COMPARATIVE ANALYSIS OF DIFFERENT BYPASSING-BASED MULTIPLIERS

Dissertation Submitted towards the partial fulfilment of requirement for the award of degree of

Master of Technology

In

VLSI Design

Submitted by:

Sunil Kumar

Roll No: 601261029

Under the guidance of

Ms. Sakshi

Assistant Professor

ELECTRONICS AND COMMUNICATION ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

(Established under the section 3 of UGC Act, 1956)

PATIALA – 147004 (PUNJAB)

JULY, 2014

DECLARATION

I hereby declare that the work which is being presented in the thesis entitled, **"Comparative Analysis of Different Bypassing-based Multipliers"** in partial fulfilment of the requirement for the award degree of master of technology in VLSI Design submitted in Electronics and Communication Engineering Department of Thapar University Patiala, is an authentic record of my own work carried out under the supervision of Ms. Sakshi, Assistant Professor, ECED and refers other researcher's work which are duly listed in the reference section.

The matter presented in this thesis has not been submitted in any other University/Institute for the award of degree.

Date: 10/7/2014

(SUNIL KUMAR)

Roll No: 601261029

It is to certify that the above statement made by the student is correct to the best of my knowledge and belief.

(Ms. Sakshi)

Assistant Professor

ECED, Thapar University

Countersigned by:

(Dr. Sanjay Sharma)

Head

ECED, Thapar University

Patiala-147004

Dean of Academic Affairs

Thapar University

Patiala- 147004

ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to **Ms. Sakshi, Assistant Professor**, Electronics and Communication Engineering Department, Thapar University, Patiala for her patient guidance and support throughout this report. I am truly very fortunate to have the opportunity to work with her. I found this guidance to be extremely valuable.

I am also thankful to our **HEAD OF THE DEPARTMENT, Dr. Sanjay Sharma** as well as **PG Coordinator, Dr. Kulbir Singh, Associate Professor,** of Electronics and Communication Engineering Department. I would like to thank entire faculty and staff of Electronics and Communication Engineering Department and then friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would also like to thank my parents for their years of unyielding love and encourage. They have always wanted the best for me and I admire their determination and sacrifice.

Date: Sunil Kumar

Place: Patiala (601261029)

ABSTRACT

Multipliers are the basic building blocks in Digital Signal Processing (DSP) applications, such as filtering and Fast Fourier Transform (FFT). Most of the power is consumed by these multipliers. These are essential subsystems not only limiting the throughput rate but also affecting the power dissipation and silicon area of digital circuits. The operation of multiplication speed is of great concern not only in DSP systems but also in general processors. Parallel array multipliers are widely used for achieving high execution speed. The Braun's multiplier, a parallel array multiplier is used to multiply the unsigned numbers. For higher bit multiplication, there is corresponding increase in area, delay and power of the Braun's multiplier.

In this project, various bypassing-based multipliers are implemented, which results in reduction of power, delay and area than that of conventional multipliers. These bypassing techniques also reduce the switching activity. Greater the number of zeros, higher power reduction is achieved. in this project, a comparative study of area, delay and power is done for 4x4 and 8x8 bypassing-based multipliers targeting Spartan-3E(XC3S500E-4FG320) FPGA system using Xilinx 14.5 ISE platform.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

Abbreviation	Meaning
1-D	One Dimensional
2-D	Two Dimensional
CAD	Computer Aided Design
VLSI	Very Large Scale Integration
CLA	Carry Look Ahead adder
CMOS	Complementary Metal Oxide Semiconductor
CSA	Carry-Save Adder
DCT	Discrete Cosine Transform
DSPs	Digital Signal Processes
FA	Full Adder
FPGA	Field Programmable Gate Array
IEEE	Institute of Electrical and Electronics Engineers
MFA	Modified Full Adder
MRBA	Modified Row Bypassing Adder
OBA	Optimized Bypassing Architecture
PP	Partial Product
RCA	Ripple Carry Adder
RoCoCo	Row and Column Compression
ROM	Read Only Memory
TDBA	Two Dimensional Bypassing Architecture

AC	Adder Cell
PMOS	Positive channel metal oxide semiconductor
NMOS	Negative channel metal oxide semiconductor
ITBs	Internal Tri-state Buffers
TG	Transmission Gate
DPTL	Double Pass Transistor Logic
NGD	Native Generic Database
UCF	User Constraints File
CLB	Combinational Logic Blocks
IOB	Input Output Blocks
PAR	Place and Route

CHAPTER 1

INTRODUCTION

As process technology has scaled down and chip size has increased, tens of millions of transistors can be integrated on a single chip. Since, in every transistor, the sum of the power dissipation can be significant, low power design has become one of the most important challenges in CMOS circuit design. Power consumption has emerged as an important parameter in VLSI design. For some applications, such as portable and battery operated devices, power dissipation is given importance comparable to, or even more than, speed and area consideration. Reducing the time delay and power consumption are very essential requirements for many applications. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system.

Multiplication is the basic and fundamental operation in most arithmetic computing system and Digital Signal Processing (DSP) systems. Multipliers are essential subsystems not only limiting the throughput rate but also affecting the power dissipation and silicon area of digital circuits. They lie in the critical delay path in many DSP algorithms and ultimately determine the algorithm's performance. The operation of multiplication speed is of great concern not only in DSP systems but also in general processors. Multiplication can be considered as a series of repeated additions. Design of portable battery operated multimedia devices requires energy efficient multiplication circuits.

For achieving various design parameters like low power, higher speed and smaller area, many adder and multiplier design have been proposed. Although the prior designs aimed more on reducing silicon area, the focus has changed, now more towards power and speed. With the increase in the demand of electronic portable devices, there is more attention towards the requirement of low power devices in recent years. Extending the operating hours with the battery residing in the device is the primary concern for portable electronic devices. Low power design can be obtained at logic, technology, system, architecture or at the circuit levels. Logic level optimization is critical for low power design circuits. With the exponential growth of electronic devices in recent years, there are new challenges to the community of integrated circuit design.

Adding more and more functionality with lot of real time computation, while operating at the maximum speed requires deep changes at all levels of chip design.

There are several parameters in CMOS circuits for power dissipation. The following equation illustrates the total power in the CMOS circuits:

$$P_{total} = P_{switching} + P_{shortcircuit} + P_{static} + P_{leakage} \quad (1)$$

Where $P_{switching}$ represents the switching component of the power and plays a dominating role in these calculations. $P_{shortcircuit}$ is the power dissipated due to the fact that during the transition at input level, PMOS and NMOS transistors of CMOS gate become simultaneously ON. P_{static} is the static power consumption which is from leakage current. In static power dissipated, the consumption is proportional to the no of used transistors. Charging and discharging of load capacitance determines the dynamic power dissipation. The following equation gives the average dynamic power dissipation of a CMOS gate:

$$P_{avg} = 0.5 \times C_L \times V_{dd}{}^2 \times f_p \times N \quad (2)$$

Where C_L represents the load capacitance, f_p represents the clock frequency, V_{dd} is the power supply voltage and N is the number of switching activity in a clock cycle. Thus by reducing the switching activity, power can be reduced. Thus to reduce the switching activity is to shut down the idle part which is not in operating condition. So the reduction of power can be achieved through the architectural modification via various bypassing techniques.

1.1 Organisation of Report

Chapter 2: Describes the literature review.

Chapter 3: Discusses the different types of multipliers, operation and adders. After that different types of bypassing techniques have been discussed.

Chapter 4: Describes FPGA implementation and its design flow.

Chapter 5: Discusses simulation and synthesis results of different bypassing-based multipliers and it also derives the conclusion and tells about the future scope.

CHAPTER 2

LITERATURE REVIEW

Tushar V. More and Dr. R.V.Kshirsagar [1] proposed a new design methodology of low power multiplier which combines both column bypassing techniques and booth recoding unit. In this, a method of increasing the no of zeros is discussed. This design first scans the number of zeros in the multiplicand. If zeros appear in the multiplicand, column bypassing technique is applied but if there is no zero in the multiplicand, then multiplicand will be given to the booth recoding unit which will force the multiplicand to have number of zeros and after that multiplication is performed. These zeros reduce the switching activity. It is known that input bit coefficient determines the switching activity of the component in the design. This means that if the input bit coefficient is zero, there is no need to activate corresponding row or column of adders. In addition to that, low power adder structure reduces the switching activity.

Ming–Chen Wen, Sying–Jyan Wang and Yen –Nan Lin [2] proposed a low-power multiplier design, in which whenever the output is known or the multiplicand term is zero, the corresponding column of adders can be turned-off. In previous designs, there is requirement of extra circuitary, however, this design maintains the original structure. There is no requirement of adding extra boundary cells. When compared with row-bypassing, column-bypassing technique achieves higher power reduction with lower hardware overhead. This design saves 10% of power for random inputs. However, if there are more 0's than 1's then higher power reduction can be achieved.

Sunjoo Hong, Hoi-Jun Yoo and Taehwan Roh [3] proposed a low-power parallel multiplier based on optimized bypassing-based architecture (OBA). There are two types of adder cells to reduce the power consumption by 15.7%. First one is the 2-dimensional bypassing adder (TDBA) which performs both row and column bypassing scheme simultaneously, and the second one is modified row-bypassing adder (MRBA) for the proposed row-bypassing scheme. As the TDBA and MRBA don't use FAs and have internal tri-state buffers (ITBs) to partially activate the logic evaluation. Therefore, there is 33.7% reduction in power consumption in TDBA and 32.0% in MRBA , compared to the FA with additional logics. Based on above adders, when the

input frequency is 50 MHz and the probability of 0 and 1 of inputs is equal, a power consumption of 145µW is achieved in the 8x8 multiplier.

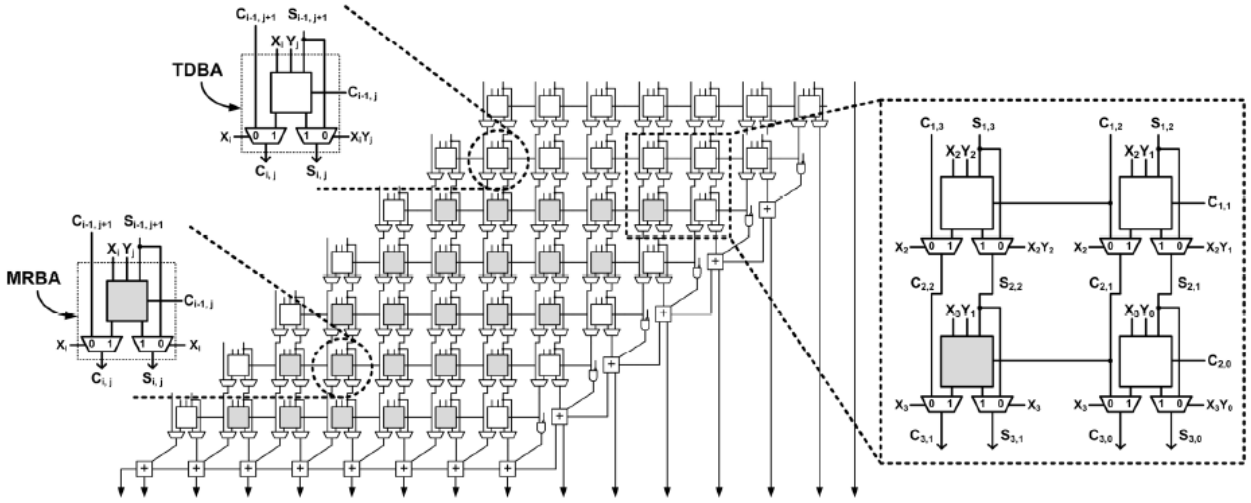


Figure 2.1 An OBA based Multiplier.

Syed Ershad ahmed, Sibi Abraham, Sreehari Veeramanchaneni, Moorthy Muthukrishnan N, M.B. Srinivasan [4] presented a twin precision multiplier with modified 2-dimensional bypass logic. This combines the properties of both twin-precision multiplier and two-dimensional bypassing-based multiplier to design a multiplier which will further decrease the power dissipation in reconfigurable multipliers. The multiplier can perform either one 8-bit multiplication or two 4- bit multiplications. The multiplier structure is modified by the addition of a 2-dimensional modified addition logic which results in reduction of dynamic power and the delay also. A reconfigurable bit will control the mode of operation of the multiplier. This design reorders the partial products of a multiplier in such a way to get the advantage of low power consumption. Reconfigurable bypassing blocks, i.e. RMRBA and RTDBA are presented to reduce the delay of the multiplier. Simulation results show that there is slight increase in the area achieving an improvement 25.5% in delay and up to 29% reduction of power-delay product when compared with existing designs.

Alvin Joseph J. Tang and Joy Alinda Reyes [5] presented a comparative analysis of different multiplier architectures. The four multipliers include the array multiplier, a bypassing-based multiplier with tree structure, a multiplier with 2-dimensional bypassing technique and a bypassing multiplier using improved column bypassing schemes. These architectures are

compared in terms of critical path delay, power dissipation and area in terms of transistors count. Due to the scaling effects on leakage current, the multipliers perform worst in terms of power compared to the array multiplier. All the multipliers are implemented in 90nm CMOS technology. Each of the three multipliers has its own trade-offs between power and delay.

