A

Project Report

On

"DESIGN OF BYPASSING MULTIPLIER"

Submitted towards the partial fulfillment of requirement for the award of degree of

Master of Technology

In

VLSI Design and CAD

Submitted by:

Manchal Ahuja

Roll No: 601161020

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)

DECLARATION

I hereby declare that the work which is being presented in the dissertation entitled, **"Design of Bypassing Multiplier"** in partial fulfilment of the requirement for the award of 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 dissertation has not been submitted in any other University/Institute for the award of degree.

Date: 12-07-2013

(MANCHAL AHUJA)
Roll No: 601161020

It is certified 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:

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. Rajesh Khanna** as well as **PG Coordinator, Dr. Kulbir Singh**, **Associate Professor**,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.

Manchal Ahuja

(601161020)

ABSTRACT

A multiplier is one of the key hardware blocks in most digital and high performance systems such as FIR filters, digital signal processors and microprocessors etc. With advances in technology, many researchers have tried and are trying to design multipliers which offer either of the following- high speed, low power consumption, regularity of layout and hence less area or even combination of them in multiplier. Thus making them suitable for various high speed, low power, and compact VLSI implementations. However area and speed are two conflicting constraints. So improving speed results always in larger areas. This project tries to find out the best trade off solution among the both of them. Generally multiplication goes in two basic steps. Partial product generation and accumulation.

In this project, implementation of the bypassing technique has helped in reduction of the power and area. The row bypassing is better as compared to conventional multiplier. On the implementation of column bypassing multiplier the extra circuitry needed is eliminated. The row and column bypassing multiplier results in the reduced switching activity. Thus,less power consumption.

TABLE OF CONTENTS

ABBREVIATIONS

Abbreviation	Meaning
PP	Partial Product
FA	Full Adder
DCT	Discrete Cosine Transform
2-D	Two Dimensional
1-D	One Dimensional
CSA	Carry Save Array
RCA	Ripple Carry Array
TFA	The delay of full adder on the path between its specified input and output
CLA	Carry-Lookahead-Adder
P	Group Carry Propagate
G	Group Carry Generate
CNSA	Conditional Sum Addition
CSLA	Carry-Select Adder
LS	Least Significant
MS	Most Significant
DSP	Digital Signal Processor

AC	Adder Cell
MFA	Modified Full Adder
BL	Bypass logic

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Digital signal processing algorithms typically require a large number of mathematical operations to be performed quickly and repeatedly on a series of data samples. Signals (perhaps from audio or video sensors) are constantly converted from analog to digital, manipulated digitally, and then converted back to analog form. Many DSP applications have constraints on latency, that is for the system to work, the DSP operation must be completed within some fixed time, and deferred (or batch) processing is not viable.

Most general-purpose microprocessors and operating systems can execute DSP algorithms successfully, but are not suitable for use in portable devices such as mobile phones and PDAs because of power supply and space constraints. A specialized digital signal processor, however, will tend to provide a lower-cost solution, with better performance, lower latency, and no requirements for specialized cooling or large batteries. The architecture of a digital signal processor is optimized specifically for digital signal processing. Most also support some of the features as an applications processor or microcontroller, since signal processing is rarely the only task of a system.

Multiplication is an essential arithmetic operation in DSP applications In digital design, multipliers are one of the most important blocks. They are used commonly in different processes like digital signal processes (DSPs), digital filter.. Other applications of multiplication are for multimedia applications such as 3D graphics and signal processing systems, and are required to execute large numbers of multiplications. Digital signal processing (DSP) is one of most important units in electronic devices. DSP performs fundamental operations which include video processing for displaying streamline image and baseband processing for communication operations.

Many adder and multiplier designs have been proposed in the past to satisfy various design objectives such as higher speed, smaller area and lower power. Even though earlier designs focused more on minimizing Silicon area, the focus has shifted, now more towards speed and power. Besides adders, digital multipliers are the most critical arithmetic functional unit in many DSP applications, e.g., Fourier Transform, DCT, digital filtering, etc. Array and parallel multipliers are very welcomed due to their high execution speed and throughput besides its high regularity.

In microprocessors multiplication operation is performed in a variety of forms in hardware and software depending on the cost and transistor budget allocated for this particular operation. In the beginning stages of computer development any complex operation was usually programmed in software or coded in the micro-code of the machine. Some limited hardware assistance was provided. Today it is more likely to find full hardware implementation of the multiplication in order to satisfy growing demand for speed and due to the decreasing cost of hardware.

With the high demanding of electronic portable devices, the requirement of low power device is getting more attention in recent years. The primary concern of electronic portable device is to extend operating hours without changing the battery residing in device. Although advanced technology enhances battery life to operate for longer hours, the complicated operations in the high-end portable devices are still power hungry and is critical for the low power design. Low power design can be achieved at system, logic, technology, architecture, and the circuit levels. Power saving can be significant if the design is planned in the earlier stage at system level. Optimizing logic level of circuit is also critical for the low power design. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. The exponential growth of electronic equipments in recent years has brought forward new challenges to the integrated circuit design community. Adding more and more functionality with lot of real-time computation, while operating at the maximum speeds requires revolutionary changes at all levels of chip design. For example, to implement a specific function at architecture level, ripple-carry, carry-save, or carry look-ahead adder can be adopted. By choosing one of these architectures, low power consumption can be achieved by trading off with other specifications such as speed or chip area.
This project was carried using HDL Verilog on Xilinx 8.2i.

1.1 Organisation of Report

Chapter 1 of the report, briefly explains about the applications of multiplier. Chapter 2 describes the multiplication concept and adders. In chapter 3, bypassing techniques are described. Chapter 4 is about results and simulation. The last, chapter 5 gives the conclusion of the report.

CHAPTER 2

LITERATURE REVIEW

Yin-Tsung Hwang ,Jin-Fa Lin ,Ming-Hwa Sheu and Chia-Jen Sheu [1] Proposed two novel low power multipliers circuits based on enhanced row bypassing schemes. The essence of the power saving idea is eliminating unnecessary computation via signal bypassing**.**

For a low-power row-bypassing multiplier, the addition operations in the j-th row can be disabled to reduce the power dissipation if the bit bj in the multiplier is 0, i. e., all partial products aibj, $0 \leq i \leq n-1$, are zero. As a result, the addition operations in the jth row of CSAs can be bypassed and the outputs from the (j-1)-th row of CSAs can be directly fed to the (j+1)-th row of CSAs without affecting the multiplication result.

Tushar V. More and Dr. R.V.Kshirsagar [2] Presented a low power column bypass multiplier design methodology that inserts more number of zeros in the multiplicand thereby reducing the number of switching activities as well as power consumption. The switching activity of the component used in the design depends on the input bit coefficient. This means if the input bit coefficient is zero, corresponding row or column of adders need not be activated. If multiplicand contains more zeros, higher power reduction can be achieved. Further low power adder structure reduces the switching activity.

Jin-Tai Yan and Zhi-Wei Chen [3] Presented a low-power multiplier design with row and column bypassing, method based on the simplification of the incremental adders and half adders instead of full adders in an array multiplier, is shown in Fig. 2.1**.** Compared with the row bypassing multiplier, the column-bypassing multipliers and the 2-dimensional bypass multiplier, the proposed multiplier reduces 25.7% of the power dissipation with only 15% hardware overhead on the average for multipliers.

The extra correcting circuits in the row-bypassing multiplier are applied to add the bypassed carry results into the multiplication result. To eliminate the extra correcting circuits and design the 2-dimensional bypassing process, the carry result in the previous row must be integrated in the 2-dimensional bypassing process. Therefore, the addition operation in the FA can be bypassed in the proposed bypassing multiplier if the product is 0 and the carry bit is 0,and if the product is 1 or the carry bit is 1, the addition operation in the (i+1), j FA can be executed.

Sunjoo Hong, Taehwan Roh, and Hoi-Jun Yoo [4] Proposed a low-power parallel multiplier based on optimized bypassing architecture (OBA) is proposed. The proposed OBA

has two kinds of adder cells to reduce power consumption by 15.7 %. One is the two-dimensional

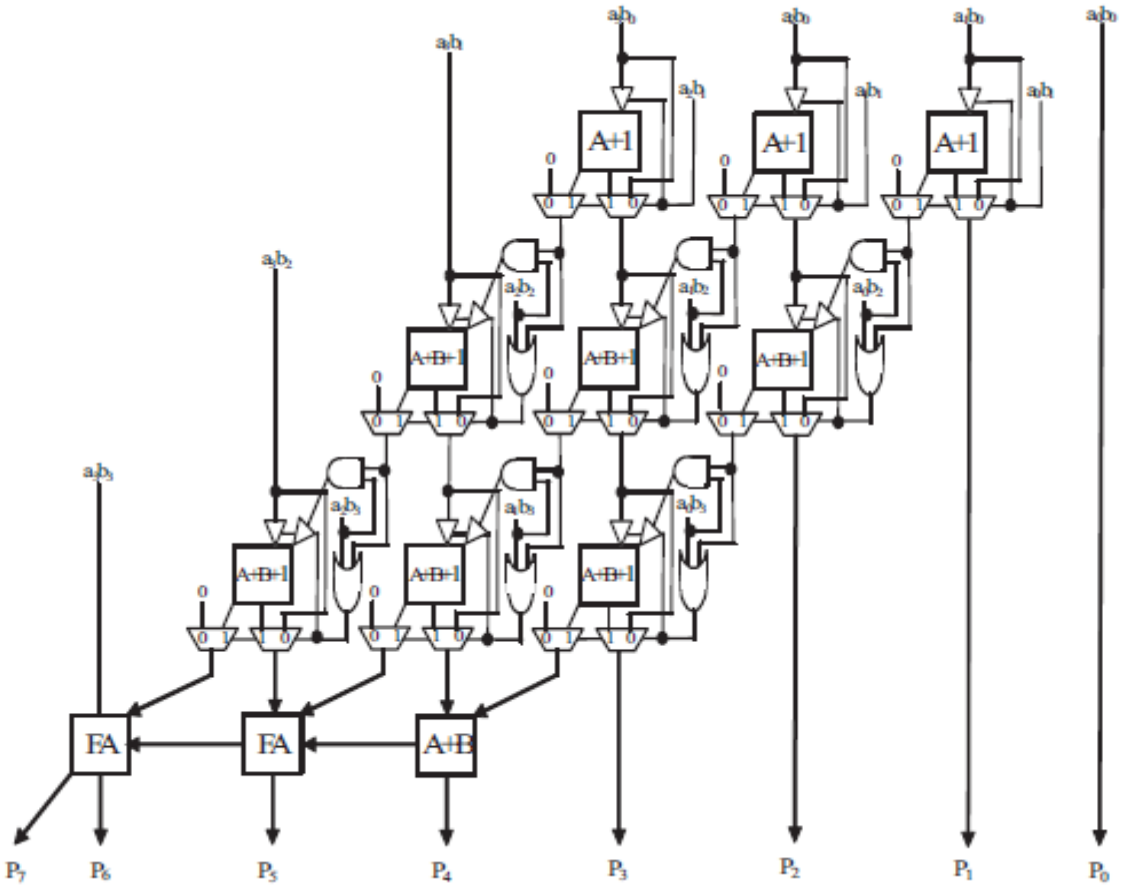


Figure 2.1: Braun multiplier using Row and Column bypassing with MFA. [3]

bypassing adder (TDBA) which performs both row and column bypassing scheme simultaneously, and the other is the modified row-bypassing adder (MRBA) for the proposed row-bypassing scheme.. Since the TDBA and MRBA have no FAs but ITBs to prevent logic evaluation when the logic is not used, they can achieve 33.7 %, 32.0 % power reduction, respectively, compared to the FA. The implementation is shown in Fig. 2.2

J. Selvakumar, Vidhyacharan Bhaskar [5] Presented a low power 4×4 digital multiplier design to reduce power consumption of digital multiplier based on 2-dimensional bypassing method. Design of portable battery operated multimedia devices requires energy-efficient multiplication circuits. The proposed bypass cells constitute the multiplier skip redundant signal transitions when the horizontally partial product or the vertical operand is zero. Hence, it is a 2-dimensional bypassing architecture using which we designed a Digital filter for low power dissipation in signal processing applications. By a small area penalty, we gain more

