IEEE J. Solid State Circuits*, vol. 21, no. 5, pp. 889-891, October 1986.

[26] H. J. M. Veendrick, "Short-Circuit Dissipation of Static CMOS Circuitry anf Its Impact on the Design of Buffer Circuits," *IEEE J. Solid State Circuits*, vol. 19, no. 4, pp. 468-473, August 1984.

[27] G. J. Fisher, "An Enhanced Power Meter for SPICE2 Circuit Simulation," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 641-643, May 1988.

[28] N. Itoh, Y. Naemura, H. Makino, Y. Nakase, T. Yoshihara and Y. Horiba, "A 600-MHz 54 x 54-bit multiplier with Rectangular-Styled Wallace Tree," *IEEE J. Solid State Circuits*, vol. 36, No. 2, February 2001.

[29] J. Butas, C. Choy, J. Povanzenec and C. F. Chan, "Asynchronous Cross-Pipelined Multiplier," *IEEE J. Solid State Circuits*, vol. 36, No. 8, August 2001.

[30] S. Kim, C.H. Ziesler and M. C. Papaefthymiou, "A true Single-Phase 8-bit Adiabatic Multiplier," *Design Automation Conference, 2001*, Preceeding, pp. 758-763.

[31] J. Lim, D. Kim, S. Kang and S. Chae, "An 8 x 8-b NRERL Serial Multiplier for Ultra-low-power Application," *IEE Proceeding*, vol. 146, pp. 327-333, Dec 2000.

[32] J. S. Wang, P. H. Yang and D. Sheng, "Design of a 3-V 300-MHz Low-Power 8 x 8-b Pipelined Multiplier Using Pulse-Triggered TSPC Flip-Flops," *IEEE J. Solid State Circuits*, vol. 35, No. 4, April 2000.

[33] A. B. Smith, N. Burgess, S. Cui and M. Liebelt, "GaAs Multiplier Design for High-Speed DSP Application," *Thirty-first ASILOMAR Conference*, 1997.

[34] J. Mori and *et al.*, " A 10-ns 54 X 54-bit Parallel Structured Full Array Multiplier with 0.5-μCMOS Technology," *IEEE J. Solid State Circuits*, vol. 26, No. 4, April 1991.

[35] K.Yano, " A 3.8-ns CMOS 16 X 16-bit Multiplier Using Complementary Pass-Transistor Logic," *IEEE J. Solid State Circuits*, vol. 25, pp. 388-395, April 1990.

[36] M. Hatamian and G. Cash "A 70-MHz 8 x 8-bit Parallel Pipelined Multiplier in 2.5μm CMOS," *IEEE J. Solid State Circuits*, vol. SC-21, No. 4, August 1986.