Jin-Tai Yan and Zhi-Wei Chen [6] proposed a low-cost low-power bypassing-based multiplier, based on the simplification of the addition operations in a low-power bypassing based multiplier. The results for row-bypassing based multiplier, column bypassing-based multiplier and 2-dimensional bypassing-based multiplier are compared. These results shows that this low-cost low-power multiplier saves hardware cost by 15.1% and reduces the power dissipation by 29.9% on the average for 4x4, 8x8 and 16x16 multipliers.

In this multiplier, there is bypassing of addition operation in $(i+1, j)^{th}$ full adder if the product, a_ib_j, matches to the carry bit, $c_{i,j-1}$, that is , there is addition operation in the $(i+1, j)^{th}$ full adder if the product, a_ib_j, does not matches with the carry bit, $c_{i,j-1}$. The control signal is obtained by XORing the result of the product, a_ib_j, and the carry bit, $c_{i,j-1}$. Here, only (A+1) HAs are used instead of full adders and (A+B+1) HAs. Also, each (A+1) HA in the CSA array is attached by only one tri-state buffer and two 2-to-1 multiplexers. So, this low-cost low-power multiplier achieves higher power reduction with less hardware cost than the earlier multipliers.

Ko-Chi Kuo and Chi-Wen Chou [7] presented a low power multiplier using a new design technique. The bypassing technique is utilized in this multiplier to minimize the switching activities and critical path is decreased by tree structure. In this design, more reduction in power dissipation is obtained by disabling the adder when some partial products of multiplier are zeros and a decrease in delay time is observed with tree structure. In this design, greater power efficiency with less extra hardware and efficient power-delay product can be achieved in comparison to that of bypassing-based multipliers.

This design combines two methods i.e. bypassing techniques and tree structure. Instead of using carry-save adders, used in bypassing multipliers, ripple-carry array is used in this design where rows of adders also have bypassing ability. The modified full adder used here only needs two tri-state buffers and one multiplexer. An AND gate is added to the last carry output in each row of full adders for correcting output when b_j is 0. However, this design also combines the tree

structure to reduce the critical path. This design needs $(\frac{5}{2}n - 3)$ full delay in the worst case for NxN multiplier design. The modified ripple carry full adder cell is illustrated in fig.

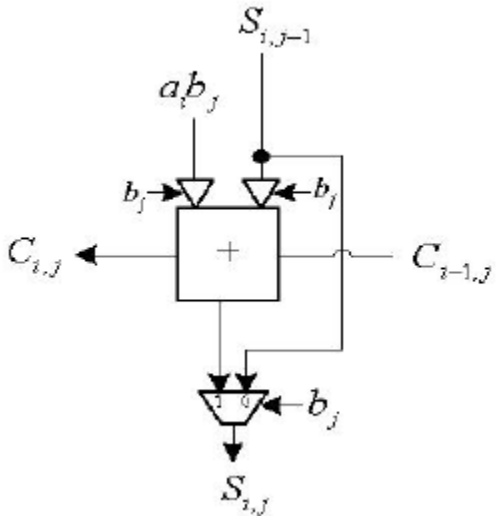


Figure 2.2 The modified ripple carry full adder cell.

Shweta S. Khobragade and Swapnali P. Karmore [8] presented a low-power multiplier which uses modified booth algorithm and column bypassing technique. Booth coding technology is used to rearrange the input bits. The function of the booth decoder is to rearrange the given booth equivalent. To increase the no of zeros, this booth technique is used. Switching activity is determined by the input bit coefficient. Whenever the input bit coefficient is zero, corresponding column or rows of the multiplier need not to be activated. There is higher power reduction if the multiplicand is having greater no of zeros. Booth encoding reduces the number of partial products in the multiplication result, thus, higher speed multiplication is achieved. Then we combine booth algorithm with column bypassing scheme. The advantage of this scheme is that it eliminates the extra correcting circuitary.

Yin-Tsung Hwang, Jin-Fa Lin, Ming-Hwa Sheu and Chia-Jen Sheu [9] proposed two novel low power multipliers based on enhanced row bypassing techniques. By eliminating the unnecessary computation through signal processing, power can be saved. In an array multiplier, computations which are futile, occur on those rows and columns of adders corresponding to 0 bits in the input bit coefficient. This design resolves the adverse DC power consumption problem due to voltage loss in gated signals. Two versions of the design in which one focuses on maximizing power saving while the other emphasizing on reducing circuit complexity are

proposed. Area overhead for both designs is confined to 23.4% and 12.8%. Also the power saving can be achieved up to 17%.

For the row-bypassing based multiplier, there is disabling of addition operation in the j^{th} row to reduce the power dissipation if the multiplier bit b_j is zero i.e. all partial products, $a_i b_j$, are zero. Thus, there is bypassing of addition operations in the j^{th} row of CSAs and the outputs from the $(j-1)^{th}$ row of CSAs without affecting the multiplication result.

R. Anitha and V. Bagyaveereswaran [10] presented the hardware implementation of the multiplier devices in FPGA devices using Verilog HDL. Bypassing techniques are used to decrease the computational delay and improve resource utilization and Braun-architectured multiplier is compared with its architectural modification i.e. row-bypassing architectures and column- bypassing architectures and the full adder structure has been moved by the fast adders. The design has been implemented on Spartan 3E, Vertex 5 and Vertex 6. The proposed design shows the less utilization when it is compared with all other multipliers. The average dealy and combinational delays are less in the Virtex – 6 Low power FPGA device. So when compared with other FPGA devices, the best result are obtained in Virtex-6 low power. Also it is feasible for the Image processing, DSP Processor and multimedia technology.

T. Francis, T. Joseph and J. K. Antony [11] presented a multiplier which aims at an optimization of power and delay of Braun's multiplier by using bypassing techniques and modification of adders. The dynamic power can be reduce by using bypassing techniques and delay can be reduced by replacing ripple carry adder in place of full adders in the last stage by optimizing adders in different logics, transmission gate (TG) and Double pass transistor logic (DPL). Adders with different logic styles are designed and implemented and their functional parameters are compared. The results show that column bypassing–based Braun multiplier is best. This new multiplier is used for implementation of a MAC unit which is more efficient.

CHAPTER 3

MULTIPLICATION

Multiplication is a mathematical operation that performs addition of integer a specified number of times. This is the basic arithmetic operation required in processors and digital signal processing systems. To perform a multiplication of M-bit by N-bit as shown in Figure 3.1, the M-bit multiplicand $A = a_{M-1}a_{M-2} \dots \dots \dots a_2a_1a_0$ is multiplied by the multiplier of N-bit $B = b_{N-1}b_{N-2} \dots \dots \dots b_1b_0$ to produce the M+N-bit product $P = p_{M+N-1}p_{M+N-2} \dots \dots \dots p_1p_0$.

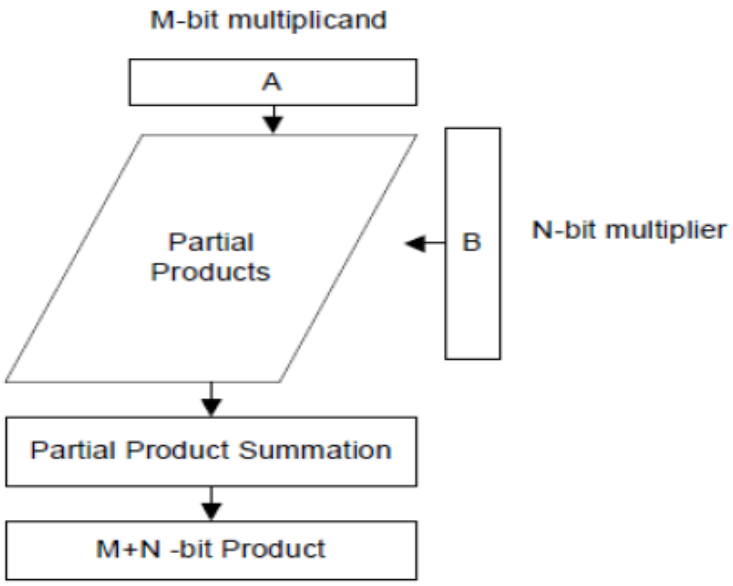


Figure 3.1 Block Diagram for Multiplier.

3.1 Types of Multipliers

Multipliers are classified by the format in which words are accessed namely:

1. Serial multiplier
2. Parallel multiplier
3. Serial-parallel multiplier

Serial Multiplier

It uses a successive addition algorithm .It has a simple structure because both the operands are entered in a serial form .Therefore the physical circuit requires minimum amount of area and less hardware.

Parallel Multiplier

Most of the digital systems incorporate a parallel multiplication unit to carry-out high speed operation due to the operands is entered in parallel which consumes less time.

Serial-Parallel Multiplier

The serial parallel multiplier serves as a good trade-off between time consuming serial multiplier and area consuming parallel multiplier .These multipliers are used when there is a demand for both high speed and small area. In a device using serial-parallel multiplier, one operand is entered serially and the other is stored in parallel with a fixed number of bits.

3.2 Operation

Conventional array multiplier is primarily used for computing multiplication of two input data. Consider the multiplication of two unsigned n-bit numbers, where A = $a_{n-1}\ a_{n-2}\ \ldots a_0$ is the multiplicand and B = $b_{n-1}\ b_{n-2}\ \ldots b_0$ is the multiplier. The product P = $p_{2n-1}, p_{2n-2}\ \ldots p_0$ can be written as follows:

$$P = AB = \left(\sum_{i=0}^{n-1} a_i . 2^i\right)\left(\sum_{j=0}^{n-1} a_j . 2^j\right) = \sum_{j=0}^{n-1}\sum_{i=0}^{n-1} (a_i b_i)\, 2^{i+j} \qquad (3)$$

where i and j are the number of bits in the multiplier and multiplicand, respectively. The terms generated are called partial products. These partial products are then added to get the final multiplier output as shown in figure 3.2.

$$\begin{array}{cccccccc}
 & & & a_3 & a_2 & a_1 & a_0 \\
 & & \times & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 \\
 & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & \\
 & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & \\
a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & \\
\hline
P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0
\end{array}$$

Figure 3.2 Multiply operation [1].

3.2.1 Steps of Multiplication

Steps for multiplication calculation comprise of two stages:

1. Partial product generation
2. Accumulation

In partial product generation, there are several techniques for generating partial products and these fall into two differing groups mainly concerned with either the complexity of the generating circuit or the number of partial product generated. The simplest technique is called the shift/add multiplication, in which partial product are generated in a traditional way by means of sequential operations or bit-at-a-time ones. It generates all the partial products after only one AND-gate delay. Another technique, proposed by Booth, aims to reduce the number of partial products generated by dividing the multiplier into overlapped three-bit sequences, which indicate the partial product to be generated. One of the techniques to design low power multiplier is through partial product reduction. The classical method for partial product reduction consists of using combinational logic-based structures to reduce the number of products until only two products are left. The addition of these two remaining products, at a later stage, produces the final result of the operation. These reduction schemes are based on CSA and require counters and compressor components organized into different topological configurations.

In accumulation, addition of all partial products is done sequentially. A carry propagate adder (CLA- carry look-ahead adder) is usually used to add the two remaining operands and obtain the final result of the multiplication.

3.3 Adders

Adders are the most commonly used in various electronics applications. According to the way of carry propagation, adders can be classified into two structures:

1. Ripple-carry Adder (RCA)
2. Carry-save Adder (CSA)

3.3.1 Ripple Carry Adders (RCAs)

The Ripple Carry Adder Architecture is well known architecture which is composed of cascaded full adders for n-bit adder. A block diagram is shown in figure 3.3 for RCA. It is formed by cascading the blocks of full adders in series. The carry out of one stage is fed directly to the carry-in of the next-stage. For an n-bit parallel adder it requires n full adders.

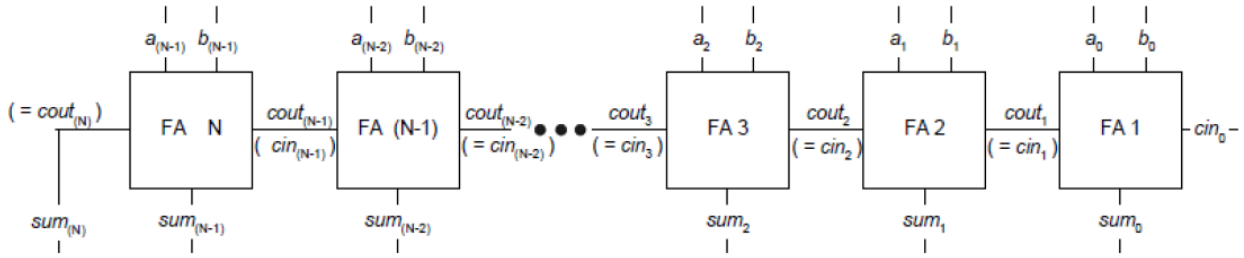


Figure 3.3 Schematic for N-bit Ripple Carry Adder [11].

• Not very efficient when large number bit numbers are used.

• Delay increases linearly with bit length.

In RCA multiplier all adder cells are composed of RCA adders. e.g. it needs 3N adders to accomplish multiplication in an NxN multiplier. However, delay time needed in the worst case is (2N+1) full adder delay. The use of an RCA adder in a multiplier is shown in figure 3.4.