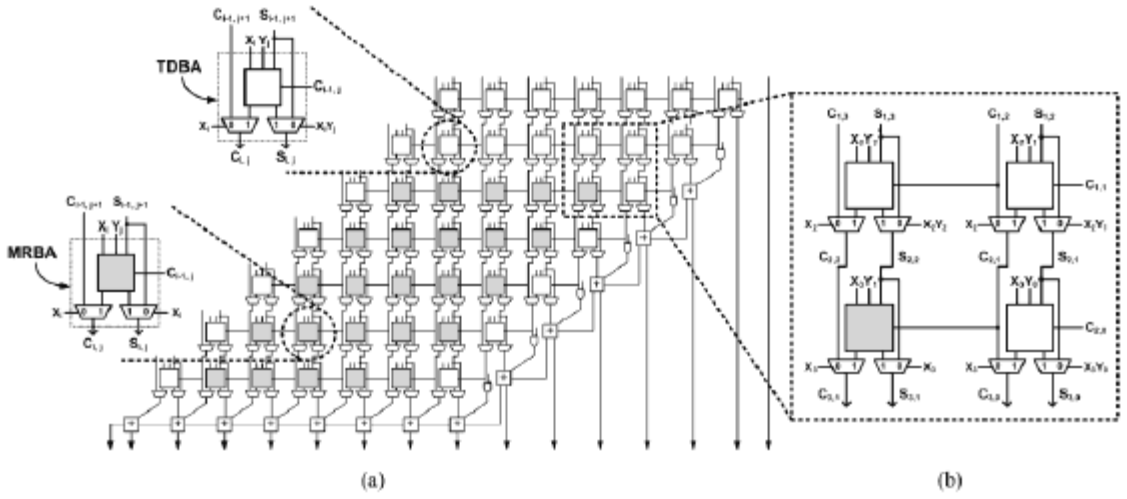


Figure 2.2 Multiplier based on OBA [4]

than 70% power saving compared to the prior designs for both 1D multiplier and conventional filter.

Ugurdag , F.Keskin.O,Tunc C,Temizkan.F,Fici,G,Dedeoglu, S [6] Proposed a high-performance hardware design, employing the component level optimizations for building fast multipliers. Most of the fast multiplier architectures uses some form of Carry Save Adder (CSA) Tree, which is also called Column Compression (CC). They proposed a new method called RoCoCo (Row and Column Compression), which also compresses the tree along rows so that the final adder is small and fast. Although CC results in faster multipliers in ASIC implementations, it is an assumption by designers that they are not the wisest choice on FPGAs. On the contrary, they were able to show through Xilinx synthesis results that RoCoCo frequently offer faster multipliers than the built-in implementation of the multiply operation in Xilinx ISE synthesis tool.

Sangjin Hong, Suhwan Kim, Marios C. Papaefthymiou and Wayne E. Stark [7] Presented a novel multiplier design methodology for performing the coefficient multiplications with very low power dissipation. Given hounds on the throughput and the quantization error, this approach scales the original coefficients to enable the partitioning of each multiplication into a collection of smaller multiplications with short critical paths. Dissipation is further reduced by disabling the multiplier rows that do not affect the

multiplication's outcome. We have used our methodology to design a low- power parallel multiplier for the Fast Fourier Transform. Simulation results show that our approach can result in significant power savings over conventional multipliers. Specifically, the coefficients are scaled and encoded to achieve minimum bit clustering and partitioning. In conjunction with a modified array multiplier that uses selective row disabling, this coefficient optimization scheme can result in significantly more efficient DSP computations.

Paul,B.C,Fujita.S,Okajima.M .[8] Presented a ROM-based 16 times 16 multiplier for low-power applications. The design uses sixteen 4 times 4 ROM-based multiplier blocks followed by carry-save adders and a final carry-select adder (all ROM-based) to obtain the 32 bit output. Eliminate identical rows and columns for optimizing the power and performance. Measurement results show a 40% reduction in power over the conventional carry-save array multiplier when operated at its maximum frequency. The ROM-based design also provides 44% less delay than the array multiplier with a minimal increase (7.7%) in power. This demonstrates the low-power operation of the ROM-based multiplier also at higher frequencies.

Zhijun Huang and Milos D. Ercegovac [9] Proposed two-dimensional (2-D) signal gating schemes for low-power array multiplier design. 2-D gating provides gating lines for both multiplicand and multiplier operands. Different regions of the multiplier are dynamically deactivated according to the actual precision of each operand. Bit-level implementation is studied in order to minimize the gating overhead and make realistic evaluation. Compared to 1-D, 2-D signal gating as better in terms of power consumption, power-delay product and power-area product. Different regions of the multiplier are dynamically deactivated according to the precision information of each variable. Compared to1-D, 2-D signal gating reduces the power dissipation by up to 35% and is also better in power-delay product and power area product.

Lid Feng, Dai Guoding, Zhuang Yiqi [10] Proposed low power DCT architecture based on scalers sharing multiplier, which reduces the computation complexity of matrix-vector multiplication by sharing a small set of products. The presented architecture also provides an easy approach for making trade off between image quality and power dissipation through scaling multiplier's precision Experimental result on a hardware FPGA platform shows that more than 35% power saving can be achieved by replacing the shift adder multipliers with the scalers sharing multipliers in the baseline design.

Mottaghi-Dastierdi.M, Afzali-Kusha.A,Pedram.M [12] Proposed, a low-power structure called bypass zero, feed A directly (BZ-FAD) for shift-and-add multipliers is proposed. The architecture considerably lowers the switching activity of conventional multipliers. The modifications to the multiplier which multiplies A by B include the removal of the shifting the B register, direct feeding of A to the adder, bypassing the adder whenever possible, using a ring counter instead of a binary counter and removal of the partial product shift. The results for 32-bit radix-2 multipliers show that the BZ-FAD architecture lowers the total switching activity up to 76% and power consumption up to 30% when compared to the conventional architecture. The proposed multiplier can be used for low-power applications where the speed is not a primary design parameter.

Rajashekhar Modugu, Yong-Bin Kim and Minsu Choi [13] Proposed efficient hardware architecture of modulo 2n + 1 multiplier is proposed and validated to address the demand. Modulo 2n+1 multiplier is one of the critical components in applications in the area of digital signal processing, data encryption and residue arithmetic that demand high-speed and low-power operation. The proposed modulo 2n+1 multiplier has three major functional modules including partial products generation module, partial products reduction module and final stage addition module. The proposed modulo 2n+1 multiplier uses novel compressor designs and sparse tree adders as primitive building blocks for fast low-power operation. The partial products reduction module is completely redesigned using the novel compressors and the final addition module is implemented using a new less complex sparse tree based inverted end-around-carry adder. The resulting modulo 2n + 1 multiplier is implemented in standard CMOS cell technology and compared both qualitatively and quantitatively with the existing hardware implementations. The unit gate model analysis and the experimental results show that the proposed implementation is faster and consume less power than similar hardware implementations making it a viable option for efficient designs.

UMA. A Vandana. AR [14] Proposed a novel VLSI SP oriented architecture for implementation of serial parallel Multipliers (SPM) . The VLSI oriented multiplier is based on a segmentation technique of SPM and the conventional full adders are replaced by low power full adder. In this paper two architectures namely Segmented Based SPM, Folded VLSI oriented segmented based SPM are compared for power and area. The proposed VLSI SP oriented architecture achieves higher throughput and less

area compared to the segmentation based serial parallel multiplier . The proposed VLSI SP oriented architecture permits the optimization of the area, speed and power.

Gang-Neng Sung, Yan-Jhih Ciou, and Chua-Chin Wang[15] Presents a low power digital multiplier design by taking advantage of a 2-dimensional bypassing method in cell-based design flow. The proposed bypassing cells constituting the multiplier skip redundant signal transitions when the horizontally partial product or the vertically operand is zero. Thorough cell-based design flow post-layout simulations show that the power delay product of the proposed 8×8 multiplier design is reduced by more than 13.8% compared to prior designs. It justifies the advantage in terms of power delay product after the software optimization.

CHAPTER 3
MULTIPLICATION

3.1 How to multiply

Sequential or bit-at –a time multiplication can be done by keeping a cumulative partial product (initialized to 0) and successively adding to it properly shifted terms xja.Since each successive number to be added to the cumulative partial product is shifted by one bit with respect to the preceding one, simpler approach is to shift the cumulative partial product by one bit in order to align its bit with those of the next partial product. Two versions of this algorithm can be devised, depending on whether the partial product terms are processed from top to bottom or from bottom to top.

In multiplication with right shifts, the partial product terms xja are accumulated from top to bottom:

$$P^{(j+1)} = (p^{(j)} + x_j a 2^k)2^{-1} \text{ with } p^{(0)} = 0 \text{ and } p^{(k)} = p$$

|----add----|

|----shift right--|

Because the right shifts will cause the first partial product to be multiplied by 2^{-k} by the time its done, we premultiply a by 2^k to offset the effect of right shifts. This premultiplication is done simply by aligning a with the upper half of the 2k- bit cumulative partial product in the addition steps(i.e.. storing a in the left half of a double-length register).

Conventional array multiplier is primarily used for computing multiplication of two input data. Two unsigned n-bits binary numbers $A=a_{n-1}a_{n-2}a_{n-3}\ldots a_0$ and $B=b_{n-1}b_{n-2}b_{n-3}\ldots b_0$ can generate a (2n−1) bit product P, which can be defined as the following:

$$P=AB= \sum_{i=0}^{n-1} a_i 2^i \sum_{j=0}^{n-1} b_j 2^j \qquad (1)$$

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. For example: P1 is the result of addition of a1b0 and a0b1. Multiplication is shown in figure 3.1.

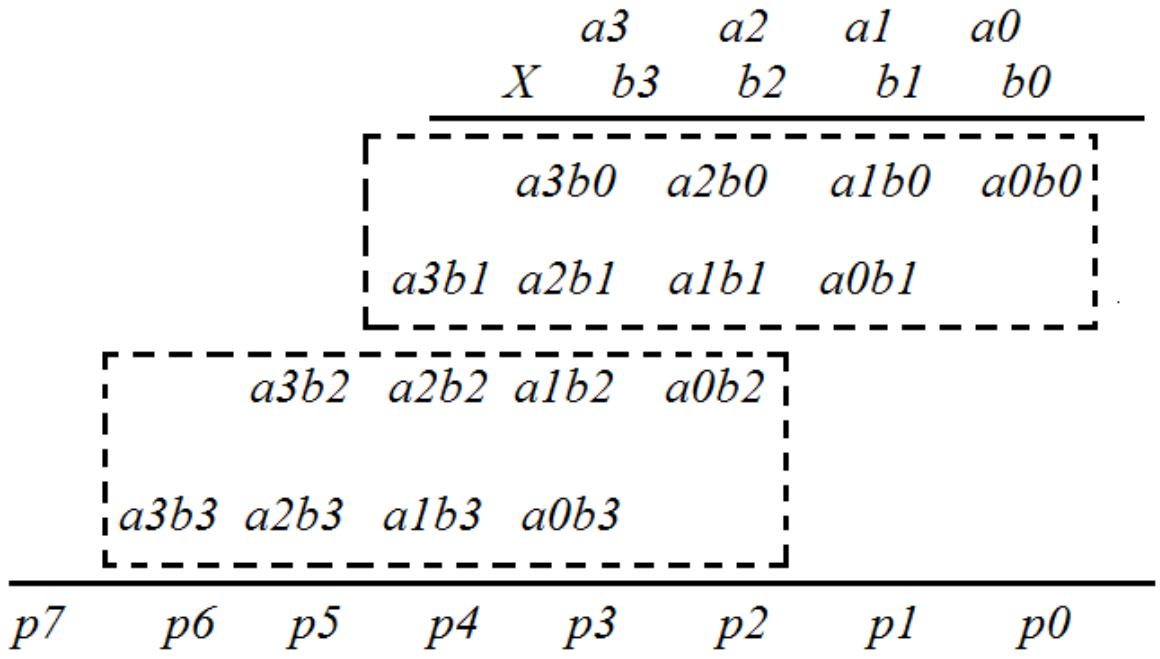


Figure: 3.1 How to multiply [3]

3.2 Stages 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 product generate 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 combinatorial 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 Types of Multipliers

1. Iterative multiplier
2. Array multipliers.

Iterative multiplier can accomplish multiplication through a series of shift and addition operations. Since it can reuse the same hardware to perform multiplication, it occupies less area than other multipliers. However, it needs more clock cycles to accomplish multiplication and cannot be realized in pipeline structure. On the other hand, array multiplier is common in multiplier design due to its regular and compact structure.

