Low Power Partial Product Reduction Stage

for Booth Multiplier

By

Mahmoud Jawharji

A thesis presented in partial fulfillment of the requirements for the
degree of

Master of Science

in

Division of Engineering Programs

State University of New York
New Paltz, New York 12561

Advisor: Prof. Damu Radhakrishnan

May 2017
Notice: Signature Page Not Included

This thesis has been signed and approved by the appropriate parties.

The signature page has been removed from this digital version for privacy reasons.

SUNY New Paltz theses are copyright protected and may be used for education or personal research only. Reproduction or distribution in part or whole is prohibited without written permission from the author.
Acknowledgements

I am grateful to my advisor, Dr. Damu Radhakrishnan, who inspired me a lot with his knowledge and commitments. His gaudiness was very important for developing this research, which cannot be done without him, and I appreciate for him all the help and knowledge I have gained from him. In addition, I want to thank Dr. Baback Izadi, who helped me a lot during my years of study, and Dr. Chirakkal Easwaran, for being part of the thesis committee.

I also would like to thank my parents, who raised me with all love and inspiration for being a productive person. Finally, I would like to thank my wife, who is always supported me and suffered with me all these years in order to achieve my goals and dreams.
Abstract

In this thesis, we explore different avenues to reduce the power consumption of a 16x16 Multiplier. Our approach focuses on an interconnection pattern for the partial product reduction stage of the multiplier, which is divided into three stages. Each stage uses, half adder, full adder and 4:2 compressor modules in its design. The outputs from each stage connect to the inputs of the next stage. The interconnection pattern is based on an effective input capacitance, a parameter defined for each input lead of a logic device. Based on our strategy, the output with the highest switching activity at stage N is connected to the input with the lowest effective capacitance at stage $N+1$. This approach will result in minimizing the overall power dissipation of the entire partial product reduction stage for the 16x16 Multiplier. The design was carried out using 50nm CMOS technology using Electric VLSI tools, and simulations were carried out using LTspice. Our design was verified by simulation, and was found to consume 1.8mW of power. This is more than 10% less compared to the ones reported in literature.
Table of Contents

1. Introduction ................................................................. 1
   1.1. Power Reduction in Digital Systems .............................. 1
   1.2. Thesis Objective and Scope ...................................... 3
   1.3 Thesis Approach ..................................................... 3
   1.4 Thesis Organization ............................................... 5

2. Power Dissipation ....................................................... 7
   2.1. CMOS Power Dissipation .......................................... 7
   2.1.1 Dynamic Power Dissipation .................................... 8
   2.1.2 Short Circuit Power Dissipation ............................... 9
   2.1.3 Leakage Power Dissipation .................................... 11

3. Multipliers ............................................................... 14
   3.1. Partial Product Generation ..................................... 15
   3.2. Partial Product Reduction ....................................... 17
      3.2.1. Partial Product Reduction using Ripple Carry Adder .... 17
      3.2.2. Partial Product Reduction using Carry Save Adder ....... 18
      3.2.3. Partial Product Reduction using Wallace/Dadda Approach .... 20
      3.2.4. Partial Product Reduction using 4:2 Compressors .......... 22
   3.3. Booth Encoding .................................................. 25
   3.4. Booth Encoding Hardware ........................................ 29
      3.4.1. Ohkubo’s Encoder and Partial Product Generation ........ 29
      3.4.2. Cho’s Encoder and Partial Product Generation ........... 31
      3.4.3. Wu’s Encoder and Partial Product Generation ............ 33

4. Power Minimization Techniques ...................................... 36
   4.1. Switching Activity ............................................... 37
List of Tables

Table 3.1. The maximum number $n(h)$ of inputs for h-level CSA tree ..................23
Table 3.2. Radix-2 Booth Coding ..............................................................................26
Table 3.3. Radix-4 Booth Coding ..............................................................................27
Table 3.4. Radix-4 Booth Coding Operations ..........................................................28
Table 3.2. The truth table for Ohkubo’s radix-4 Booth encoder .........................30
Table 3.3. The truth table for Cho’s radix-4 Booth encoder .................................33
Table 3.4. The truth table for Wu’s radix-4 Booth encoder .................................34
Table 4.1(a). Half adder output probabilities .........................................................38
Table 4.1(b). Full adder output probabilities ..........................................................38
Table 4.1(c). 4:2 Compressor output probabilities ..................................................39
Table 4.2. Summary for Full Adder Effective Capacitance .....................................48
Table 4.3. Summary for 4:2 Compressor’s Effective Capacitances ......................50
Table 5.1. The Truth Table of logical AND Operation ............................................59
Table 5.2. Truth Table of XOR Operation ..............................................................61
Table 5.3. Truth Table of the Half Adder .................................................................65
Table 5.1. Effect of X1 on power dissipation of the 4:2 Compressor ....................76
Table 5.2. Summary of the Power Dissipation of the 4:2 Compressor due to each input ..........................................................76
Table 5.3. Effect of A on power dissipation of the Full Adder Unit .................77
Table 5.4. Summary of the Power Dissipation of the Full Adder Unit due to each input ..........................................................77
Table 5.5. Power Dissipation for the proposed interconnection of three 4:2 Compressors .