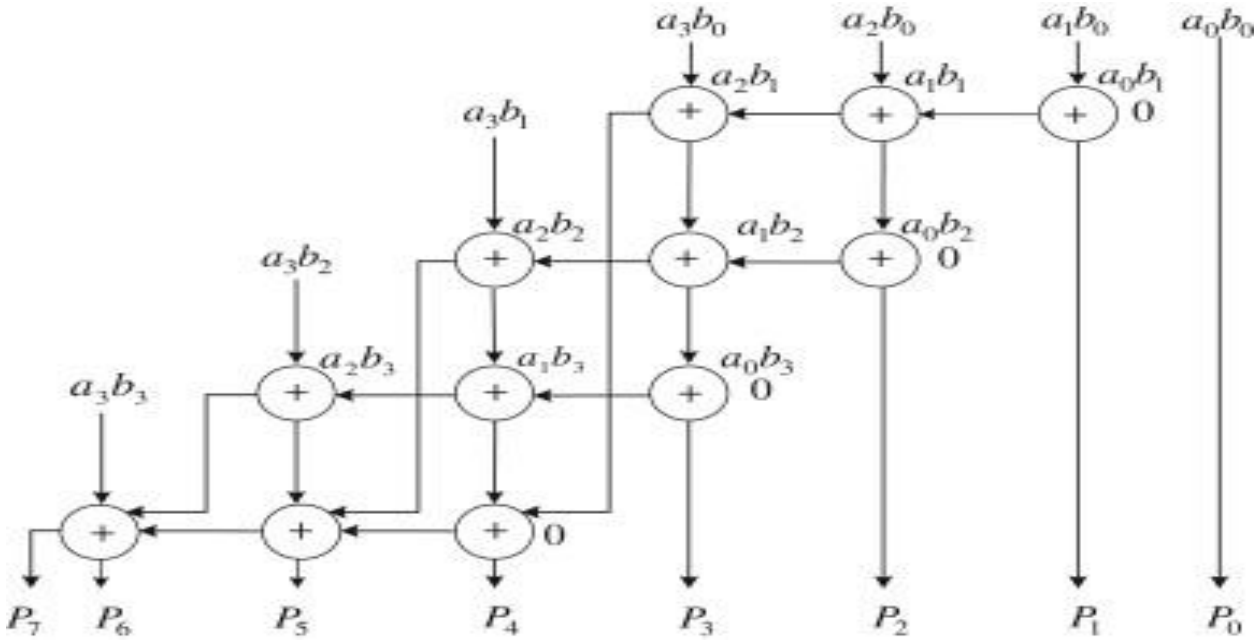


Figure: 3.4 Accumulation of PPs using Ripple Carry Adder [11].

3.3.2 Carry Save Adders (CSAs)

Carry-save adders (CSAs) are efficient operators when three or more operands are to be added. Their efficiency is due to the fact that the addition is performed without propagating carries. Carry-save addition was introduced in the context of sequential multiplication. In the simplest implementation of a CSA, the basic elements of the CSA are full-adders (FAs) and half-adders (HAs).

The half-adder in Figure 3.5(a) adds two binary inputs, *A* and *B*. The result is 0, 1 or 2, so two bits are required to represent the value. They are called the sum and carry-out denoted by *S* and C_{out}. The C_{out} output has double the weight of the other bits. The full-adder in Figure 3.5(b) has a third input called C_{in}. The truth-table and behavioural representation of the HA and FA are given in Tables 3.1 and 3.2. The notations, $\oplus$,$\cdot$ and + in Tables 3.1 and 3.2 represent logical XOR, AND and OR operators, respectively. Typical gate-level implementations of HA and FA are depicted in Figure 3.6(a) and Figure 3.6(b). From behavioural perspective, all three inputs of the FA are equivalent, because the FA is symmetric with respect to its inputs (Table 3.2). However, in the actual implementation of FAs, the structure is not symmetric with respect to the inputs. Different permutations of the inputs to a FA or a HA may result in different overall computation delays, dynamic power consumption, or leakage current. For instance, in the implementation shown in Figure 3.6(b), transitions at the inputs A and B have to pass through two XOR gates to reach the output *S*, while it is only one XOR gate for the input C_{in}.

A row of full-adders can be viewed as a mechanism to reduce three numbers to two numbers. An RCA turns into a CSA if carries are saved rather than propagated. Let *X*, *Y* and *Z* be the input bit-vectors to a three-operand CSA, and let *C* and *S* be the carry and sum output bit-vectors, respectively. Then it can be written:

$$X + Y + Z = S + 2C \tag{4}$$

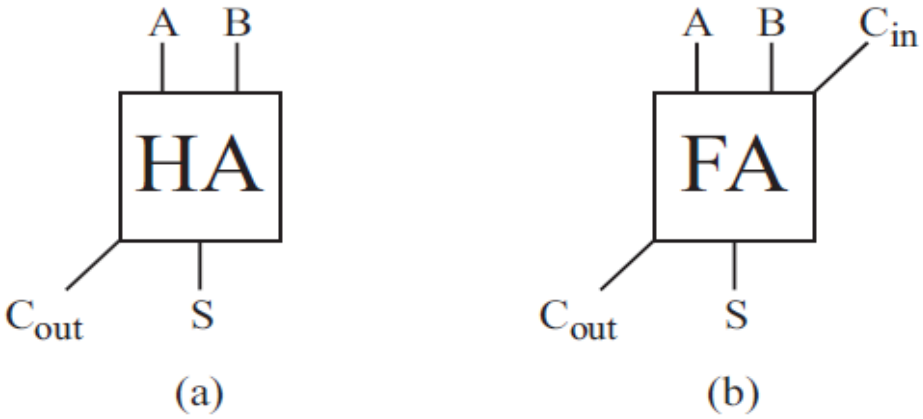


Figure 3.5 (a) A Half-Adder **(b) A Full-Adder**

In CSA, the main adder cells consist of CSA adders. RCA adders are used in the final row of adder. In this array, it also needs 3N adders to accomplish multiplication. However, delay time needed in the worst case is (N+2) full adder delay. The use of CSA is shown in figure 3.7.

Table 3.1 Truth table and behavioural representation of Half-Adder

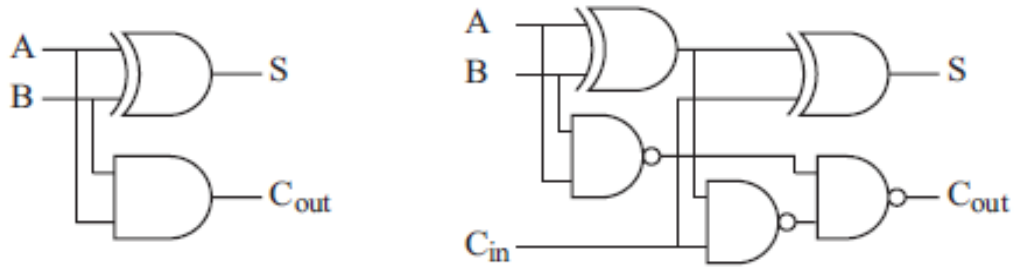
A	B	S	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$S = A \oplus B$$
$$C_{in} = A \cdot B$$

Table 3.2 Truth Table and behavioural representation of Full-Adder

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

$$S = A \oplus B \oplus C$$
$$C_{in} = A \cdot B + B \cdot C_{in} + C_{in} \cdot A$$



(a) Half-adder **(b) Full-adder**

Figure 3.6 A typical gate-level implementation of full-adder and half-adder.

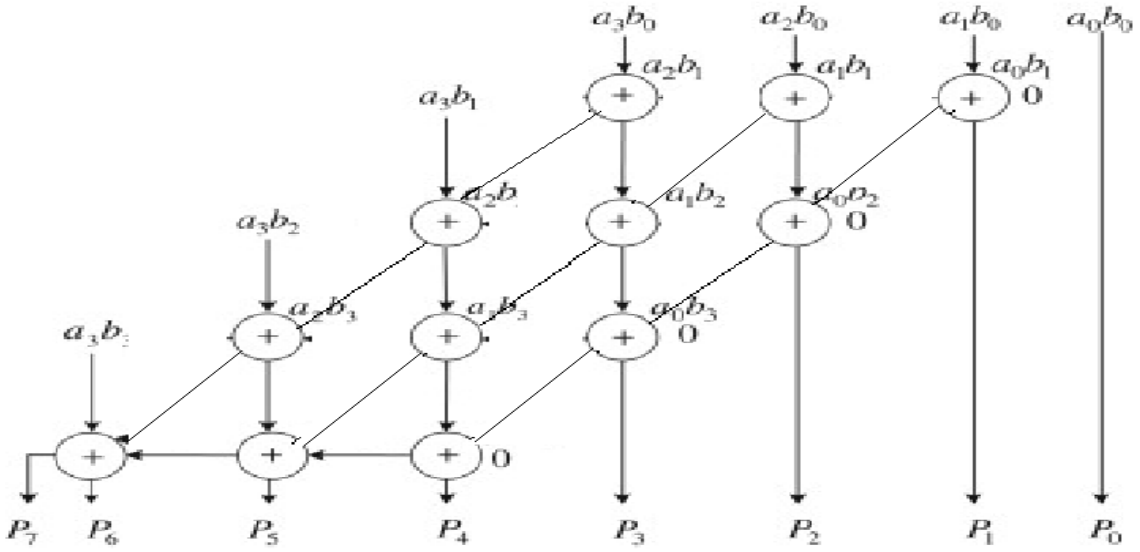


Figure: 3.7 Accumulation of PPs using Carry Save Adder [11].

3.4 Bypassing Techniques

Switching component of the power is the dominating factor in the calculation of total power in CMOS circuit. The power consumption can be reduced by reducing the switching activity of a given logic circuit without changing its function. An obvious method to reduce the switching activity is to shut down the idle part of the circuit, which is not in operating condition. So, to reduce switching activity, there are some bypassing techniques which skip the addition when the partial product of a row or column is zero.

A general parallel array multiplier generates the partial products first and then computes these partial products in parallel and then by shifting and accumulating the partial products, thus producing the result. Switching activity is poorly correlated with the input bit coefficient. In particular, reducing the switching activity of the component used in the design can minimize the power dissipation. Array multiplier has a limitation that if the input bit coefficient is zero or the partial product terms are 0 then it can't stop the switching activity resulting in unnecessary time delay. So to reduce these switching activities, bypassing techniques were introduced. The schematic diagram of an Array multiplier is shown in figure 3.8.

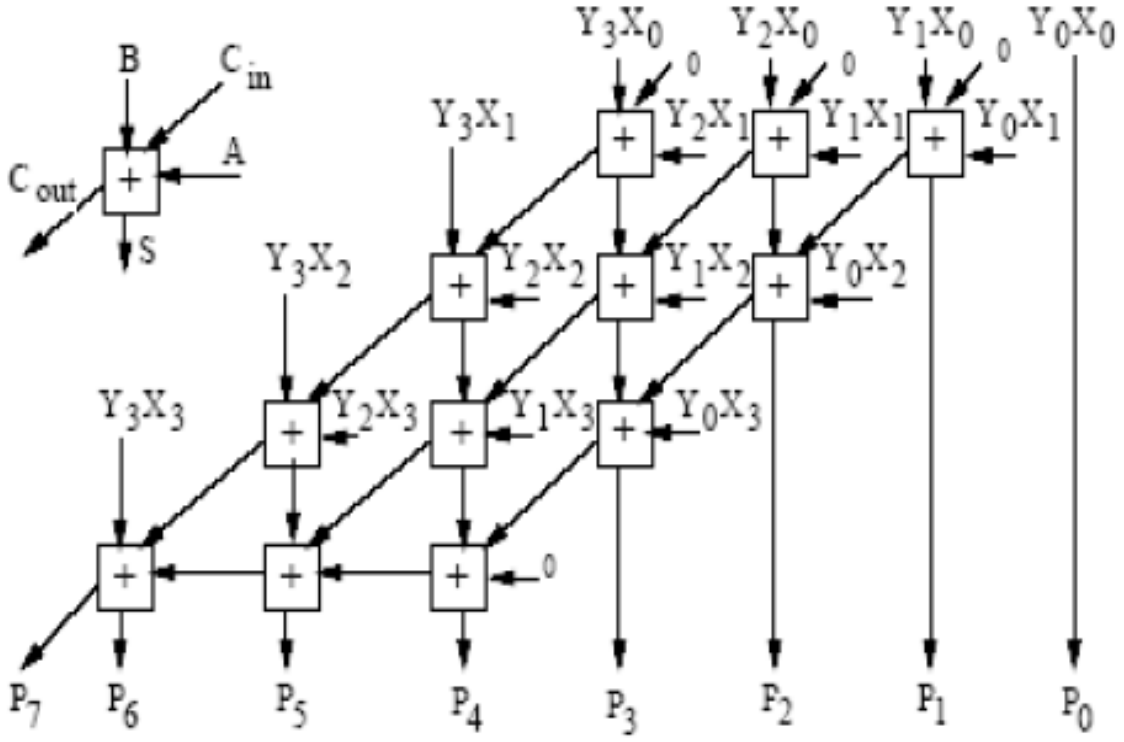


Figure 3.8 Schematic diagram of Array multiplier [15].

Now, to avoid the unnecessary computations, adders are disabled without affecting the result, this approach is known as bypassing techniques. These bypassing techniques are of two types:

1. 1-D Bypassing techniques
2. 2-D Bypassing techniques

3.4.1 1-D Bypassing Techniques

1-Dimensional bypassing techniques include row bypassing technique and column bypassing technique. Bypassing scheme is determined by the input bit coefficient. If the input bit coefficient is 0, corresponding row or column need not to be activated. If the multiplicand or multiplier term contain more no of zeros, there will be higher power reduction.