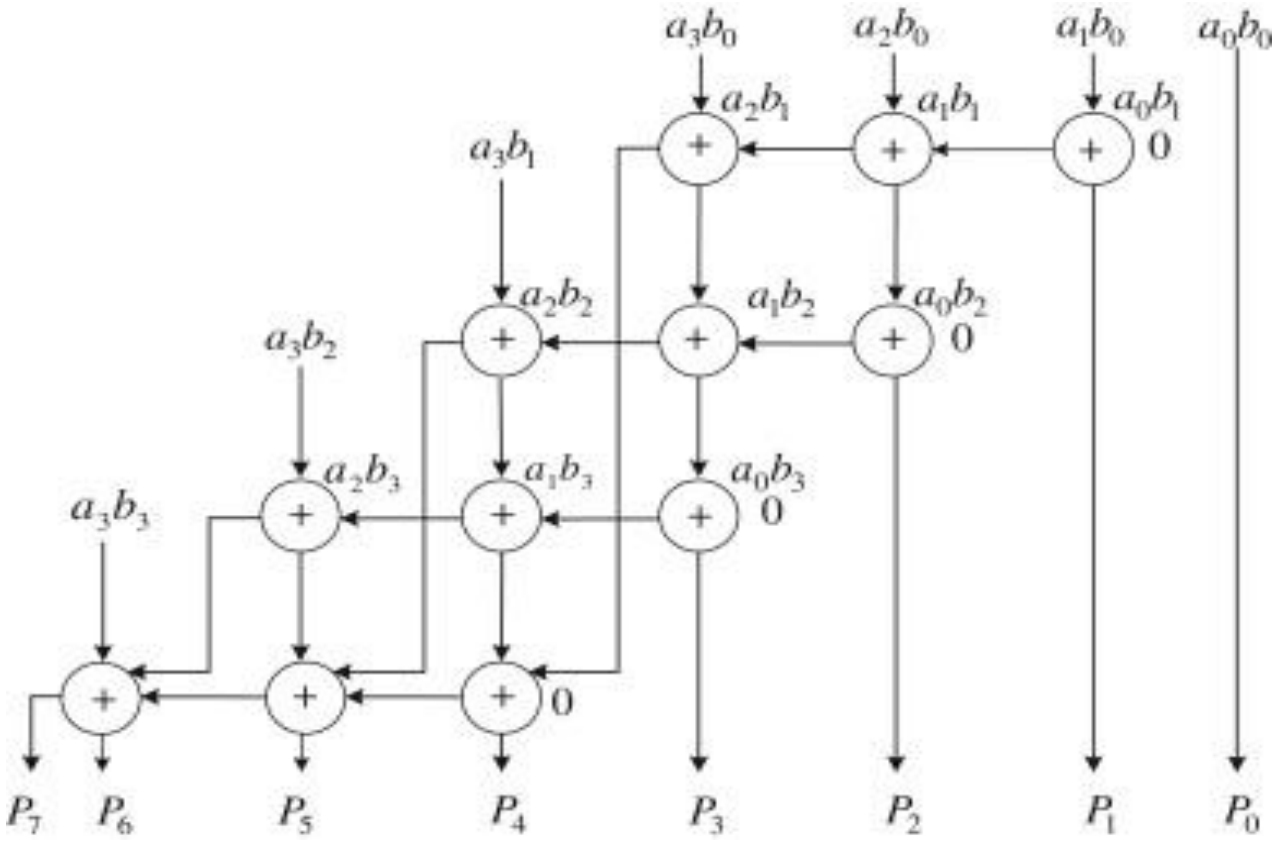
The structure of array multiplier is organized by several stages of adders and and-gates. The advantage of this structure is that the arrangement of its adders is very regular and is favourable for layout due to this advantage. It also can be realized with parallel structure. However it occupies more area and hardware than that of iterative multiplier.

According to the way of carry propagation, array multiplier can be classified into two structures:

1. ripple-carry array (RCA)
2. carry-save array (CSA)

In RCA multiplier all adder cells are composed of ripple carry adders,as in fig 3.2. RCA circuit uses multiple full adders to add N-bit numbers. Each full adder inputs a C_{in}, which is the C_{out} of the previous adder. This adder is called a ripple-carry adder, since each carry bit "ripples" to the next full adder. The first and only the first full adder may be replaced by a half adder. The layout of a ripple-carry adder is simple, which allows for fast design time. However, the ripple-carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder.

For example, it needs 3N adders to accomplish multiplication in an N× N multiplier. However, delay time needed in the worst case is (2N+1) full adder delay.



Figure

3.2.Accumulation of PPs using Ripple Carry Adder [11]

In accumulation using CSA, the main adder cells consist of carry save adders, shown in fig. 3.3. A carry-save adder is a type of digital adder, to compute the sum of three or more n-bit numbers in binary. It differs from other digital adders in that it outputs two numbers of the same dimensions as the inputs, one which is a sequence of partial sum bits and another which is a sequence of carry bits. The carry-save unit consists of n full adders, each of which computes a single sum and carry bit based solely on the corresponding bits of the three input numbers. The entire sum can then be computed by:

1. Shifting the carry sequence sc left by one place.

2. Appending a 0 to the front (most significant bit) of the partial sum sequence ps.

3. Using a ripple carry adder to add these two together and produce the resulting n +1-bit value.

Ripple Carry 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.

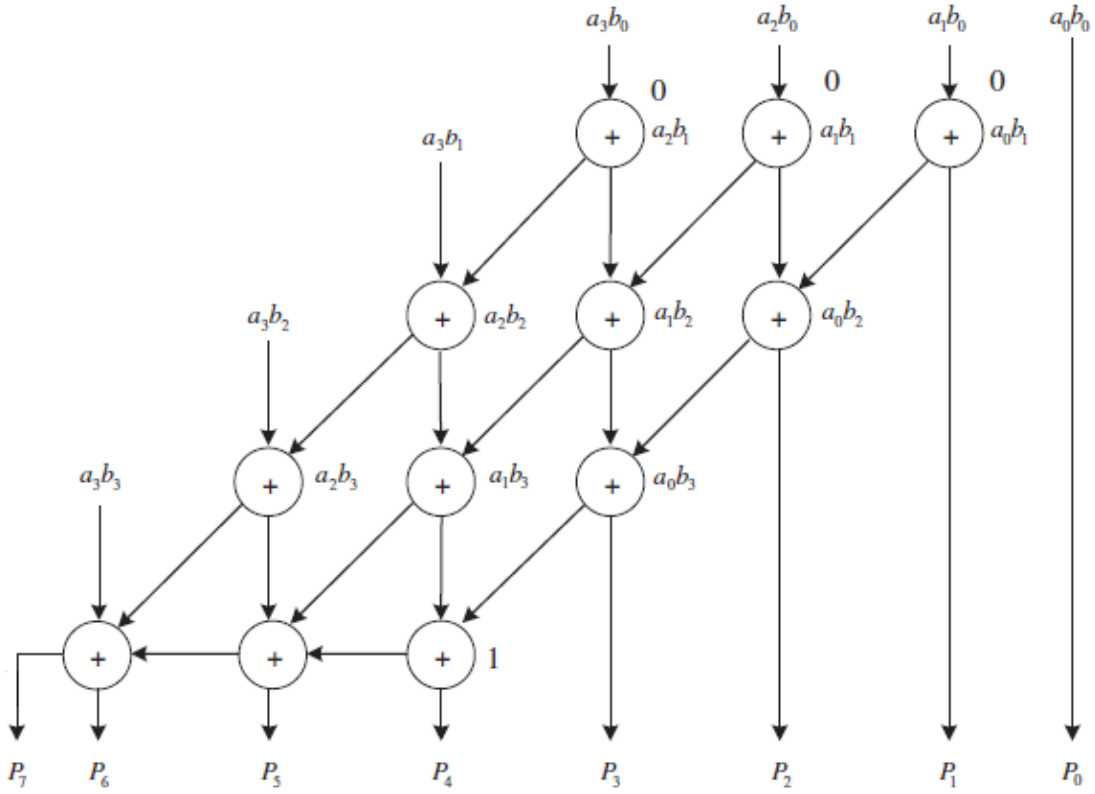


Figure: 3.3 Accumulation of PPs using Carry Save Adder [11]

3.4 Adders

Adders are most commonly used in various electronic applications e.g. Digital signal processing in which adders are used to perform various algorithms like FIR, IIR etc. As portable multimedia and communications applications emerge, the need for low power digital circuits becomes more prominent. Addition process is the most used operation in any DSP because addition is involved in all other mathematical operations. Therefore, adders design is considered critical because it influences the performance of the system in terms of power and delay. Adders construct a major block in any DSP since all the arithmetic operations (subtraction, multiplication and division) rely on addition. However, due to the carry propagation process, adders tend to be slow and consume enormous glitching power. Therefore, adders are considered the bottle neck of any DSP design.

3.4.1 Ripple Carry Adder

Ripple carry adder can be designed by cascading full adder in series i.e. carry from previous full adder is connected as input carry for the next stage. The structure of RCA is shown in Fig 3.5. Full adder is a basic building block of Ripple carry adder. Therefore, to design n-bit parallel adder, it requires n full adders. Operations of a Full Adder are defined by the Boolean equations for the sum and carry signals:

$s_i = a_i \oplus b_i \oplus c_i$

$c_{i+1} = a_i b_i c_i + a_i b_i c_i + a_i b_i c_i + a_i b_i c_i$

where ai, bi, and ci are the inputs to the i-th full adder stage, and s_i and c_{i+1} are the sum and carry outputs from the i-th stage, respectively. From the above equation we realize that the realization of the sum function requires two XOR logic gates. The carry function is further rewritten defining the Carry-Propagate p_i and Carry-Generate g_i terms:

$p_i = a_i \oplus b_i$, $g_i = a_i \bullet b_i$

At a given stage i-th, a carry is generated if g_i is true (i.e., both a_i and b_i are ONEs), and if p_i is true,a stage propagates an input carry to its output (i.e., either a_i or b_i is a ONE). The logical implementation of the full adder is shown in Fig. 3.4.

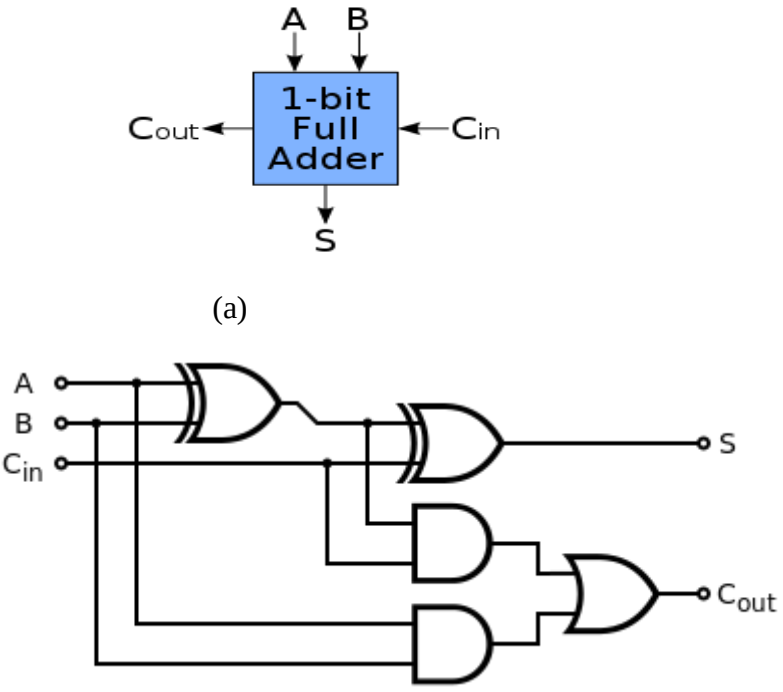


Figure 3.4(a) Block diagram of Full Adder. (b) Gate level schematic of Full Adder.