(i) Row Bypassing Technique

In row bypassing multiplier, if the multiplier bit is zero, the addition operation corresponding to that row is disabled i.e. if the multiplier bit b_j is 0 then by disabling the addition operation in j-th row, switching activity can be reduced and hence reduction in power dissipation can be achieved.

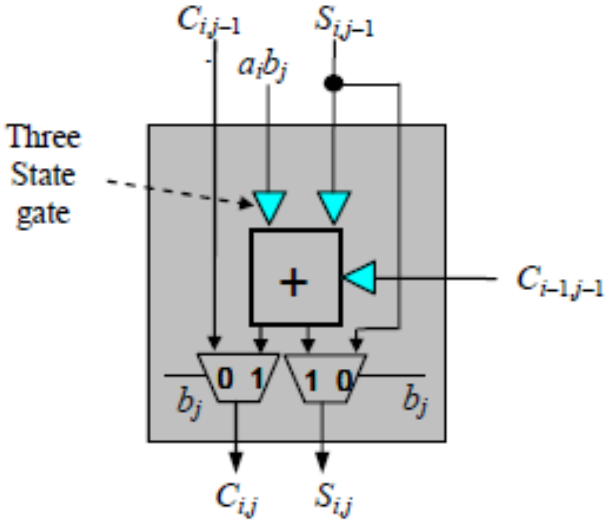


Figure: 3.9 The internal circuit of the FA cell [15].

That means in the multiplier if a bit is zero then that row of adders will get disabled. Let us take an example of 1011 x 1010. Here, in the first and the third positions, multiplier bit consists of zero. During multiplication, adders of first and third row get disabled and the previous sum is taken as the current sum. The structure for row-bypassing is illustrated in figure 3.9.

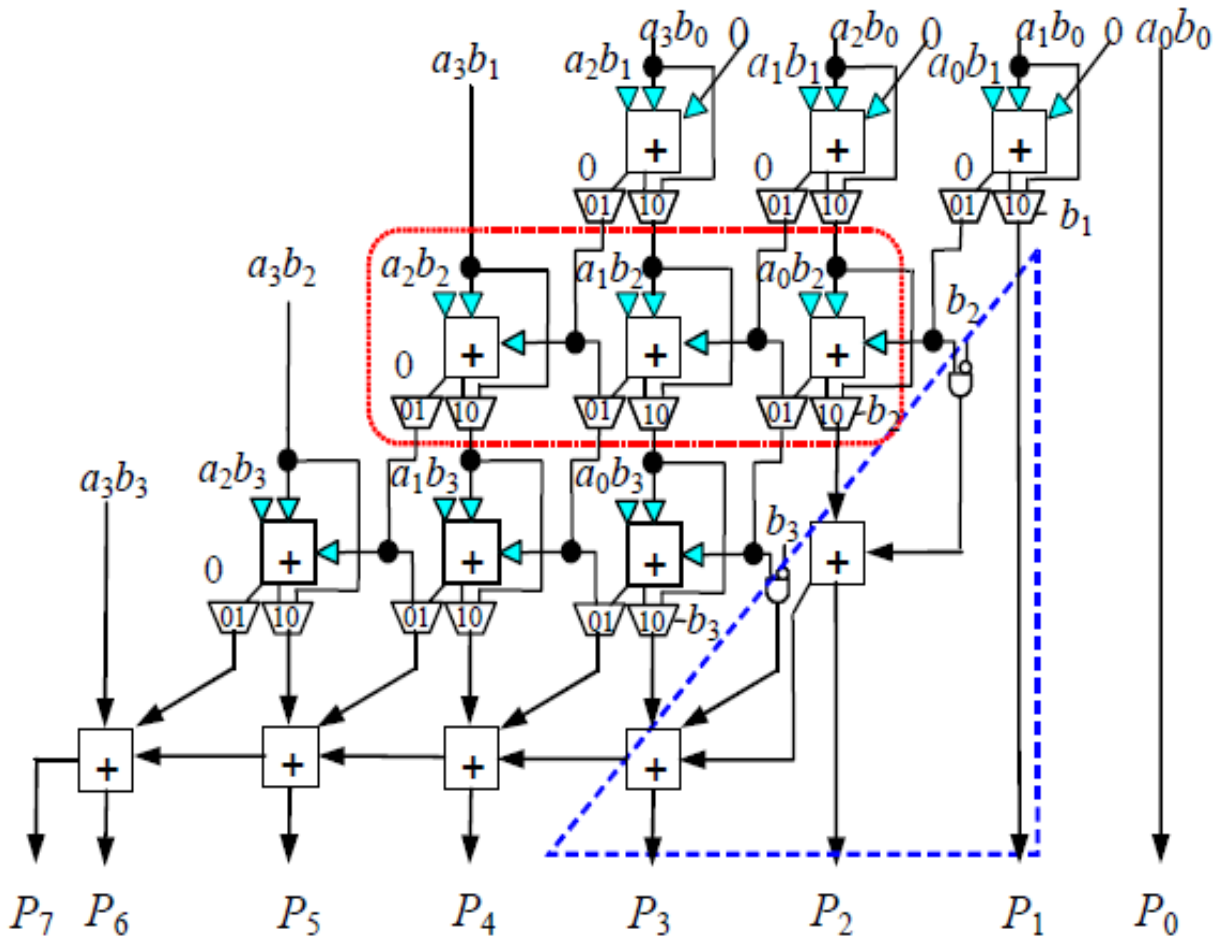


Figure 3.10 A 4×4 Braun multiplier with row-bypassing [9].

Here, instead of full adders, a special circuitry called as adding cell as shown in figure 3.10 is used. It consists of tri-state gates, full adder and multiplexers. The partial products to be summed up are given as inputs to the full adder through three state gates. The corresponding multiplier bit is the enable input to the three state gates and multiplexers. If this multiplier bit is 0 then the tri-state gates goes into high impedance state and thus inputs are not given to the full adder. The previous sum is taken as the present sum. If this bit is 1 then the tri-state gates get enabled and the inputs are given to the full adder. Thus the sum is generated and this is taken as the present sum.

The three state gates in this adding cell will enabled only when b_j =1 and then the adder will get input. If bj =0 then the previous carry and previous sum will be taken as the present carry and sum. Thus, this adding cell (AC) performs the row bypassing. In this way the switching activity can be reduced if the multiplicand bit is zero. Thus, in row bypassing multiplier, switching activity is less than that of Braun multiplier. But the only disadvantage of this row

bypassing multiplier is that it needs extra circuitry than Braun multiplier. This limitation can be overcome by the column bypass multiplier.

Advantages:

Helps in power reduction as well as area, as number of computations performed were reduced as the multiplier bit is 0.

Limitations:

As the bypassing of the row occurs, the previous carry is transferred to next row, which requires extra circuitry.

(ii) Column Bypassing Technique:

Let us take the multiplication of 1010 x 1000. Since the multiplicand bit consists of zeros in the first and third position, so, corresponding columns will get disabled. Now, consider another example of 1111 x 1000. Since multiplicand contains no zero so, all columns will get switched. The structure of column bypassing-based multiplier is shown in figure 3.11.

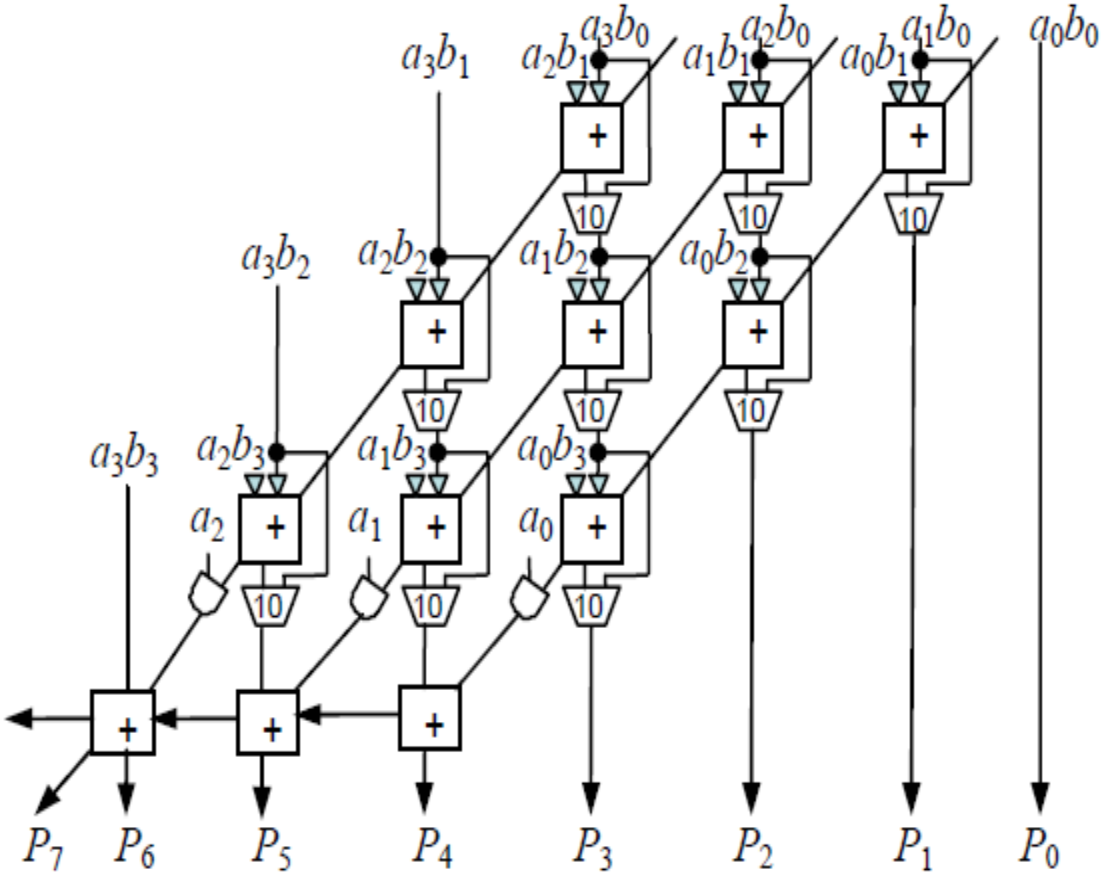


Figure 3.11 4x4 Schematic Diagram of Column bypassing multiplier [2].

Consider a multiplier with 'a' as multiplier bit and 'b' as a multiplicand bit. For a low-power column bypassing-based multiplier, if the multiplicand bit a_i is zero, the addition operation in the $(i+1)^{th}$ can be bypassed by disabling the adders of that column. i.e. all the partial products a_ib_j are zero where $0<j<n-1$. Modified full adder in the multiplier design is simpler than that of the adder used in the row bypassing multiplier. The structure for modified full adder is shown in figure 3.12. Each modified FA in CSA array is connected to two tri-state buffers and one 2-to-1 multiplexer. As the multiplicand bit is 0, their inputs in the $(i+1)^{th}$ column will be disabled and carry out in the column must be set to zero for producing the correct output. Hence, the process can be corrected by adding an AND gate at the output of the last row of CSAs. However, the extra correcting circuitary is not needed in the multiplier design in row bypassing.

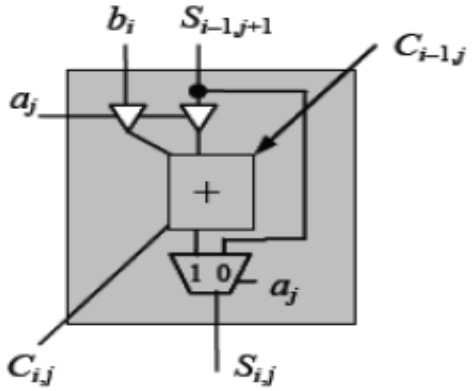


Figure 3.12 Modified Full Adder Cell for Column Bypassing [12].

In this modified full adder, there will be enabling of three states only when $a_j=1$ and then only inputs enter into the adder cell. If $a_j=0$ then the previous sum is taken as present sum. Thus column bypassing can be done by this adder cell.

There are two advantages to this approach:

1. It eliminates the extra correcting circuit as used in row bypassing.
2. The modified FA is simpler than that used in the row-bypassing multiplier, i.e. it requires only one multiplexer than two, as in row bypassing. This results in less power and area.

3.4.2 2-D Bypassing Techniques

2-Dimensional Bypassing Techniques is based on the concept of row and column bypassing. This technique detects the nullity of the partial products as well as the multiplicand at the same time to determine whether the adding cells on the corresponding row and those on the

corresponding column are skipped or not, respectively. However, the quantities of extra bypass logic decrease the capacity of reducing the power dissipation and also the area overhead is large.

(i) Simple 2-D Bypassing Technique

For a low-power 2-dimensional bypassing-based multiplier, the addition operations in the(i+1)-th column or the j-th row can be bypassed if the bit, x_i, in the multiplicand is 0 or the bit, y_j, in the multiplier is 0. In other words, as fast as the y_j is found to be 0, the results from the previous column are passed to the next column. However, there is a conflict when one adding cell, $AC_{ij,}$ encounters a condition that $x_i=y_j=0$.

Lets us assume that i=2, j=1 and $x_2 = y_1= 0$ in figure 3.13 , then according to 1-dimensional bypass method, row 2 and column 1 are expected to be bypassed. If the carry output of the adding cell AC_{12} is 1, it should be propagated to carry input of AC_{31} and then its carry output. However, if $y_1=0$, there is bypassing of adder cell AC_{31} and consequently the carry will be lost which results in an error since the output from the carry of AC_{31} will be 0. Since the column 0 is bypassed, the carry in will be missed by the adder cell AC_{30}. Hence, the bypassing circuit must be included in certain adding cells to form a correct adder cell (AC).

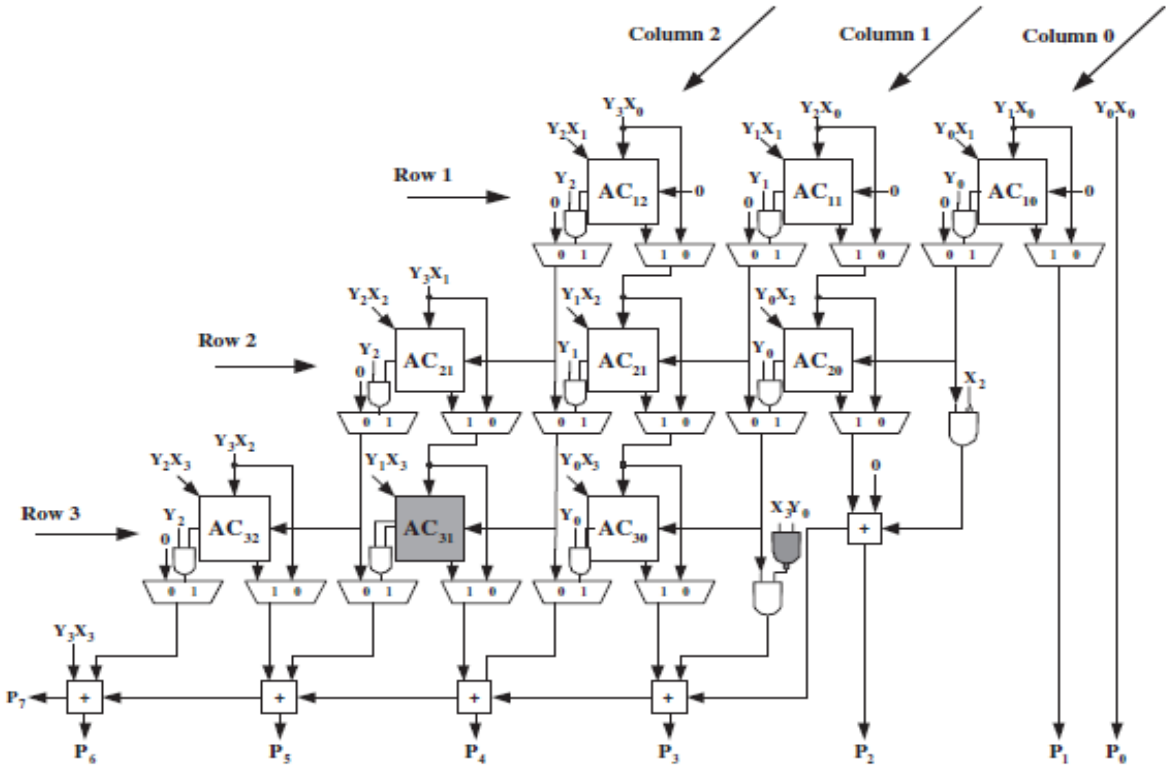


Figure 3.13 A 4x4 2-dimensional bypassing-based multiplier with correct adder cell [12].

In the adding cell with bypass logic, a simple rule is given that the adding cell, $AC_{ij,}$ can not be bypassed if x_i is not equal to 0 and the carry in is 1. The adding cell AC_{30} can be further simplified to save the bypass logic area. For this, the adding cell with bypass logic is replaced by a single NAND gate. The structure for new adding cells is shown in figure 3.14 and 3.15

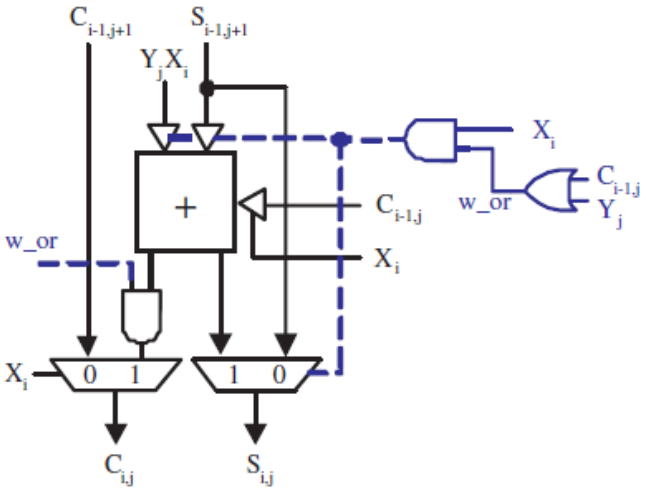


Figure 3.14 Adding cell with Bypass logic [12].

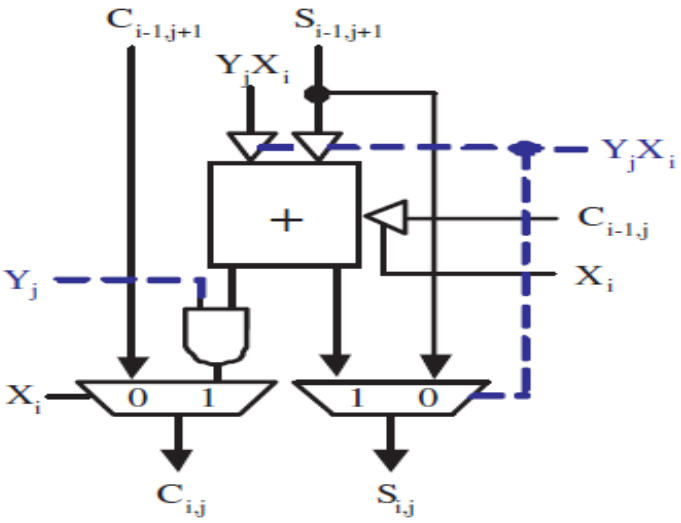


Figure 3.15 Adding cell without bypass logic [12].

(ii) 2-Dimensional Bypassing using both (A+B+1) and (A+1) adders

According to the bypassing technique described in section 3.4.1, if the multiplicand bit, $a_{i,}$ is 0 or the multiplier bit, $b_{j,}$ is 0 then addition operation can be bypassed in the $(i+1)^{th}$ column or the j^{th} row. However, to add the bypassed carry result into the multiplication result, the extra correcting circuitary is applied in the row-bypassing multiplier. To eliminate the extra correcting circuitary

in the design of 2-Dimensional bypassing process, the previous row carry must be integrated in the 2-dimensional bypassing process. So in this multiplier, if the product, a_ib_j, is 0 and the carry bit, $c_{i,j-1}$, is 1, there is bypassing of addition operation in the (i+1,j) FA. In other words, if the product, a_ib_j, is 1 or the bit, $c_{i,j-1}$, is 1, the addition operation can be executed in the (i+1, j) FA.

By using the bypassing condition, the $(i+1, j)^{th}$ FA only performs the A+1 operation if the product, a_ibj, is zero and the carry bit, $c_{i,j-1}$, is 0, or a_ib_j, is 0 and carry bit, $c_{i,j-1}$, is 1. On the other hand, if the product, a_ib_j, is 1 and the carry bit, $c_{i,j-1}$, is 1, the $(i+1,j)^{th}$ FA only performs the A+2 operation. The AND result of a_ib_j, and $c_{i,j-1}$, can replace the carry bit in the (i+1,j) FA and half adder, A+B+1 can replace the (i+1,j) FA where n>j>1. The first row of half adders of CSAs is replaced by the incremental half adders, A+1 due to the 2-dimensional bypassing process. Apart from that, each A+1 adder contains one tri-state buffer and two 2to1 multiplexer and each A+B+1 adder contains two tri-state buffers and two 2-to-1 multiplexers. The logical circuit for the various adders used is shown in fig. 3.17 and a 4x4 2-dimensional bypassing multiplier using A+1 and A+B+1 adders is shown in figure 3.16.

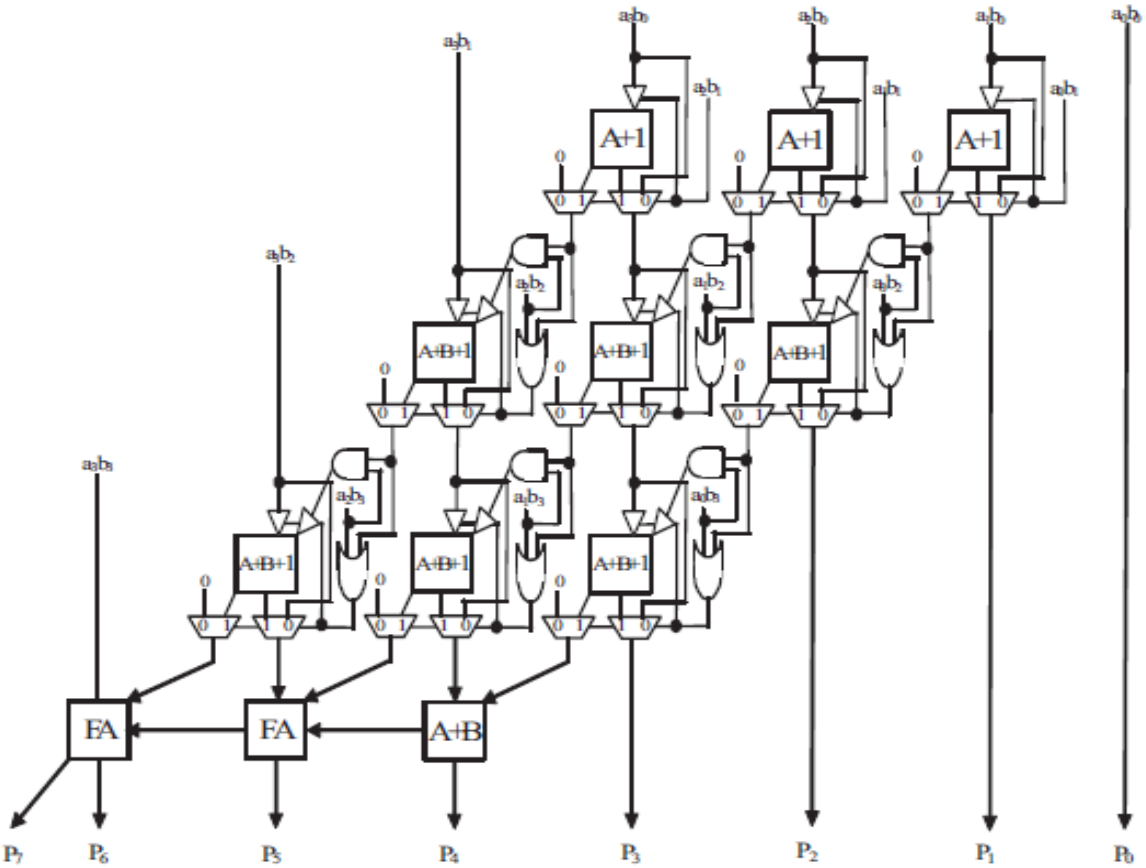


Figure 3.16 A 2-dimensional bypassing multiplier using A+1 and A+B+1 adder [6].

This type of bypassing multiplier achieves higher power reduction with lower area overhead. The area consumed by this multiplier is less than that of row bypassing, column bypassing or in simple 2-dimensional bypassing multiplier and the power dissipation is also less.

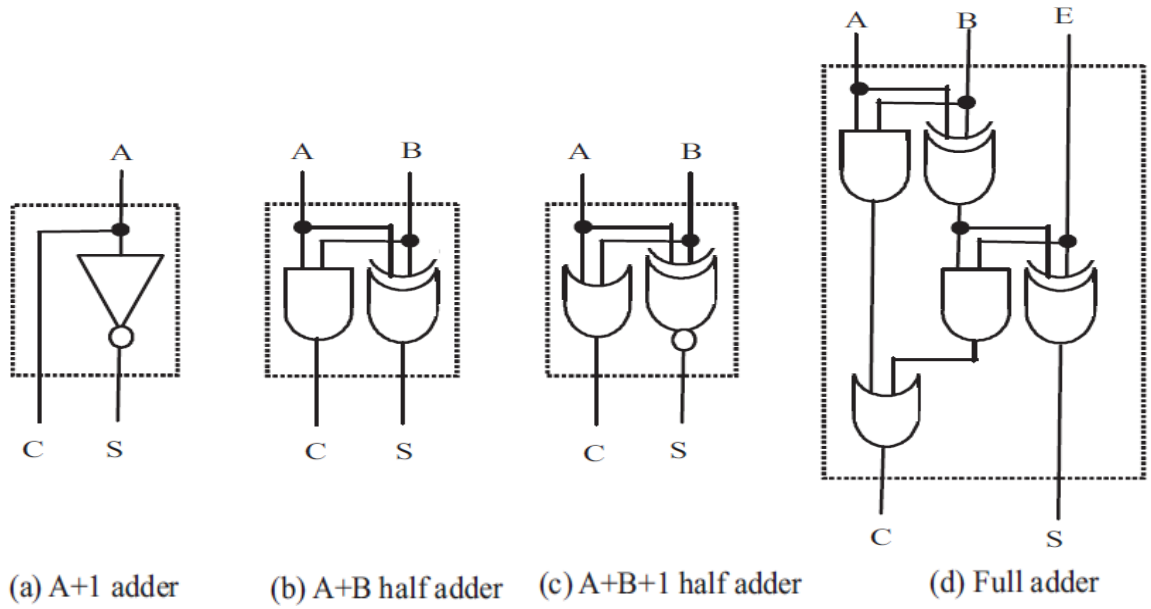


Figure 3.17 Logical circuits for the different adders.