The path from the input to the output signal that is likely to take the longest time is designated as a "critical path". In the case of a RCA, this is the path from the least significant input x0 or y0 to the last sum bit sn. The major limitation of Ripple carry adder is that as the bit length goes on increases, delay also increases. Therefore, Ripple carry adder is not suitable if large number bits are to be added. The major element that causes delay is carry propagation, therefore it is important to calculate carry delay from input to output. For n-bit Ripple carry adder, Delay for carry can

be calculated as: -

TC = TFA ((x0, y0) to c0) + (n-2) * TFA (cin to cout)) +TFA (cin to sout (n-1)).

Where TFA (input to output) represent the delay of full adder on the path between it's specified input and output.

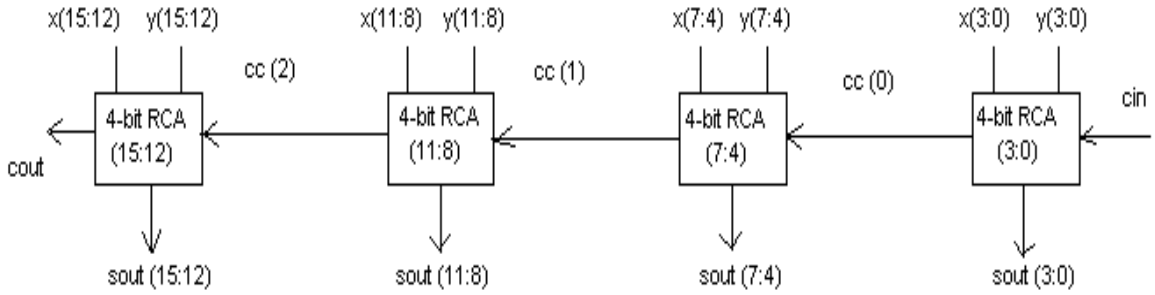


Figure 3.5 Schematic block diagram of 16-bit ripple carry adder.

3.4.2. Carry Save Adder.

Carry save adder is used to compute sum of three or more n-bit binary numbers. Carry save adder is same as a full adder. As shown in figure 3.7, here computation of sum of two 16-bit binary numbers, so we take 16 half adders at first stage instead of using 16 full adders. Therefore, carry save unit consists of 16 half adders, each of which computes single sum and carry bit based only on the corresponding bits of the two input numbers.

When adding together three or more numbers, using a carry-save adder followed by a ripple carry adder is faster than using two ripple carry adders. This is because a ripple carry adder cannot compute a sum bit without waiting for the previous carry bit to be produced, and thus has a delay equal to that of *n* full adders. A carry-save adder, however, produces all of its output values in parallel, and thus has the same delay as a single full-adder. Thus the total computation time (in units of full-adder delay time) for a carry-save adder plus a

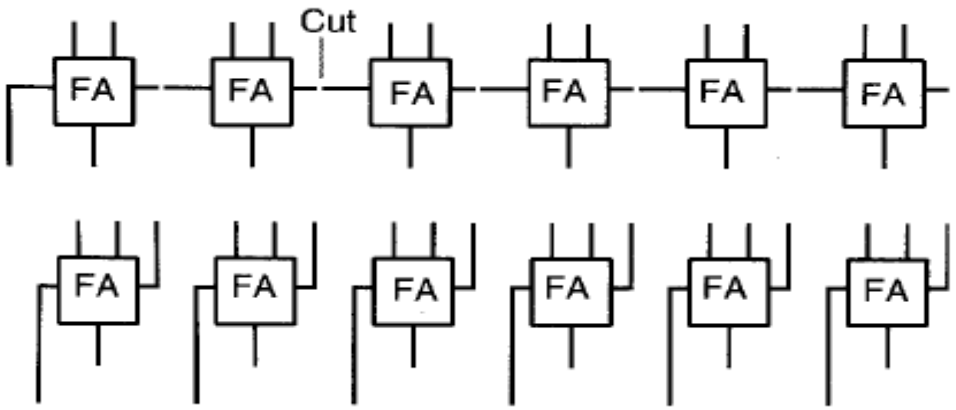


Figure 3.6 A RCA turns into CSA if carries are saved (stored) rather than propagated.

ripple carry adder is $n + 1$, whereas for two ripple carry adders it would be $2n$.

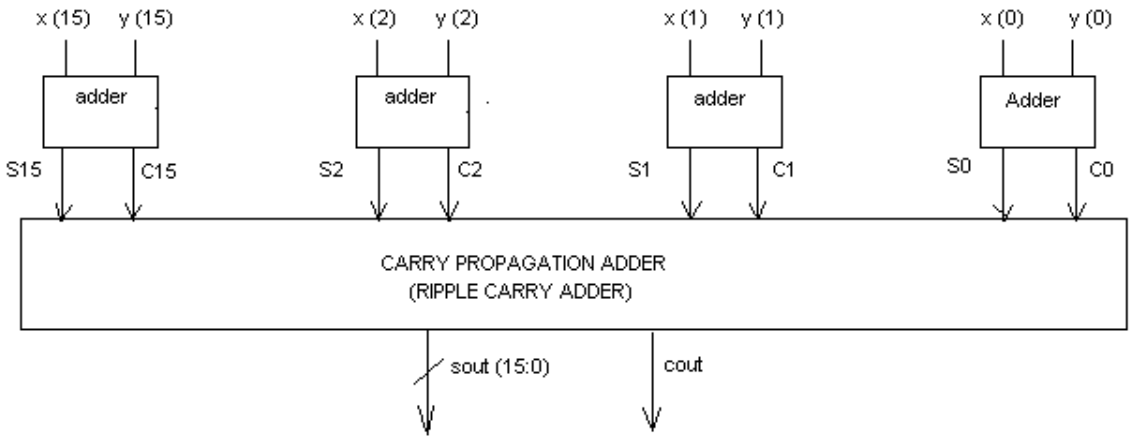


Figure 3.7 Computation flow of CSA

3.4.3 Carry Lookahead Adder

A significant speed improvement in the implementation of a parallel adder was introduced by a Carry-Lookahead-Adder (CLA) developed by Weinberger and Smith in 1958. The CLA adder is theoretically one of the fastest schemes used for the addition of two numbers, since the delay to add two numbers depends on the logarithm of the size of the operands.

The Carry Loookahead Adder uses modified full adders (modified in the sense that a carry output is not formed) for each bit position and Look ahead modules which are used to generate carry signals independently for a group of *k*-bits,shown in Fig 3.9. In most common case *k*=4. in Fig. 3.8. Extending the carry equation to a second stage in a Ripple-Carry-Adder we obtain In addition to carry signal for the group, Look ahead modules produce group carry

generate (G) and group carry propagate (P) outputs that indicate that a carry is generated within the group, or that an incoming carry would propagate across the group, as shown:

$$c_{i+1}=G_i+P_ic_i$$

As opposed to RCA or CSA the critical path in the CLA travels in vertical direction rather than a horizontal one. Therefore the delay of CLA is not directly proportional to the size of the adder N, but to the number of levels used.

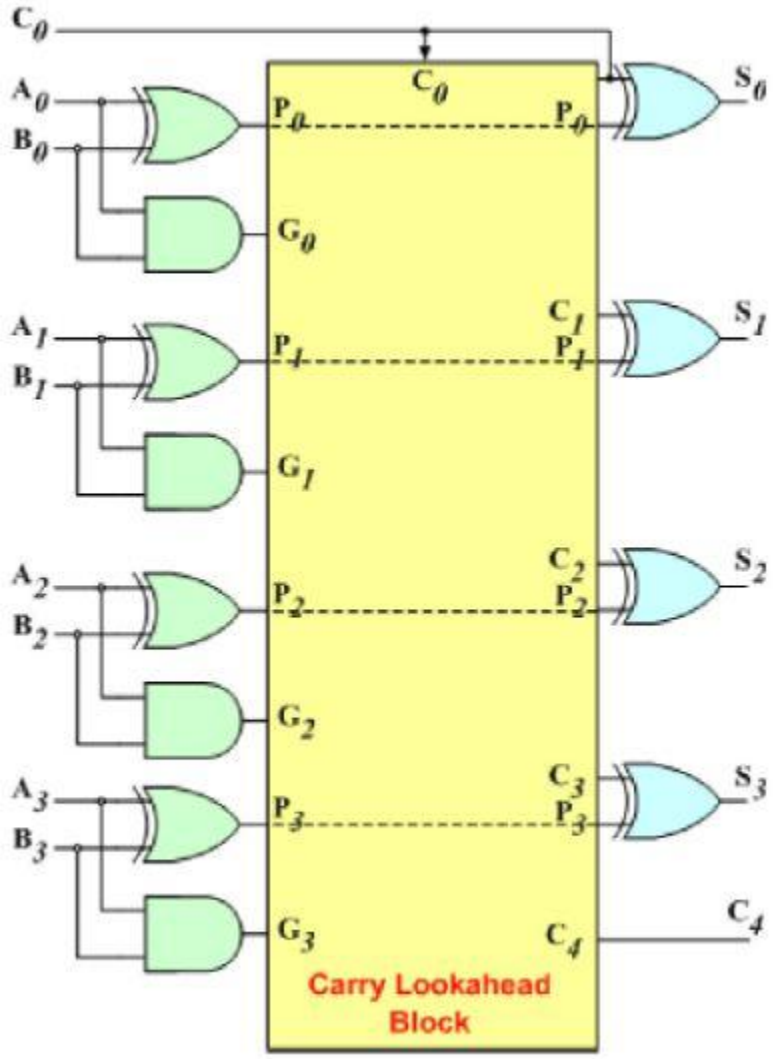


Figure 3.8 Carry lookahead adder with generate and propagate signals.

The latency through this 16-bit adder consists of the time required for:

1. Producing the G and P for individual bit positions (1 gate level).
2. Producing the G and P signals for 4-bit blocks (2 gate levels).
3. Predicting the carry-in signals c4, c8 and c12 for the blocks (2 gate levels).
4. Predicting the internal carries within each 4-bit block (2 gate levels).

5. Computing the sum bits (2 gate levels).

Thus the total latency for 16-bit adder is a 9-gate level that is very less as compared to 16-bit ripple carry adder that is 32 gate levels.

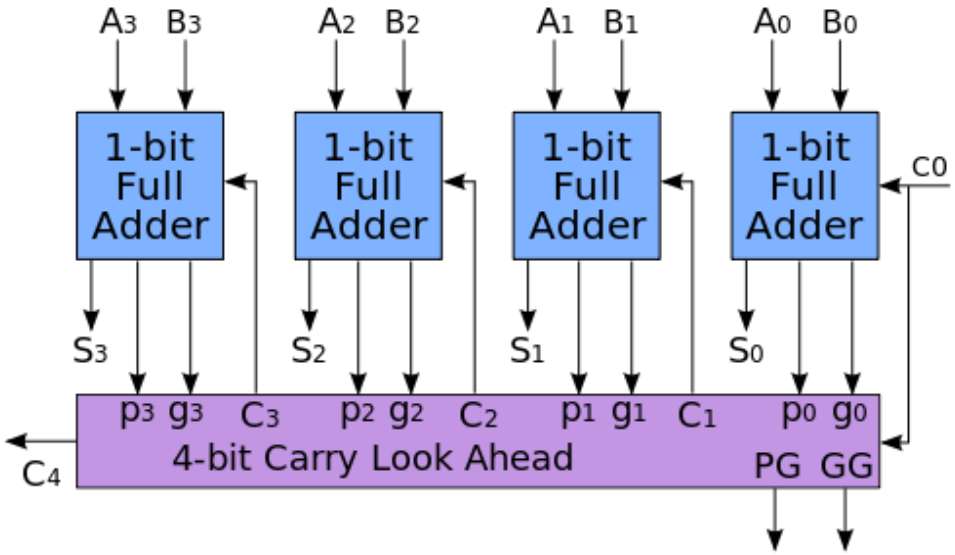


Figure 3.9 Schematic block diagram of 16-bit carry lookahead adder.

3.4.4 Carry Select Adder

The theoretically fastest scheme for addition of two numbers is "Conditional-Sum Addition" (CNSA) proposed by Sklansky in 1960. The essence of the CNSA scheme is in the realization that we can add two numbers without waiting for the carry signal to be available. Simply, the numbers are added in two instances: one assuming Cin = 0 and the other assuming Cin = 1. The conditionally produced results: Sum0, Sum1 and Carry0, Carry1 are selected by a multiplexer using an incoming carry signal Cin as a multiplexer control. Similarly to the CLA the input bits are divided into groups which are in case of CSNA added "conditionally". It is apparent that while building CSNA the hardware complexity starts to grow rapidly starting from the Least Significant bit position. Therefore, in practice, the full-blown implementation of the CNSA is not found. However, the idea of adding the Most Significant (MS) portion of the operands conditionally and selecting the results once the carry-in signal is computed in the Least Significant (LS) portion is attractive. Such a scheme (which is a subset of CNSA) is known as "Carry-Select Adder"(CSLA) .The Carry Select Adder (CSLA) divides the words to be added into blocks and forms two sums for each block in parallel (one with a carry in of zero and the other with a carry in of one)

As shown in an example of a 16 bit carry select adder in Fig. 3.10, the carry-out from

the previous LS 4-bit block controls a multiplexer that selects the appropriate sum from the MS portion. The carry out is computed using the equation for the carry out of the group, since the group propagate signal Pi is the carry out of an adder with a carry input of one and the group generate Gi signal is the carry out of an adder with a carry input of zero. This speeds-up the computation of the carry signal necessary for selection in the next block. The upper 8-bits are computed conditionally using two CSLAs similar to the one used in the LS 8-bit portion. The delay of this adder is determined by the speed of the LS k-bit block (4-bit RCA) and delay of multiplexers in the MS path.

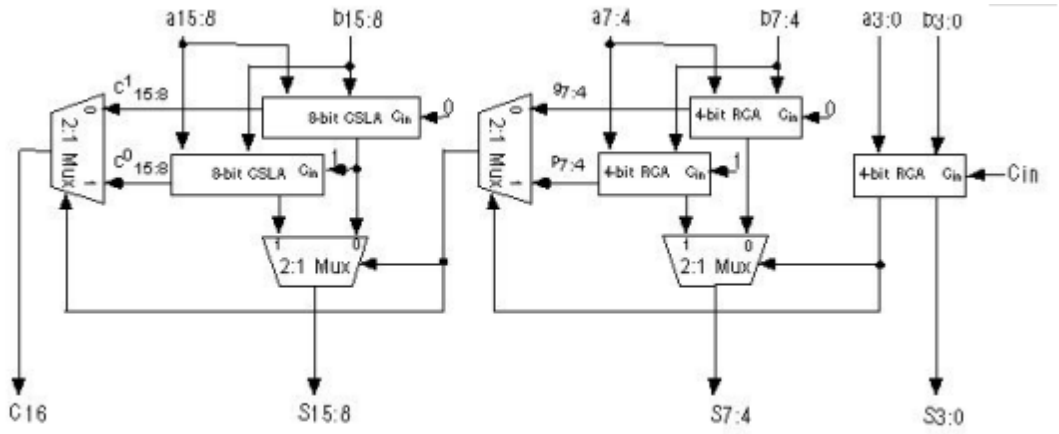


Figure 3.10 Schematic block diagram of 16-bit Carry select adder

CHAPTER 4

BYPASSING MULTIPLIER

4.1 1-Dimensional Multiplier

Many prior digital multipliers were aimed at transition or switch reductions to reduce power dissipation as well. A leapfrog multiplier was proposed in, by using a hardware bypassing approach to avoid the redundant computations by disabling the adder units whose partial product becomes zero.

4.1.1 Row Bypassing Technique

In conventional array multiplier, the summands $P_k = Y_i .X_j$ are generated in parallel with AND gates, and then added in array of 1-bit full adders. In Braun multiplier of m x n requires m x (n-1) adders and n x m gates. The multiplier depth corresponds to the word length n of the operand X.

Traditionally, the transition activity of the multiplier is poorly correlated with X. In particular, if the j-th bit of X is 0,the j-th row of adders does not need to be activated, since the corresponding partial product is 0.The adders of the conventional array multiplier, however, perform summation of the zero partial products and, as result , exhibit redundant signal switching. The increased activity of the internal nodes results in unnecessary power dissipation. To eliminate the redundant signal transitions, proposal to diable the adders whose partial product is zero, while shifting and bypassing the partial product of the previous adder rows to the next row of adders.

Consider a multiplier with multiplier bits Y and multiplicand bits as X. A simple thought to improve the power efficiency was, as soon as Xj was found to be zero, the corresponding partial product (row direction) is automatically reset and bypassed to avoid triggering those adding units in the row. Hence, two MUXs (multiplexer) are required in the adding unit to realize the bypassing operation. If the j-th bit of X is 0, since the corresponding partial product is 0. The adders of the conventional array multiplier, perform summation of the zero partial products and, as result, exhibit redundant signal switching. The increased activity of the internal nodes results in unnecessary power dissipation. To eliminate the redundant signal transitions, disable the adders whose partial product is zero, while shifting and bypassing the partial product of the previous adder rows to the next row of adders.

For a low-power row-bypassing multiplier, the addition operations in the j-th row can be disabled to reduce the power dissipation if the bit X_j in the multiplier is 0, i. e., all partial products X_iY_j , $0 \leq i \leq n-1$, are zero. As a result, the addition operations in the j-th row of CSAs in Fig. 4.2 can be bypassed and the outputs from the (j-1)-th row of CSAs can be directly fed to the (j+1)-th row of CSAs without affecting the multiplication result.

In the multiplier design, each MFA as shown in Fig 4.1 in the CSA array is attached by three tri-state buffers and two 2-to-1 multiplexers. Two multiplexers augmented to the outputs of the full-adder transmit the carry-input and the sum of the previous addition to the outputs, when the corresponding bit X_j is zero. The tri-state buffers, placed at the inputs of the adder cell, disable signal transitions in those adding cells which are bypassed. The output carry-bits (c) are passed downwards, instead of to the right. Because the addition operations of the rightmost FAs in the CSA rows are able to be bypassed, the extra correcting circuits must be added to correct the multiplication result.

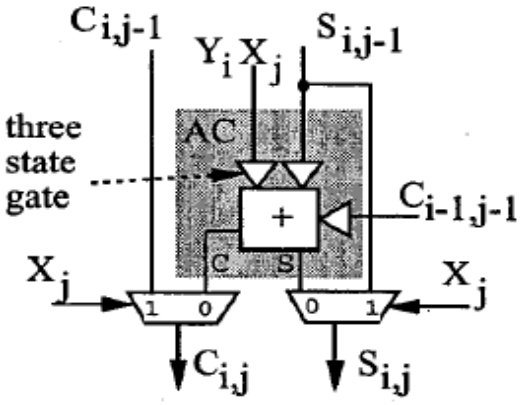


Fig 4.1 MFA in row bypassing [16]

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.

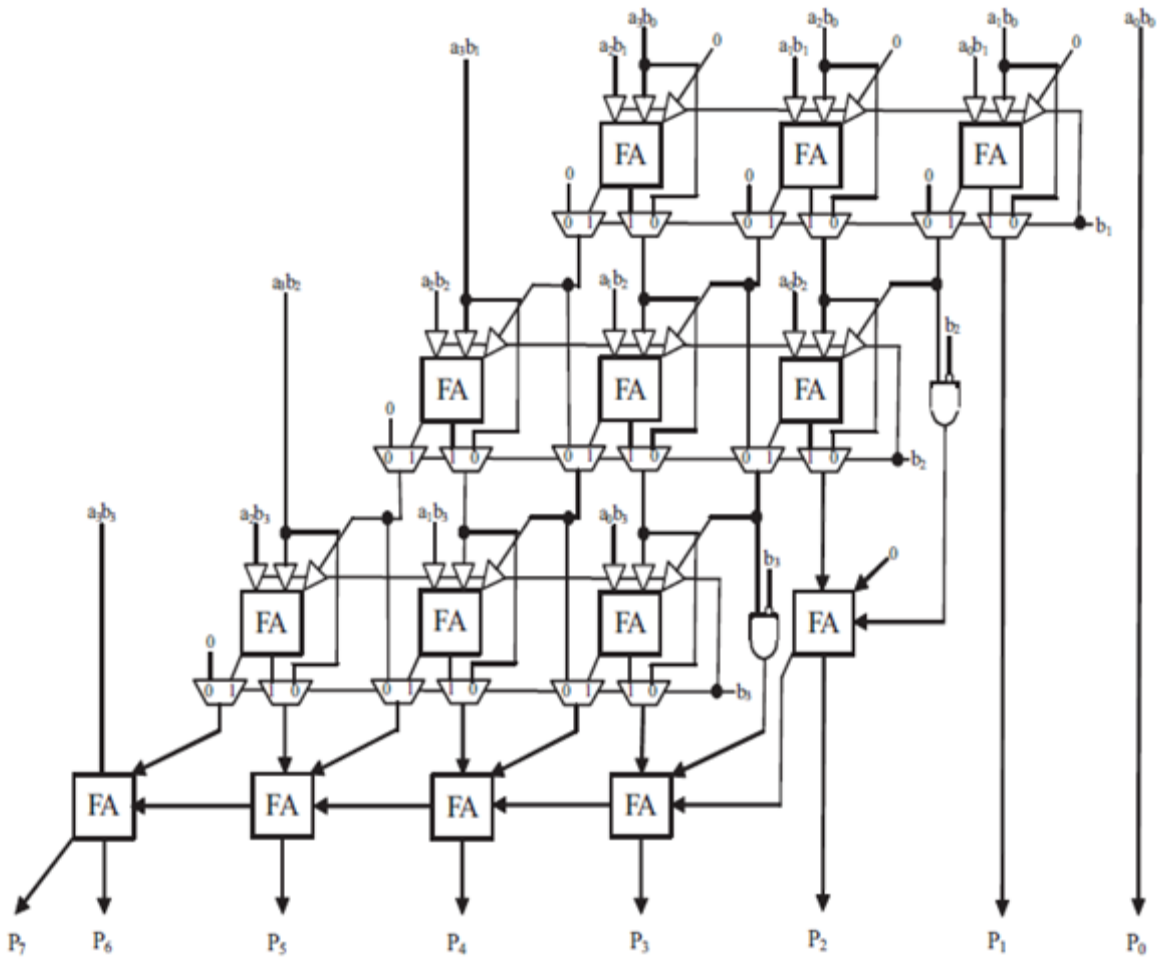


Fig 4.2 4x4 Braun multiplier using row bypassing [3]

4.1.2 Column Bypassing Technique

One of the techniques of bypassing rows of full adders, a multiplier design in which columns of adders are bypassed. In this approach, the operations in a column can be disabled if the corresponding bit in the multiplicand is 0. The tructure for column bypassimg multiplier is depicted in Fig 4.4. There are advantages to this approach, over the row bypassing multiplier.

. Consider a multiplier with multiplicand bits as b and multiplier bits as a. For a low-power column-bypassing multiplier, the addition operations in the (i+1)-th column can be bypassed if the bit a_i, in the multiplicand is 0, i. e., all partial products a_ib_j, $0 < j < n-1$, are zero. In the multiplier design, the MFA is simpler than that in the row bypassing multiplier is shown on Fig 4.3. Each modified FA in the CSA array is only attached by two tri-state buffers and one 2-to-1 multiplexer. As the bit, a_i, in the multiplicand is 0, their inputs in the (i+1)-th column will be disabled and the carry output in the column must be set to be 0 to produce the correct output. Hence, the protecting process can be done by adding an and gate

at the outputs of the last row of CSAs. However, the multiplier design does not need the extra correcting circuit as illustrated in row bypassing.

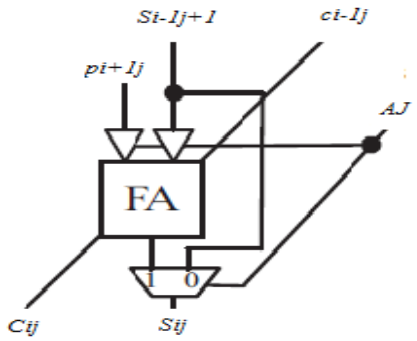


Fig.4.3 MFA for column bypassing [17]

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.