(iii) 2-Dimensional bypassing multiplier using only (A+1) adder:

In this multiplier, only A+1 half adders are used instead of A+B+1 half adders and full adders. Here if the product, a_ib_j, matches with the carry bit, $c_{i,j-1}$, the addition operation in the (i+1,j)th FA can be bypassed. In other words, if there is mismatch of the product, a_ib_j, and the carry bit, $c_{i,j-1}$, the $(i+1,j)^{th}$ FA must perform the addition operation. This can be done by XORing of a_ib_j and $c_{i,j-1}$.

In this multiplier, if there is mismatch of product, a_ib_j, and carry bit, $c_{i,j-1}$, the $(i+1, j)^{th}$ FA only performs the A+1 operation . on the other hand , the $(i+1,j)^{th}$ FA perform the addition by 2 or o when the product, a_ib_j, matches with the carry bit, $c_{i,j-1}$. So, in the $(i+1,j)^{th}$ FA ,the resultant carry bit, $c_{i+1,j}$, can be bypassed from the previous carry bit, $c_{i,j-1}$ and a low cost incremental adder can replace the $(i+1, j)^{th}$ FA. Apart from that, each modified adder, A+1 contains one tri state buffer and two 2-to-1 multiplexers. In the same way, a low cost incremental adder, A+1, can replace the HA with bypassing condition as $a_ib_j=0$. In figure 3.18, the modified design of a HA and FA is shown. A 4x4 low-cost bypassing based multiplier using only A+1 half adder is

illustrated in figure 3.19. In this multiplier, higher power reduction can be achieved in comparison to all other multipliers discussed above. The area overhead is also less in this case.

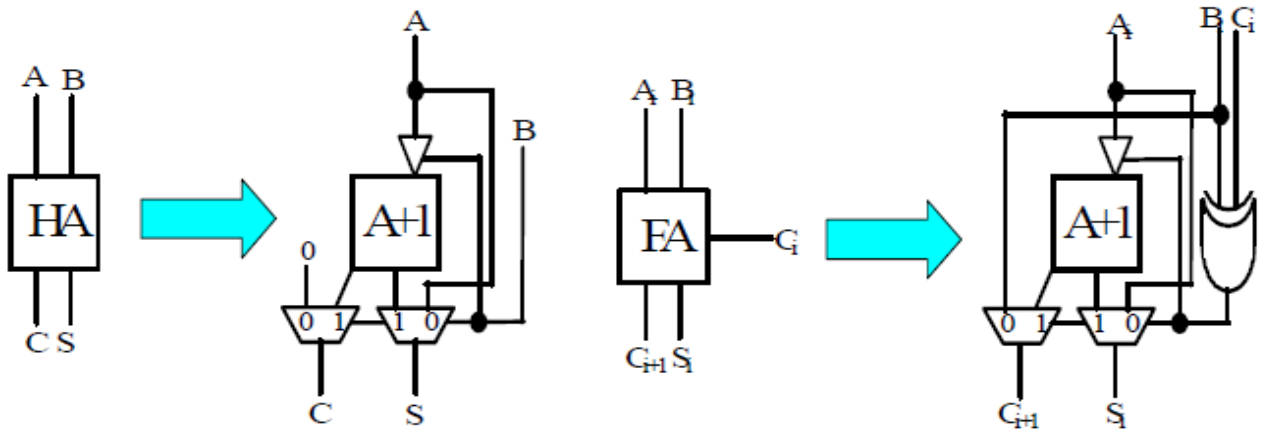


Figure 3.18 Modified half adder and a full adder.

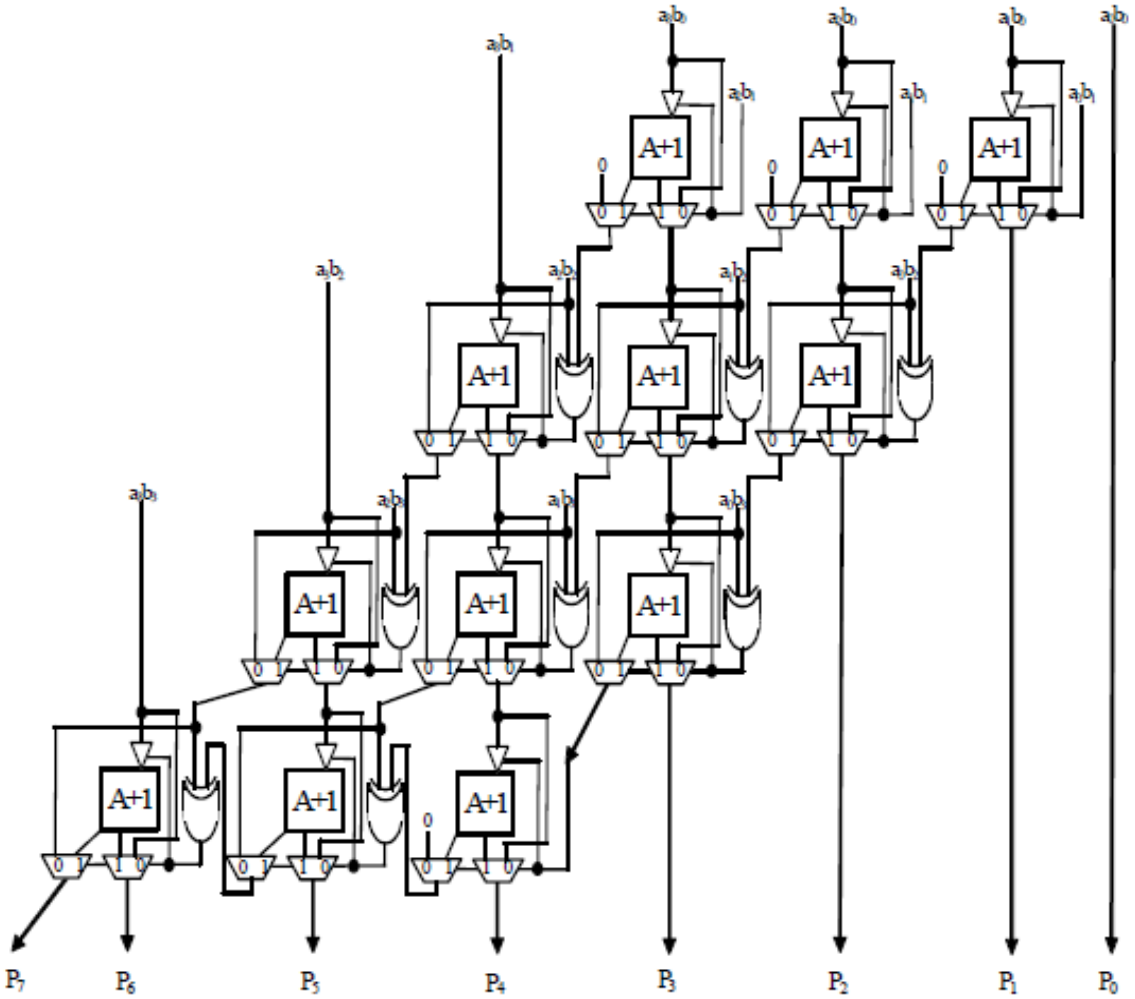


Figure 3.19 A 4x4 2-dimensional bypassing based multiplier using only A+1 half adder [6].

CHAPTER 4

FPGA IMPLEMENTATION

4.1 FPGA Implementation

The FPGA that is used for the implementation of the circuit is the Xilinx Spartan 3E (Family), XC3S5000 (Device). The working environment/tool for the design is the Xilinx ISE 8.2i is used for FPGA Design flow of Verilog code.

4.1.1 Overview of FPGA Design Flow

As the FPGA architecture evolves and its complexity increases. Today, most FPGA vendors provide a fairly complete set of design tools that allows automatic synthesis and compilation from design specifications in hardware specification languages, such as Verilog or VHDL, all the way down to a bit stream to program FPGA chips. A typical FPGA design flow includes the steps and components shown in Figure 4.1. Inputs to the design flow typically include the HDL specification of the design, design constraints, and specification of target FPGA devices. We further elaborate on these components of the design input in the following: Design constraints typically include the expected operating frequencies of different clocks, the delay bounds of the signal path delays from input pads to output pads (I/O delay), from the input pads to registers (setup time), and from registers to output pads (clock-to-output delay). In some cases, delays between some specific pairs of registers may be constrained.

The second design input component is the choice of FPGA device. Each FPGA vendor typically provides a wide range of FPGA devices, with different performance, cost, and power tradeoffs. The selection of target device may be an iterative process. The designer may start with a small (low capacity) device with a nominal speed-grade. But, if synthesis effort fails to map the design into the target device, the designer has to upgrade to a high-capacity device. Similarly, if the synthesis result fails to meet the operating frequency, he has to upgrade to a device with higher speed-grade. In both the cases, the cost of the FPGA device will increase in some cases by 50% or even by 100%. This clearly underscores the need to have better synthesis tools since their quality directly impacts the performance and cost of FPGA designs.

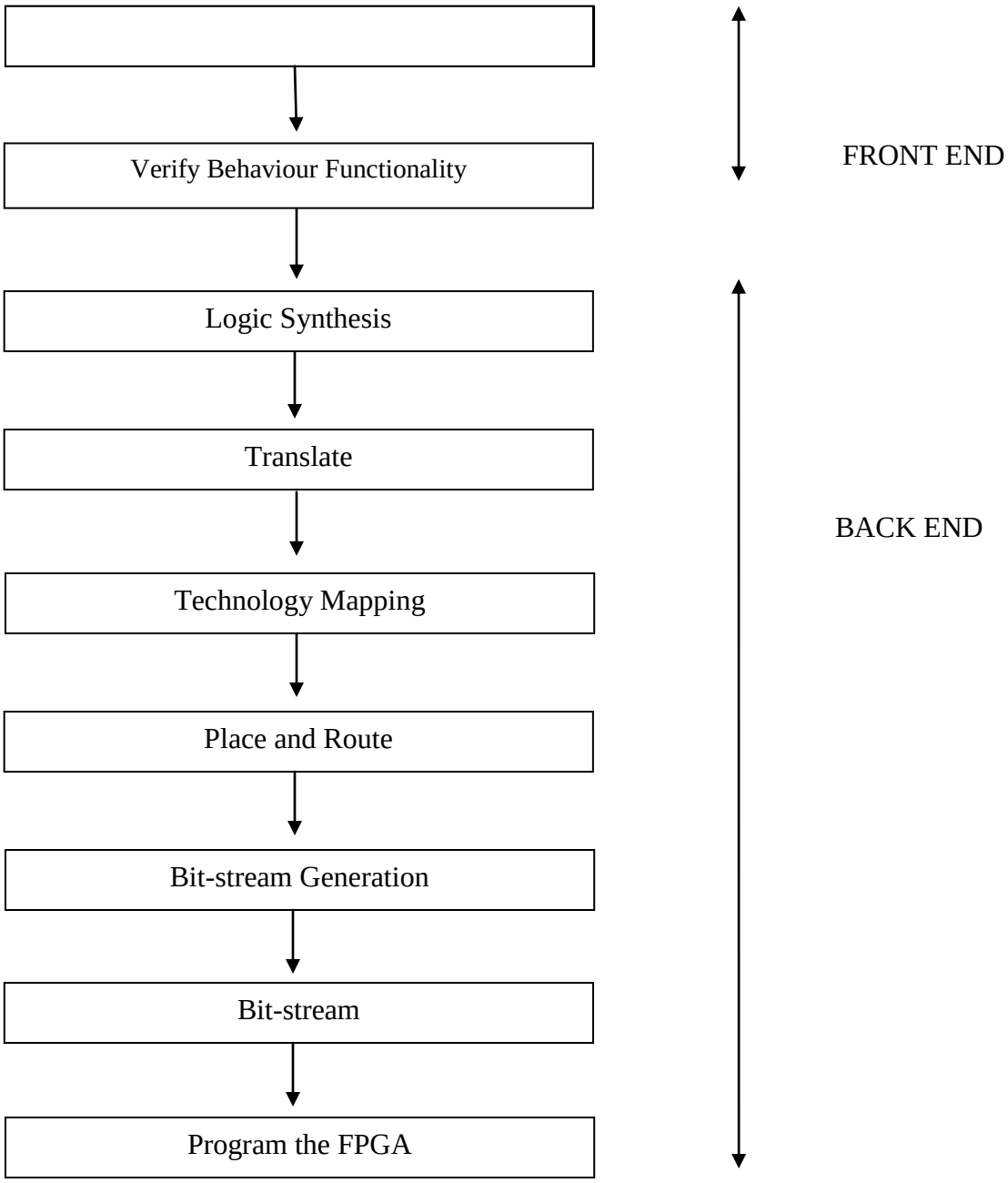


Figure 4.1 FPGA Design Flow.

Design Entity

The basic architecture of the system is designed in this step which is coded in a Hardware Description Language like Verilog or VHDL. A design module is split into two parts, each of which is called a design unit in Verilog. The module declaration represents the external interface to the design module. The module internals represents the internal description of the design module-its behaviour, its structure, or a mixture of both.

Behavioural Simulation

After the design phase, create a test bench waveform containing input stimulus to verify the functionality of the Verilog code module using a simulation software i.e. Modelsim SE for different inputs to generate outputs and if it verifies then proceed further, otherwise modifications and necessary corrections will be done in the HDL code. This is called as the behavioural simulation.

Design Synthesis

After the correct simulations results, the design is then synthesized. During synthesis, the Xilinx ISE tool does the following operations:

a) HDL Compilation: The tool compiles all the sub-modules in the main module if any and then checks the syntax of the code written for the design.

b) Design Hierarchy Analysis: Analysis the hierarchy of the design.

c) HDL Synthesis: The process which translates VHDL or Verilog code into a device netlist format, i.e. a complete circuit with logical elements such as Multiplexer, Adder/subtractors, counters, registers, flip flops Latches, Comparators, XORs, tristate buffers, decoders, etc. for the design. If the design contains more than one sub designs, ex. to implement a processor, we need a CPU as one design element and RAM as another and so on, and then the synthesis process generates netlist for each design element. Synthesis process will check code syntax and analyze the hierarchy of the design which ensures that the design is optimized for the design architecture, the designer has selected. The resulting netlist is saved to an NGC (Native Generic Circuit) file (for Xilinx® Synthesis Technology (XST)).

d) Advanced HDL Synthesis: Low Level synthesis: The blocks synthesized in the HDL synthesis and the Advanced HDL synthesis are further defined in terms of the low level blocks such as buffers, lookup tables. It also optimizes the logic entities in the design by eliminating the redundant logic, if any. The tool then generates a 'netlist' file (NGC file) and then optimizes it. The final netIist output file has an extension of .ngc. This NGC file contains both the design data and the constraints. The optimization goal can be pre-specified to be the faster speed of operation or the minimum area of implementation before running this process. The level optimization effort can also be specified. The higher the effort, the more optimized is the design but higher effort can also be specified. The higher the effort, the more

optimized is the design but higher effort requires larger CPU time (i.e. the design time) because multiple optimization algorithms are tried to get the best result for the target architecture.

Design Implementation

The design implementation process consists of the following sub processes:

1. Translation: The Translate process combines all the input netlists and constraints to a logic design file. This information is saved an a NGD (Native Generic Database) file. This can be done using NGD Build program and .ngd file describes the logical design reduced to the Xilinx device primitive cells. Here, defining constraints is nothing but, assigning the ports in the design to the physical elements (ex. pins, switches, buttons etc) of the targeted device and specifying time requirements of the design. This information is stored in a file named UCF (User Constraints File). Tools used to create or modify the UCF are PACE, Constraint Editor Etc.

2. Mapping: The Map process is run after the Translate process is complete. Map process divides the whole circuit with logical elements into sub blocks such that they can be fit into the FPGA logic blocks. That means map process fits the logic defined by the NGD file into the targeted FPGA elements (Combinational Logic Blocks (CLB), Input Output Blocks (lOB)) and generates an NCD (Native Circuit Description) file which physically represents the design mapped to the components of FPGA. MAP program is used for this purpose.

3. Place and Route: Place and Route (PAR) program is used for this process. The place and route process places the sub blocks from the map process into logic blocks according to the constraints and connects the logic blocks. Example if a sub block is placed in a logic block which is very near to IO pin, then it may save the time but it may affect some other constraint. So trade-off between all the constraints is taken account by the place and route process .The PAR tool takes the mapped NCD file as input and produces a completely routed NCD file as output. Output NCD file consist the routing information.

4. Bit-stream Generation: The collection of binary data used to program the reconfigurable logic device is most commonly referred to as a "bit-stream," although this is somewhat misleading because the data are no more bit oriented than that of an instruction set processor and there is generally no "streaming." While in an instruction set processor the configuration

data are in fact continuously streamed into the internal units, they are typically loaded into the reconfigurable logic device only once during an initial setup phase.

5. Functional Simulation: Post-Translate (functional) simulation can be performed prior to mapping of the design. This simulation process allows the user to verify that the design has been synthesized correctly and any differences due to the lower level of abstraction can be identified.

6. Static timing analysis: Three types of static timing analysis can be performed that are:

(i) Post-fit Static timing analysis: The timing results of the Post-fit process can be analyzed. The Analyze Post-Fit Static timing process opens the timing Analyzer window, which interactively select timing paths in design for tracing.

(ii) Post-Map Static Timing Analysis: Analyze the timing results of the Map process. Post Map timing reports can be very useful in evaluating timing performance. Although route delays are not accounted for the logic delays can provide valuable information about the design. If logic delays account for a significant portion (>50%) of the total allowable delay of a path, the path may not be able to meet your timing requirements when routing delays are added. Routing delays typically account for 45% to 65% of the total path delays. By identifying problem paths, redesign the logic paths to use fewer levels of logic, tag the paths for specialized routing resources, move to a faster device, or allocate more time for the path. If logic-only-delays account for much less (35%) than the total allowable delay for a path or timing constraint, then the place-and-route software can use very low placement effort levels. In these cases, reducing effort levels allow to decrease runtimes while still meeting performance requirements.

(iii) Post Place and Route Static Timing Analysis: Analyze the timing results of the Post-Place and Route process. Post-PAR timing reports incorporate all delays to provide a comprehensive timing summary. If a placed and routed design has met all of timing constraints, then proceed by creating configuration data and downloading a device. On the other hand, identify problems and the timing reports, try fixing the problems by increasing the placer effort level, using re-entrant routing, or using multi-pass place and route. Redesign the logic paths to use fewer levels of logic, tag the paths for specialized routing resources, move to a faster device, or allocate more time for the paths.

(iv) Timing Simulation: Perform Post-Place and Route simulation after the design has been

and routed. After the design has been through all of the Xilinx implementation tools, a timing simulation netlist can be created. This simulation process allows tosee how the design will behave in the circuit. Before performing this simulation it will benefit to create a test bench or test fixture to apply stimulus to the design. After this a .ncd file will be created that is used for the generation of power results of the design.

CHAPTER 5

RESULTS & CONCLUSION

This chapter discusses the implementation, simulation and synthesis results of different bypassing-based multipliers. Multipliers synthesized using Xilinx ISE 14.5 targeting Sparten-3E (XC3S500E-4FG320) FPGA.

5.1 Simulation & Synthesis results

After implementing the design using Verilog HDL, the verification of their functional correctness is done through behavioural simulation in Xilinx ISE 14.5. The simulation results for 8x8 2-dimensional bypassing-based multiplier using (A+1), (A+B+1) HAs and FAs is shown in figure 5.1

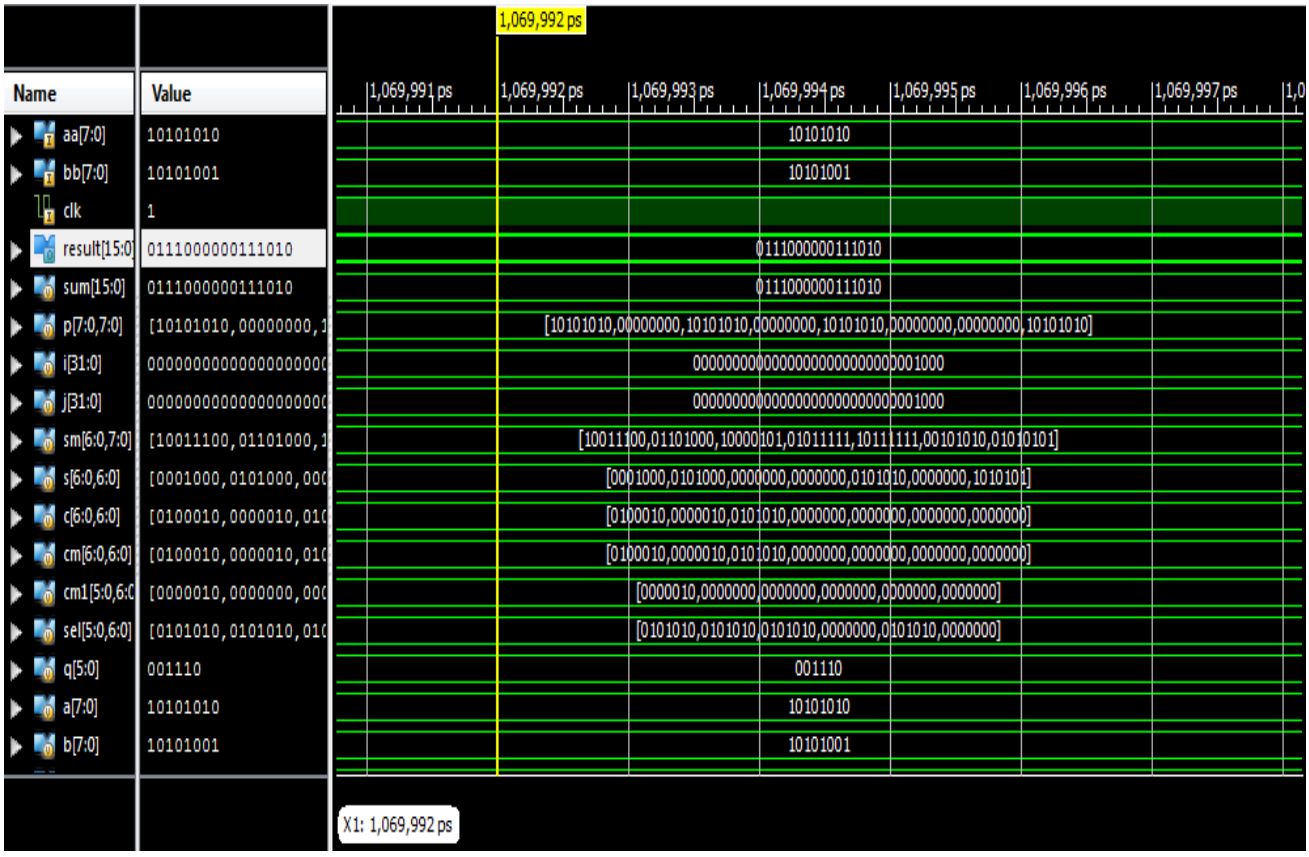


Fig. 5.1 A 2-Dimensional bypassing-based 8x8 multiplier using (A+1) and (A+B+1) HAs.

The design of Row bypassing, Column bypassing, standard 2-Dimensional bypassing multipliers, 2-Dimensional bypassing multiplier using (A+1) and (A+B+1) HAs and 2-Dimensional bypassing multiplier using only (A+1) HAs are simulated using Verilog HDL.

These designs are synthesized and implemented in Xilinx ISE 14.5 targeting Sparten-3E (XC3S500E-4FG320) FPGA. Comparison of delay, area utilization and dynamic power is shown in table 5.1. The graphical representation is shown in figure 5.2, 5.3 and 5.4.

Table 5.1 Comparison of Area utilization in terms of no of slices, Total delay and Dynamic power of multipliers for 4x4 and 8x8 using different bypassing techniques

Sr. No.	Bypassing Techniques	No. of slices (out of 4656) (4x4/8x8)	Total Delay (in ns) (4x4/8x8)	Dynamic Power (mW) (4x4/8x8)
1.	[2]	14(0%)/73(1%)	4.487/11.941	0.018/0.024
2.	[13]	13(0%)/67(1%)	4.273/11.382	0.018/0.021
3.	[12]	14(0%)/85(1%)	5.708/13.272	0.016/0.019
4.	[6]	13(0%)/47(1%)	5.979/15.994	0.015/0.018

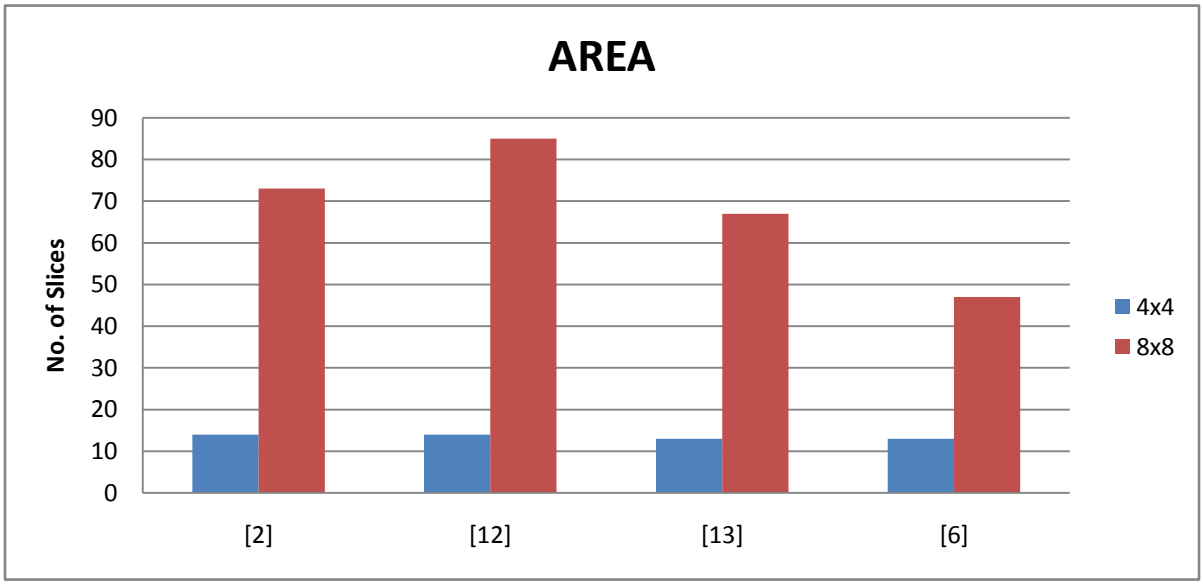


Fig. 5.2 Area comparison of different bypassing based multipliers for 4x4 and 8x8 graphically