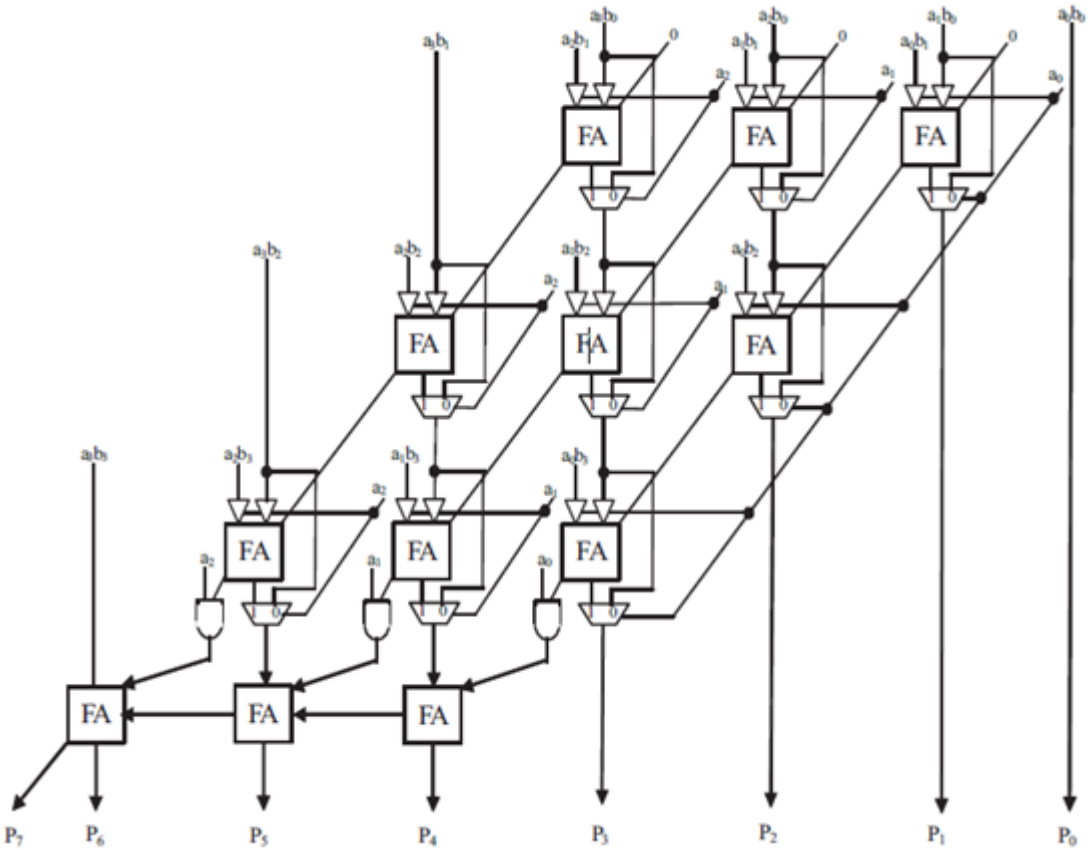


Fig 4.4 4x4 Braun multiplier using column bypassing [3]

Assume the execution of 1010x1111 as shown in fig 4.5. It can be verified that, for FAs in the first and third diagonals, two out of the three input bits are 0: the 'carry' bit from its upper right FA, and the partial product a_ib_j. As a result, the output carry bit of such an FA is 0, and the output sum bit is simply equal to the third bit, which is the 'sum' output of its upper FA. Therefore, when a_i is 0, the operations in the corresponding diagonal can be disabled since all the outputs are known. We refer to the FAs in a diagonal in fig 4.5 as a column. Let FAi, j be the full adders locating in row i and column j, 0< i, j< n-2, in the (n-1)x(n-1)array. $FA_{0,0}$ is the adder at the upper-right corner. The following theorem establishes reason for column bypassing.

Theorem 1: When a_j=0, the output of a column j adder cell FAi, j can be specified as follows.

1. The output carry bit is 0.
2. The output sum bit is equal to the output sum bit of $Fa_{i-1, j+1}$.

Proof: We prove this theorem by induction.

1. Consider row 0. Note that, in row 0, there are only two bits to be added. Adder $FA_{0, j}$ carries out $a_jb_1+a_{j+1}b_0$. If a_j=0, then the output carry bit must be zero, and the out sum bit is equal to $a_{j+1}b_0$.
2. Assume that the theorem holds for row i.
3. In row i+1, the inputs of $FA_{i+1, j}$ are carry bit from $FA_{i, j}$, sum bit from $FA_{i, j+1}$, and the partial product a_jb_{i+1}. Since a_j=0, two out of the three inputs are 0, and the output sum bit is equal to the sum bit sent by $FA_{i, j+1.}$

According to theorem 1, when aj=0, the operations in column j can be ignored and thus the full adders can be disabled since the outputs are known.

4.2 2-dimensional bypassing

Many prior digital multipliers were aimed at transition or switch reductions to reduce power dissipation as well. A leapfrog multiplier was proposed in, by using a hardware bypassing approach to avoid the redundant computations by disabling the adder units whose partial product becomes zero. Another power saving approach is to skip the computation caused by the sign extension bits which are located at the left side of operands. Another technique was close to a "bypassing" multiplier which skips the addition when the partial product of a row is zero. One of the power-saving strategies by grouping the operands with the same sign and then computing them separately to avoid unnecessary transitions. All of these prior methods

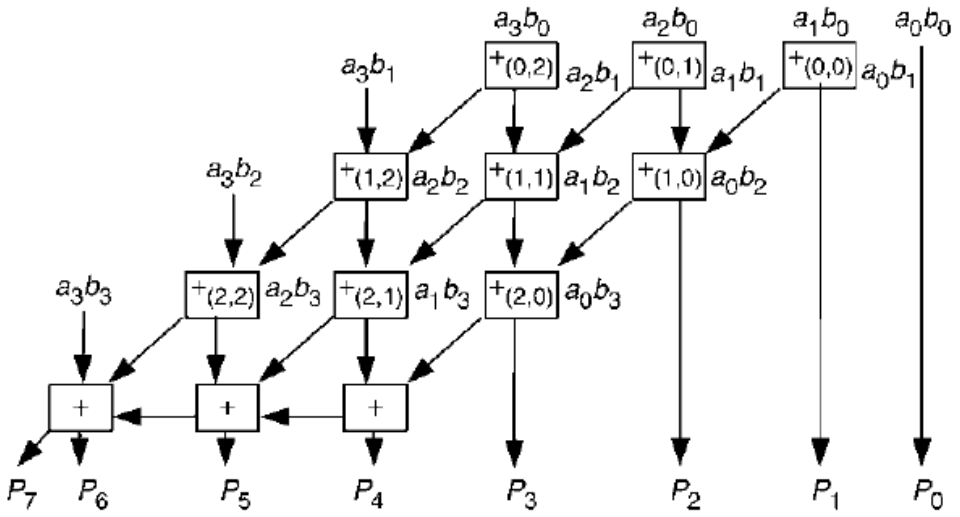


Fig 4.5 Example of column bypassing.1010x1111 [17]

depend on certain decision logic given that a partial product is zero to either skip or shut down adding cells in a row-based manner. In other words, all the prior works utilized a one-dimensional bypassing approach basically. Thus, a 2-dimensional bypassing approach which 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.

4.2.1 Problem in 2-dimensional bypassing

Consider a multiplier with multiplier bits as X and the multiplicand bits as Y. For a low-power 2-dimensional bypassing multiplier, it is desired that the addition operations in the (i+1)-th column or the j-th row can be bypassed for the power reduction if the bit Yj , in the multiplicand is 0 or the bit, Xj, in the multiplier is 0. In other words, as soon as the Yj is found to be zero, the results from the adding units residing in the previous column are automatically passed to the corresponding adding units in the next column .To correct the carry propagation in the multiplication result, the addition operations in the (i+1)-th column or the j-th row cannot be bypassed if the bit, Yi, is 0, the bit, Xj, is 0, and the carry bit, Ci,j-1, is 1. Hence, the bypass logics are added into the necessary FA to form a correct adder cell(AC).However, a conflict appears when one adding cell, ACij , encounters a scenario that Xi =Yj = 0.

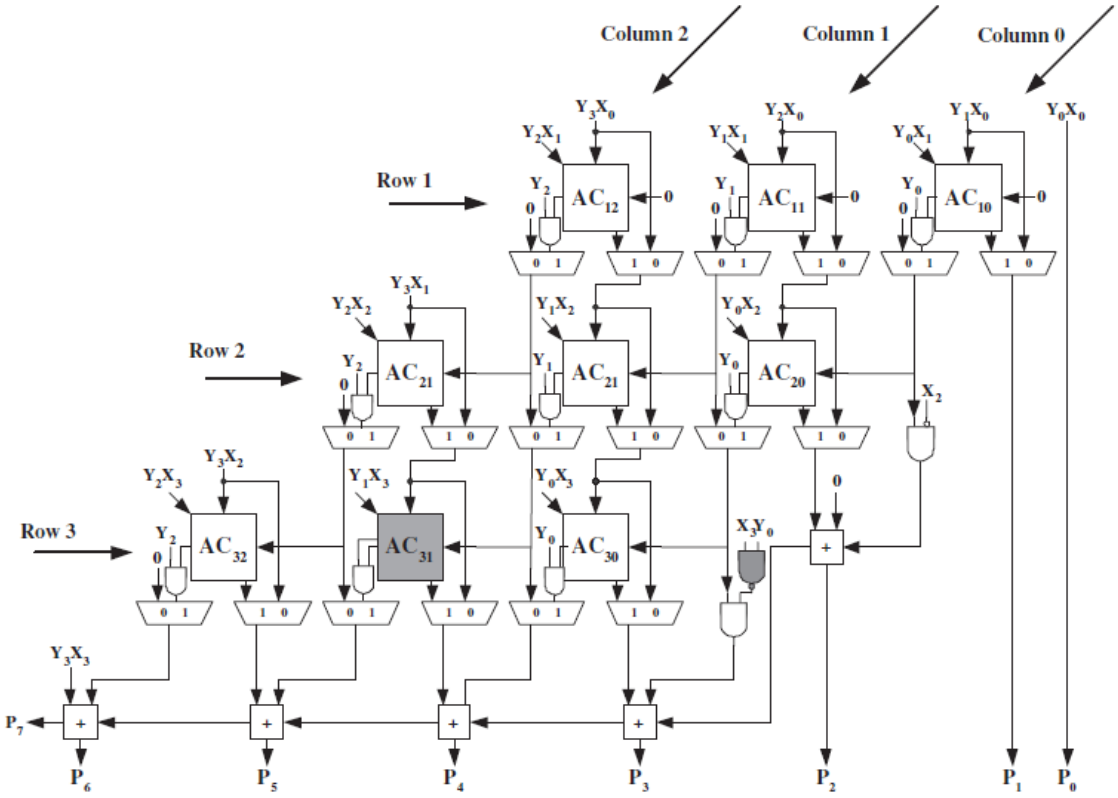


Fig.4.6 4x4 Braun multiplier using row and column bypassing [15]

4.2.2 Solution for 2-dimensional bypassing

For instance, assume i = 2, j = 1 and X2 = Y1 =0 in Fig. 4.6 which shows a 2-dimensional bypassing 4×4 multiplier design. Then, we expect the row 2 and the column 1 are bypassed if we directly apply the prior 1-dimensional bypassing method. If the carry out of the adding cell AC12 is "1", it should be propagated to the carry in of AC31 and then its carry out. However, the carry bit will be lost if AC31 is bypassed due to Y1 = 0. Consequently, an error is occurred, since the carry out of AC31 will be zero. Another instance occurs conflict when X2 = Y0 = 0, the adding cell AC30 will miss the carry in because the column 0 is bypassed. Thus, propose to include a bypass logic (BL) in certain adding cells. A simple rule is introduced: If and only if Xi is not equal to "0" and the carry in is "1", then the adding cell, ACij , cannot be bypassed**.** Hence, an adding cell with the bypass logic is proposed. It is also represented as in Fig 4.7.

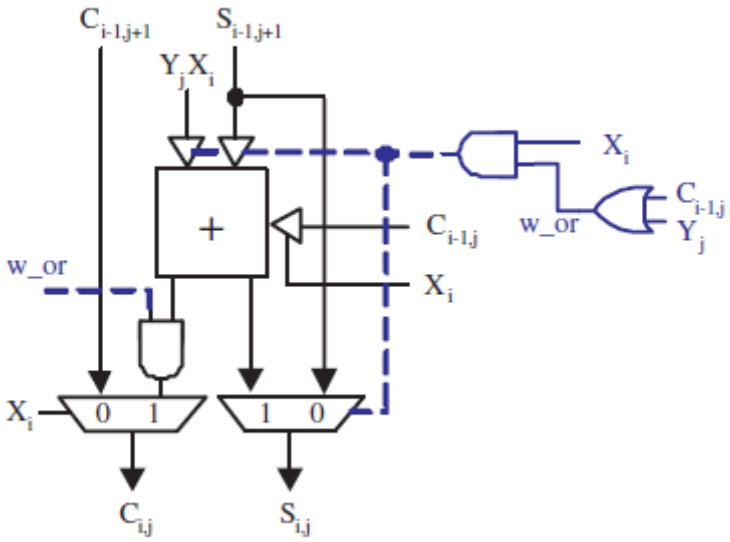


Fig 4.7 AC with bypass logic

Others AC_{ij}s are shown as in Fig 4.8.

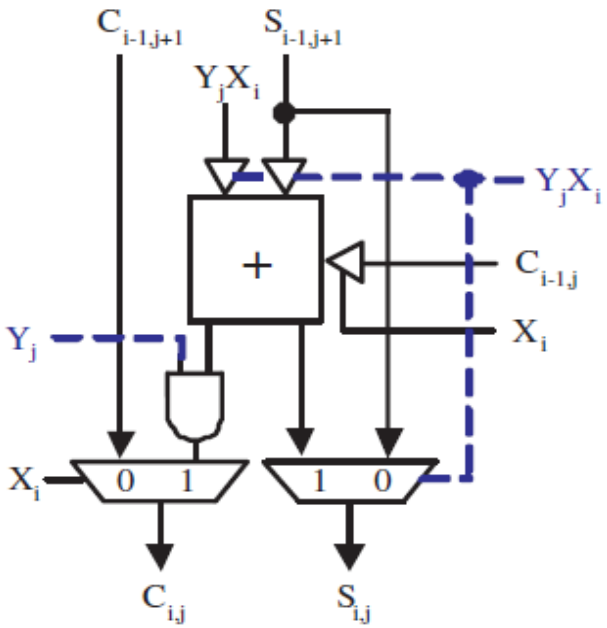


Fig: 4.8 AC without bypass logic.

In order to save more bypass logic area, the adding cell AC30 can be further simplified such that a single nand gate is used to replace the adding cell with bypass logic. It still provides a correct operation when X2 = Y0 = 0 and carry out of AC11 is "1". It is obvious that not every adding cell needs the bypass logic. For instance, those adding cells in charge of the

calculation of LSBs of X and Y . It will be very area-efficient if we can identify which adding cells require the bypass logic to produce a correct multiplication result. Given n = 4, it can be easily concluded that AC31 is the only unit with the necessity of a bypass logic. If n = 5 and the identical array structure is used, then AC31, AC32,AC41,AC42 need the bypass logic to attain correct results. By a similar induction, for any n×n multipliers, where $n \geq 4$, all of the adding cells, ACij, where $n - 1 \geq i \geq 3$ and $n - 3 \geq j \geq 1$, must contain the bypass logic to execute the correct multiplication. In other words, when n = 4, there is only one adding cell which must contain the bypass logic. If n = 5, then the 5 × 5 multiplier has a total of $(5 - 3) \times (5 - 3) = 4$ adding cells with bypass logic. If n = 8, a total of $(8 - 3) \times (8 - 3) = 25$ adding cells with bypass logic are required. In short, the number of the required adding cells with bypass logic is as follows.

$1 = (4 - 3) \times (4 - 3)$, n= 4

$4 = (5 - 3) \times (5 - 3)$, n= 5

$9 = (6 - 3) \times (6 - 3)$, n= 6

...

Therefore, the following rule is concluded.

A total of $(n - 3)^2$ adding cells with bypass logic are required to constitute a 2-dimensional bypassing multiplier, $\forall n > 3$.

Notably, if an earlier adding cell with the bypass logic is set to be activated, all of the following adding cells in the same column must be activated, too. Otherwise, a carry generated in the earlier adding cell will be lost in the bypassing chain.

For instance, if the adding cell AC32 is activated, then the following adding cell, AC42, must be activated to ensure a carry (=1) is propagated correctly from the carry in of AC32 to the carry out of AC42 and even further.

CHAPTER 5
SIMULATION AND RESULTS

5.1 Simulation

After implementing the design using Verilog HDL.The verification of their functional correctness is done through behavioral simulation in Xilinx ISE 8.2 i.The simulation results for 4x4 2-dimensional bypassing multipler i.e row and column bypassing is shown in fig 5.1.

Fig: 5.1

Row and column bypassing multiplier.

5.2 Results

The design of standard Braun multiplier, Row bypassing, Column bypassing and Row and Column Bypassing multiplier for 4×4 ,8x8, 16x16 are simulated and synthesized using Verilog HDL. Table 3 shows comparison of different designs for maximum combinational delay**,** implemented in Xilinx ISE 8.2i targeting Spartan – 3E (xc3s500e-4fg320) FPGA . Using Design Vision Synopsys, the designs are synthesized for dynamic power and area as listed in Table 1 and 2.

Area (no. of Nand gates)	4x4	8x8	16x16
Braun Multiplier	1853.006	7058.923	29679.35
Row Bypassing	2012.371	9140.039	39106.871
Column Bypassing	2143.612	8986.924	37091.375
2-D Bypassing	1993.622	8618.198	34960.262

Table 1.Comparison of Area of different multipliers for 4x4, 8x8, and 16x16.

Dynamic Power(mW)	4x4	8x8	16x16
Row Bypassing	0.1877	0.7765	3.2918
Column Bypassing	0.2089	0.9781	4.6604
2-D Bypassing	0.1885	0.7312	3.0054

Table 2.Comparison of Dynamic Power dissipation for different multipliers, for 4x4, 8x8, and 16x16.

Delay(ns)	4x4	8x8	16x16
Braun Multiplier	11.596	20.522	38.057
Row Bypassing	12.88	23.252	46.073
Column Bypassing	9.364	19.231	36.875
Row & Column Bypassing	13.036	25.646	47.267

Table 3. Comparison of maximum combinational delay of different multipliers for 4x4, 8x8, and 16x16 on Spartan – 3E (xc3s500e-4fg320).

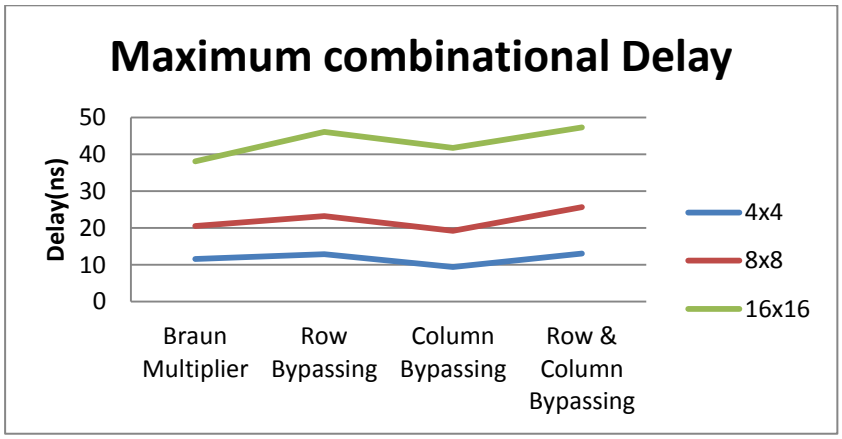


Figure 5.2. The comparison of combinational path delay of braun, row bypassing, column bypassing and row and column bypassing for 4x4, 8x8 , and 16x16.

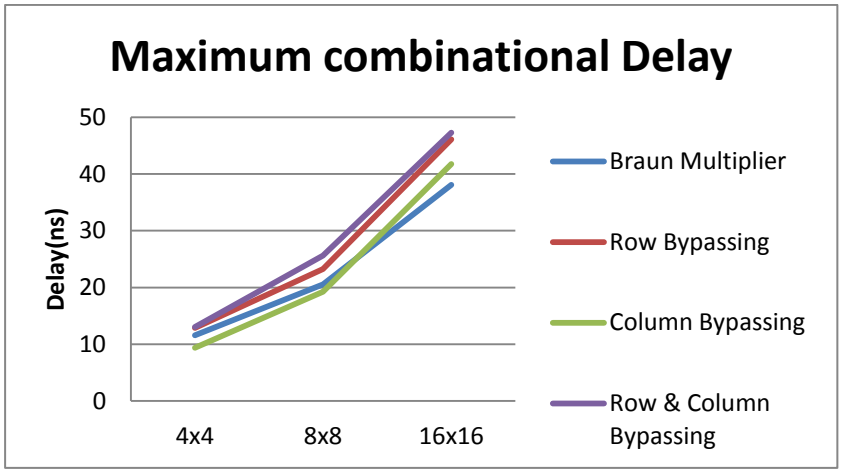
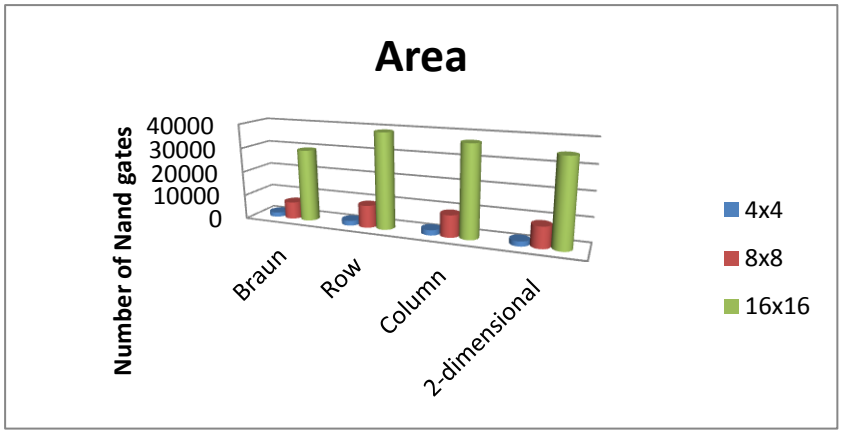


Figure 5.3. The combinational delay comparison of 4x4, 8x8, 16x16 for Braun multiplier, row bypassing, column bypassing and row and column bypassing.



– Figure 5.4 Area comparison of Braun multiplier,Row bypassing, Column bypassing and Row and Column bypassing for 4x4, 8x8, and 16x16.