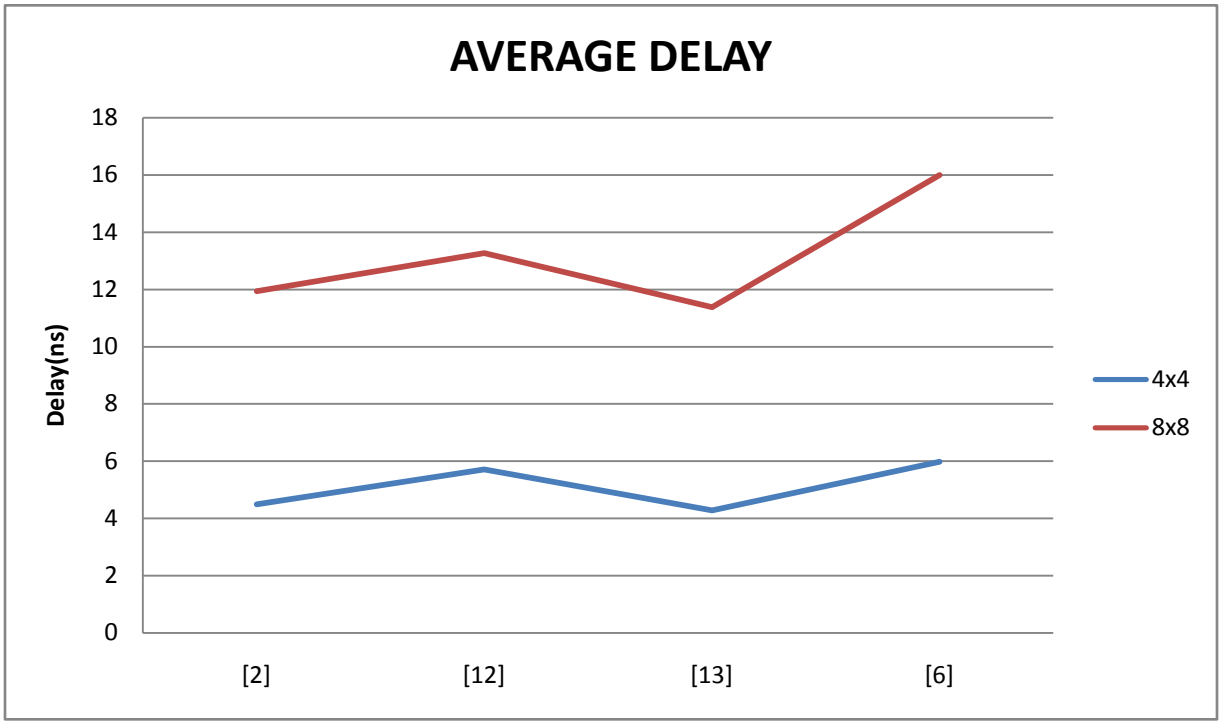


Figure 5.3 Delay Comparison of different bypassing-based multipliers for 4x4 and 8x8 graphically.

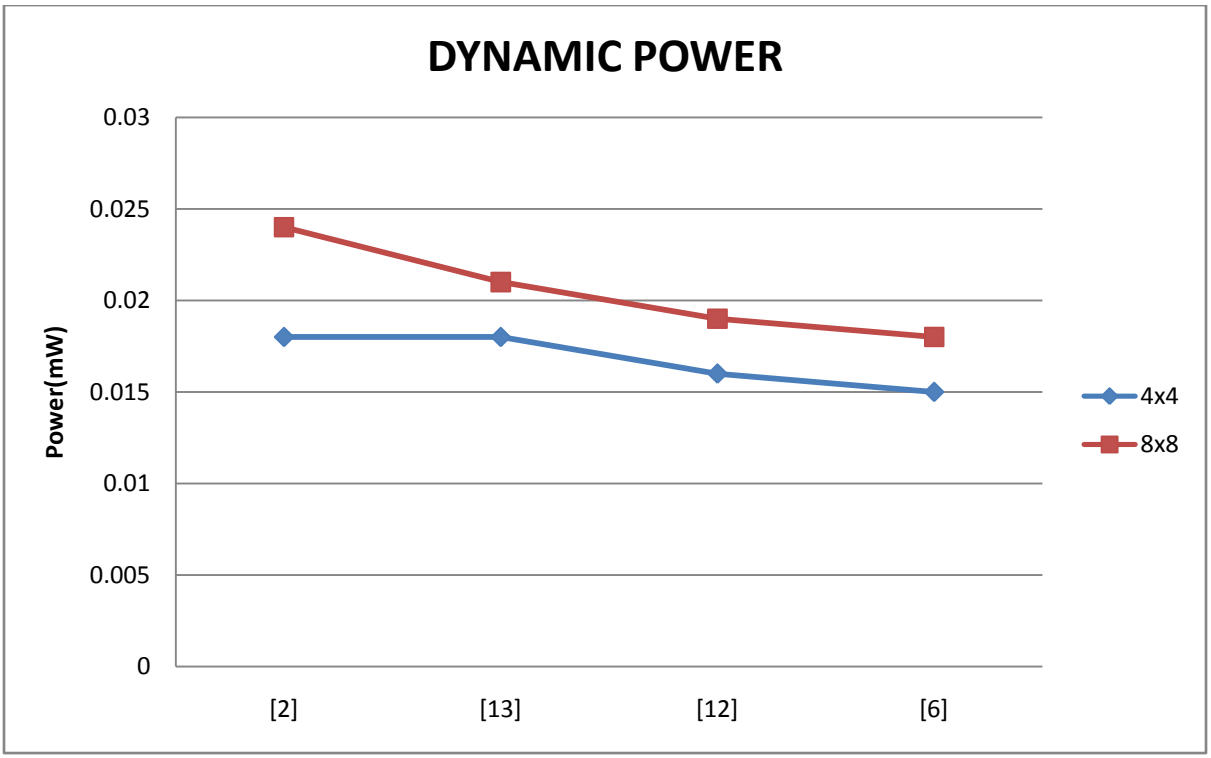


Figure 5.4 Dynamic Power comparison of different bypassing based multipliers for 4x4 and 8x8 graphically.

5.2 Conclusion

Multiplier is the building block in processors. After putting a lot of efforts, we learnt the importance of the multiplier and the usage of bypassing techniques in the multiplier. These bypassing techniques improve the efficiency of the multiplier. There are two types of bypassing techniques:

1. 1-D bypassing
2. 2-D bypassing

A 1-D bypassing involves row bypassing and column bypassing. Row bypassing is implemented when the multiplier bit is 0. It is better in terms of power than the conventional multiplier. Due to the problem of carry propagation, row bypassing results in usage of extra circuitary. Column bypassing is implemented when the multiplicand bit is 0. This technique is efficient in terms of both area and power, as the MFA of the column bypassing uses only one multiplexer than that of row bypassing which uses two multiplexer. It also eliminates the usage of extra circuitary. To further improve the efficiency and reduce the switching activity, 2-D bypassing is implemented which involves both row as well as column bypassing. Thus, this results in less power dissipation than that of row bypassing and column bypassing individually. Based on the simplification of addition operations, a 2-D bypassing-based multiplier using (A+1), (A+B+1) HAs and FAs, a 2-D bypassing based multiplier using only (A+1) HAs are implemented. On comparing, the results shows that the 2-D multiplier using both (A+1) and (A+B+1) HAs gives the minimum delay while the 2-D bypassing-based multiplier using only (A+1) HAs gives minimum power dissipation. Also, the area utilization is minimum for the multiplier using only (A+1) HAs. Since the area overhead is more in other bypassing-based multipliers. So the extra transistors also consume more power. In other words, the more the number of additional transistors in the bypassing-based multiplier design, the lesser the ability of power reduction. Since the area overhead is least in case of multiplier using only (A+1) HAs, so it achieves higher power reduction.

5.3 Future Scope

The present work on the multipliers can be extended in various directions. Based on the study, literature survey and understanding established, the following suggestions are proposed:

- To implement the 2-Dimensional bypassing techniques along with the booth algorithm.

- To design a bypassing multiplier which focus on both, generation as well as accumulation stage.
- In most of the designs, main focus is given to reduce power of the multiplier through row or column bypassing, but the improvement in speed of the multiplier is not taken into much consideration.

LIST OF PUBLICATION

- Sunil Kumar and Sakshi Bajaj, " Comparative Analysis of different Bypassing-based Multipliers," in *proc. International Conference on Innovations in Electricals and Computer Science Engineering(IEECSE-2014)*, pp. 34 -37, 22 June 2014.

REFERENCES

[1] Tushar V. More and Dr. R.V. Kshirsagar, "Design of Low Power Column Bypass Multiplier using FPGA," in *proc. IEEE international conference on Electronics Computer Technology(ICECT)*, vol. 3, pp. 431-435, April 2011.

[2] Ming-Chen Wen, Sying-Jyan Wang and Yen-Nan Lin, " Low-power Parallel Multiplier with Column Bypassing," in *proc. IEEE International Symposium on Circuits and Systems(ISCAS)*, vol. 2, pp. 1638-1641, May 2005.

[3] Sunjoo Hong, Taehwan Roh and Hoi-Jun Yoo, " A 145 μW 8x8 Parallel Multiplier based on Optimized Bypassing Architecture," in *proc. IEEE International Symposium on Circuits and Systems(ISCAS)*, pp. 1175-1178, May 2011.

[4] Syed Ershad Ahmed, Sibi Abraham, Shreehari Veeramanchaneni, N. Moorthy Muthukrishnan and M.B. Srinivasan, " A Modified Twin Precision Multiplier with 2D Bypassing technique," in *proc. International Symposium on Electronic System Design(ISED)*, pp. 102-106, December 2012.

[5] Alvin Joseph J.Tang and Joy Alinda Reyes, "Comparative Analysis of Low Power Multiplier Architectures," in *proc. Modelling Symposium (AMS)*, pp. 270-274, May 2011.

[6] Jin-Tai yan and Zhi-Wei Chen, "Low-Cost Low-Power Bypassing-Based Multiplier Design," in *proc. IEEE International Syposium on Circuits and Systems (ISCAS)*, pp. 2338-2341, June 2010.

[7] Ko-Chi Kuo and Chi-Wen Chou, "Low Power Multiplier with Bypassing and Tree structure," in *proc. IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, pp. 602-605, December 2006.

[8] Shweta S. Khobragade and Swapnali P. Karmore, "Review on: Low Power Design of Modified Booth Multiplier" in *proc. International Journal of Engineering and Advanced Technology (IJEAT)*, ISSN: 2249-8958, vol.2, Issue-5, June 2013.

[9] Y.T. Hwang, J.F. Lin, M.H. Sheu and C.J. Sheu, "Low-Power Multiplier Design with Row and Column Bypassing," in *proc. IEEE International SOC Conference (SOCC)*, pp. 345-349, June 2005.

[10] R. Anitha and V. Bagyaveereswaran, "Comparative study of Braun's Multiplier using FPGA Devices," in *proc. International Journal of Engineering Science and Technology (IJEST),*vol. 3, no. 6, June 2011.

[11] T. Francis, T. Joseph and J. K. Antony, "Modified MAC unit for low power high speed DSP application using multiplier with bypassing technique and optimized adders," in *Proc. IEEE International conference on Computing, Communications and Networking Technologies (ICCCNT),* pp. 1-4, July 2013.

[12] Gang-Neng Sung, Yan-Jhih Ciou and Chua-Chin Wang, "A Power-Aware 2-Dimensional Bypassing Multiplier Using Cell-Based Design Flow," in *proc. IEEE International Symposium on Circuits and Systems (ISCAS),* pp. 3338-3341, May 2008.

[13] Jin-Tai Yan and Zhi-Wei Chen, "Low-Power Multiplier Design with Row and Column Bypassing," in *proc. IEEE International SOC conference(SOCC),* pp. 227-230, September 2009.

[14] Ying-Tsung Hwang, Jin-Fa Lin, Ming-Hwa Sheu and Chia-Jen Sheu, "Low Power Multiplier Designs Based on Improved Column Bypassing Schemes," in *proc. IEEE Asia Pacific Conference on Circuits and Systems (APCCAS),* pp. 594-597, December 2006.

[15] Jun-ni Ohban, Vasily G. Moshnyaga and Koji Inoue, "Multiplier Energy Reduction Through Bypassing of Partial Products," in *proc. IEEE Asia-Pacific Conference on Circuits and Systems,* vol. 2, pp. 13-17, 2002.

[16] J. Selvakumar and V.Bhaskar, "A Low Power Multiplier Architecture based on Bypassing technique for digital filter," in *proc. International conference on Sustainable Energy and Intelligent Systems (SESICON),* pp. 260-263, June 2011.

[17] Z.Huang and M. D. Ercegovac, "Two-Dimensional Signal Gating for Low-Power Array Multiplier Design," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS),* vol. 1, pp. 1985-1997, 2002.

[18] G. Economakos, D. Bekiaris and K. Pekmestzi, " A Mixed Style Architecture for Low Power multipliers based on a Bypass Technique," in *Proc. IEEE International Conference on Design and Technology of Integrated Systems in Nano scale Era(DTIS),* pp. 1-6, March 2010.