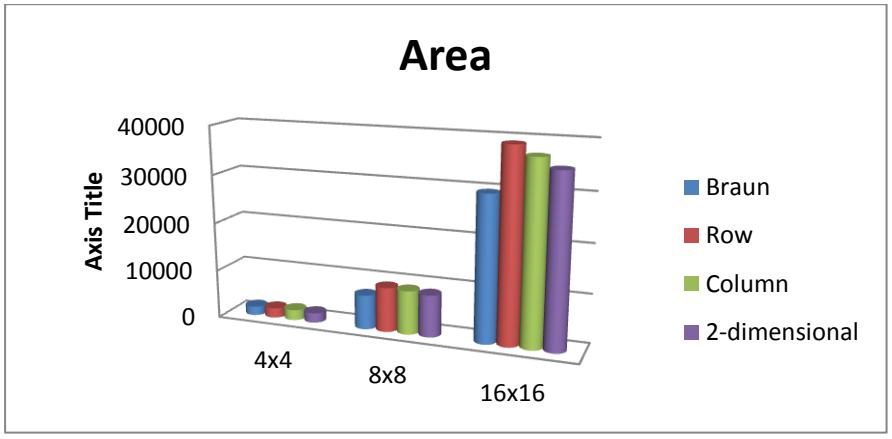
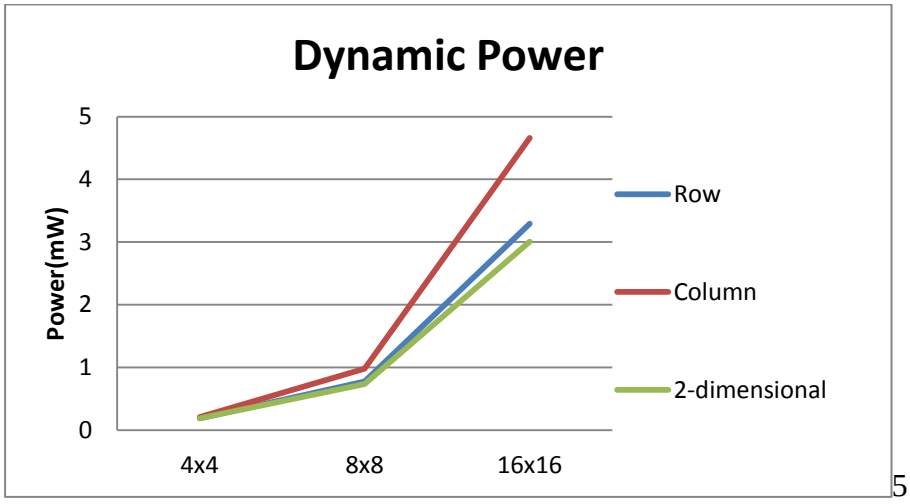


Figure 5.5. Area comparison for 4x4,8x8, 16x16 of Braun multiplier ,Row bypassing, Column bypassing and Row and Column bypassing.



5

Figure 5.6. The dynamic power of 4x4, 8x8, 16x16 for braun , row bypassing, column bypassing and row and column bypassing

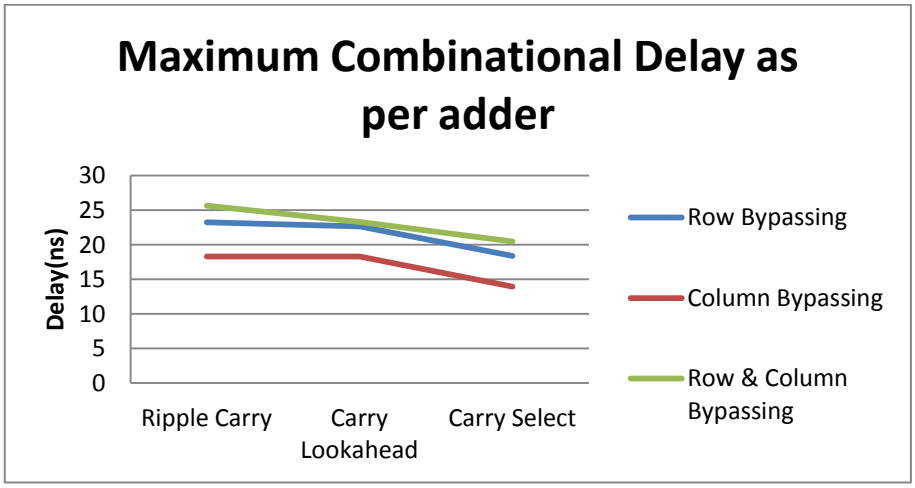


Figure 5.7. The dynamic power of 8x8 of row bypassing, column bypassing and row and column bypassing,for different adders.

Delay(ns)	Ripple Carry	Carry Lookahead	Carry Select
Row Bypassing	23.252	22.618	18.374
Column Bypassing	18.267	18.267	13.951
Row & Column Bypassing	25.646	23.268	20.467

Table 4. Comparison of maximum combinational delay of different multipliers of 8x8 for Ripple Carry, Carry lookahead, and Carry Select adder on Spartan – 3E (xc3s500e-4fg320).

Power(mW)	Carry Look Ahead Adder	Carry Select Adder	Ripple Carry Adder
Row Bypassing	0.75657	0.8977972	0.7749969
Column Bypassing	0.82588	0.9174367	0.8065
2-dimensional Bypassing	0.80609	0.8781577	0.7312136

Table 5. Comparison of dynamic power of different multipliers of 8x8 for Ripple Carry,Carry lookahead, and Carry Select adder on Spartan – 3E (xc3s500e-4fg320).

Area (Nand gate)	Ripple Carry Adder	Carry LookAhead Adder	Carry Select Adder
Row Bypassing	6405.84	8246.347	9056.834
Column Bypassing	7252.66	8196.35	9140.04
2-dimensional Bypassing	6589.809	6880.81	8986.925

Table 6. Comparison of Area of different multipliers of 8x8 for Ripple Carry, Carry lookahead, and Carry Select adder on Spartan – 3E (xc3s500e-4fg320).

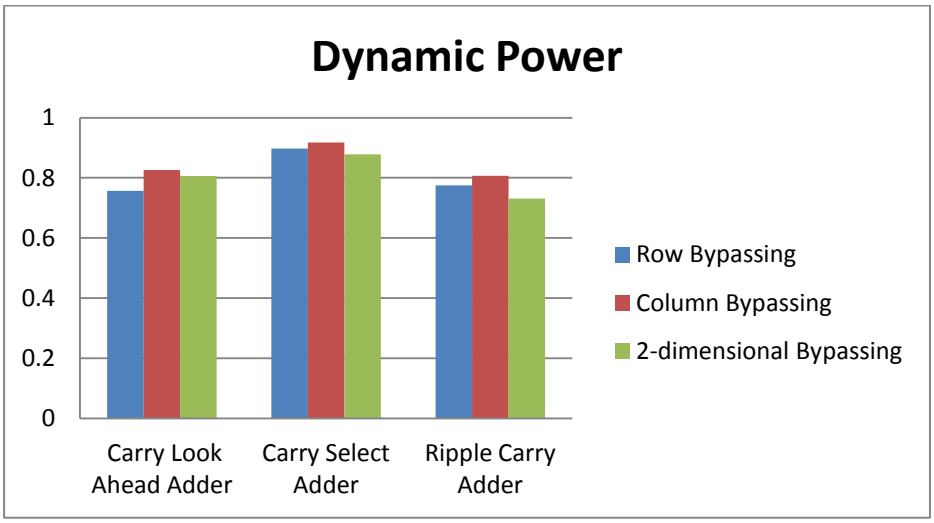


Figure 5.8. The dynamic power of 8x8 of row bypassing, column bypassing and row and column bypassing,for different adders.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

After putting lot of hard efforts, we learnt importance of the multiplier and the usage of bypassing techniques in the multiplier. The multiplier is the most important block in processors. Thus this project Design of bypassing multiplier , helps to improve the efficiency of the multiplier through bypassing techniques. There are three bypassing techniques:

1. Row Bypassing

2. Column Bypassing

3. Row and Column Bypassing

The Row bypassing technique implements bypassing if the multiplicand bit makes the partial product zero. It is better in terms of power as compared to the Braun multiplier. The row bypassing technique results in usage of the extra circuitry, to avoid the problem of carry propagation. Thus, an improvement is Column bypassing multiplier. This technique uses less area and power, as the MFA of the column bypassing multiplier uses only one multiplexer, whereas the row bypassing technique uses two multiplexer. To further improve the efficiency of the multiplier, and reduce the switching activity, Row and Column bypassing technique is implemented. If the multiplicand or multiplier bit is zero, full adder is not implementing. Thus, this results in less power consumption with respect to one dimensional bypassing. On comparing,the results show highest speed for column bypassing when compared to Braun multiplier and bypassing multipliers. For area and dynamic power, two dimensional bypassing consumes least power and area as compared to row bypassing and column bypassing. Thus, we can conclude column bypassing gives least delay and two dimensional bypassing technique has lowest dynamic power consumption. The use of different adders like carry lookahead adder (CLA), carry select adder apart from ripple carry adder(RCA) in the last stage helped to improve in terms of delay.

6.2 Future Scope

The present work on new multiplier architecture can be extended in various directions. Based on the initial study, and the understanding established, the following suggestions are proposed suggestions:

- To implement the row and column bypassing technique on different tree algorithms.
- To focus on both, generation as well as accumulation stage.

REFERENCES

1. 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 2009)* , June2005, PP. .345-349.
2. T. 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),* 2011 ,Vol.3 , PP. 431 - 435.
3. J.T. Yan and Z.W Chen, "Low-Power Multiplier Design with Row and Column Bypassing," in *Proc. IEEE SOC Conference (SOCC 2009*), June 2002, PP.1227-230
4. S. Hong, T. Roh, and H.J Yoo "A 145uW 8×8 Parallel Multiplier based on Optimized Bypassing Architecture", in *Proc. IEEE International Symposium on Circuits and System(ICSAS),* 2011, Vol:55,October1998, PP. 1253-1258.
5. J. Selvakumar, 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 2011),* June 2011, PP. 260-263.
6. Ugurdag , F.Keskin.O,Tunc C,Temizkan.F,Fici,G,Dedeoglu, "RoCoCo: Row and Column Compression for high-performance multiplication on FPGAs"in *Proc. 2011 9th East - West Design and Test Symposium,(EWDTS)* ,2011,PP. 839-844.
7. S.Hong, S.Kim, M. C. Papaefthymiou and W. E. Stark "Low Power Multiplier Design Using Coefficient Multiplication", in *Proc. Twelfth Annual IEEE International ASIC/SOC Conference,*1999, Vol.33,19th February 2009,PP.341-343.
8. B .Paul,,C Fujita.S, M.Okajima" ROM-based 16 times 16 multiplier for low-power applications."IEEE Journal of Solid-State Circuits,2009,Vol.44, PP.902-908.
9. 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)* ,2002,Vol. 1,PP.1985-1997.
10. L.I.D Feng, D.A.I Guoding, Z. Yiqi "Low power DCT architecture based on scalers sharing multiplier" in *Proc. IEEE 7th International Conference on Solid-State and Integrated Circuits Technology*,2004, Vol 3,PP.293-296.
11. K.C Kuo, C.W Chou, "Low power and high speed multiplier design with row bypassing and parallel architecture", Microelectronics Journal 41, 2010, pp. 639–650

12. M.M.Dastierdi, A. A.Kusha, M.Pedram "A low-power structure called bypass zero, feed A directly (BZ-FAD)" *in Proc. IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* 2005 , vol. 17, pp.302-306.
13. R.Modugu, Y.B. Kim and M.Choi "A efficient hardware architecture of modulo 2n + 1 multiplier",IEEE *Trans on Circuits and Systems 1:Fundamental Theory and Applications,* 2003,Vol. 50,PP. 1296-1303 .
14. A.R Vandana. A "novel VLSI SP oriented architecture for implementation of serial parallel Multipliers (SPM)". *in Proc International Conference on Devices, Circuits and Systems (ICDCS),* 2012, vol 56, pp 451-455.
15. G.N.Sung, Y.J.Ciou, C.C.Wang, "A power aware 2-dimensional bypassing multiplier using cell – based design flow", *in Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2008),* May 2008, pp. 3338 – 3341.
16. J. Ohban, V. G. Moshnyaga, K. Inoue, "Multiplier energy reduction through Bypassing of partial products", *in Proc. IEEE Asia-Pacific Conference on Circuits and Systems(APCCAS'02),* 2002, vol.2, pp.13-17
17. M. C. Wen , S. J. Wang and Y. M. Lin, "Low power parallel multiplier with column bypassing", *in Proc. IEEE International Symposium on Circuits and Systems, (ISCAS 2005),* 2005, vol.2, pp.1638-1641.
18. B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs, Oxford University Press, 2000.
19. J.T.Yan and Z.W.Chen, "Low-cost low-power bypassing-based multiplier design*", in Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2010),* 2010, pp.2338-2341.
20. R.P.P. Singh, P. Kumar and B.Singh," Performance Analysis of Fast Adders Using VHDL*", in Proc. International Conference on Advances in Recent Technologies in Communication and Computing,* 2009, PP. 189 – 193.