................................................................. 79

Table 6.1. Power Dissipation of the Partial Product Reduction Stage . ................. 84
List of Figures

Figure 2.1(a). Inverter, charging mode .................................................. 9
Figure 2.1(b). Inverter, discharging mode ................................................. 9

Figure 2.2. Short Circuit Current in Inverter. ........................................... 10
Figure 2.3. Inverter, leakage power dissipation ......................................... 11
Figure 2.4. Gate Leakage current in a CMOS transistor ............................ 12

Figure 3.1. Digital Multiplication Flow ..................................................... 15
Figure 3.2. Partial Product generation for 6x6 Multiplier ............................. 16
Figure 3.3. Partial Product generation for 6x6 Multiplier ............................. 19
Figure 3.4. Partial Product Reduction in 6x6 PPA using CSA ...................... 20
Figure 3.5. Wallace Reduction of 6x6 PPA using CSA ............................... 21
Figure 3.6. 4:2 compressor using two full adders ..................................... 24
Figure 3.7. Jiang’s 4:2 Compressor Design ................................................. 25
Figure 3.8. PPG for Radix-4 Booth encoding for 8x8 Multiplier .................. 28
Figure 3.9(a). Ohkubo’s Radix-4 Booth Encoder ....................................... 30
Figure 3.9(b). Ohkubo’s Radix-4 partial product generator ........................ 31
Figure 3.10(a). Cho’s Radix-4 Booth Encoder ........................................... 33
Figure 3.10(b). Cho’s Partial Product Generator ......................................... 33
Figure 3.11(a). Wu’s Radix-4 Booth Encoder ............................................ 35
Figure 3.11(b). Wu’s Partial Product Generator ......................................... 35
Figure 4.1(a). Half adder logic circuit. .................................................. 36
Figure 4.1(b). Full adder logic circuit. .................................................. 36
Figure 4.1(c). Jiang’s 4:2 compressor logic circuit. ................................... 37

Figure 4.1. Partial Product Reduction Stages ............................................. 38
Figure 4.2. Partial Product Reduction Stages, showing the structure of the 4:2 compressor, full adder, and half adder blocks ............................................. 40

Figure 4.3(a). 4:2 Compressor ............................................. 41
Figure 4.3(b). Full Adder ............................................. 41
Figure 4.3(c). 4:2 compressor ............................................. 41
Figure 4.4(a). 4:2 Compressor capacitance ............................................. 42
Figure 4.4(b). Full Adder capacitance ............................................. 42
Figure 4.5. Capacitance of unit size Inverter ............................................. 43
Figure 4.6. Capacitance of XOR gate ............................................. 44
Figure 4.7. Capacitance of MUX gate ............................................. 45
Figure 4.8(a). Effective capacitance for A ............................................. 47
Figure 4.8(b). Effective capacitance for B ............................................. 47
Figure 4.8(c). Total effective capacitance for Cin. ............................................. 48
Figure 4.9(a). Effective capacitance for X1. ............................................. 49
Figure 4.9(b). Effective capacitance for X2. ............................................. 49
Figure 4.9(c). Effective capacitance for X3. ............................................. 49
Figure 4.9(d). Effective capacitance for X4. ............................................. 49
Figure 4.9(e). Effective capacitance for Cin. ............................................. 50
Figure 4.10. Connection between two 4:2 compressor in the first stage and one 4:2 compressor in the second stage ............................................. 53
Figure 5.1(a). Schematic Design of Inverter. ............................................. 56
Figure 5.1(b). Layout Design of Inverter. ............................................. 57
Figure 5.2(a). FO4 Schematic Design in Electric VLSI. ............................................. 58
Figure 6.1. Proposed Interconnection for the Partial Product Reduction Stages. ........ 82

Figure 6.2. Nageshwar’s Proposed Interconnection for the Partial Product Reduction Stages ......................... 83

Figure 6.3. Random Interconnection for the Partial Product Reduction Stages. ........ 84
Chapter 1

Introduction

Power reduction is one of the major challenges facing today's VLSI designers. With every new generation, the circuit complexity and processing speed increases as predicted by Moore’s law in 1960’s, but only the rate has slowed down. With this, the power dissipation inside the chip also increases in an exponential manner. In general, the majority of power dissipation in the chip is directly proportional to the speed of operation and hence the removal of heat from the chip becomes extremely important as the frequency goes up. Most of the time, the only means for economic heat removal is by air-cooling. However, our smart devices, such as iPhones and iPads, do not contain any cooling mechanism inside the device. This resulted in the frequency limitation between 4-5GHz in current processors.

1.1. Power Reduction in Digital Systems

A lot of research has been focused into the minimization of power in VLSI designs. An efficient method of reducing the power dissipation is the supply voltage reduction, since it gives quadratic reduction in the overall power. However, the supply voltage is limited by the minimum transistor threshold voltage and subsequent reduction in circuit noise margins. The parasitic capacitance reduction by efficient routing of the chip layout also reduces power, as the dynamic power consumption is directly related to capacitance [1]. At the circuit level, lot of work has been done to minimize the switching activity occurring at the internal nodes [2]. In addition, system level approaches, such as parallel processing and pipelining are implemented in many recent designs due to the availability of multiple cores in the same chip.
In this research, we are concerned with the power minimization in hardware multipliers. This has merit because of the large amount of arithmetic processing occurring in present day handheld devices. Hardware multipliers are always preferred over software ones, especially when speed is a critical factor. As an example, a large number of multiplication operations are performed while processing large video images. In such cases hardware multipliers are preferred over software ones when speed is of concern.

A multiplier, in general, can be divided into a number of sub modules. At the top level, a partial product generator generates all partial products in parallel. Next, these partial products are added together using a partial product reduction stage, to reduce them down to two operands. To reduce the number of partial products, which effectively minimizes the delay for the addition of these partial products, modified Booth techniques are used in many designs. And finally a carry propagate adder is used to combine the two operands to generate the final result. We restrict our research here to the partial product reduction stage. In fact, our goal is to minimize the power consumed by the partial product reduction stage, as this takes up a big portion of the total power consumed by the multiplier.

The partial product reduction stage is usually implemented by an array of carry save adder stages, as proposed by Wallace and Dadda, resulting in logarithmic delay for the addition of partial products [3][4]. But the use of 4:2 compressors instead of carry save adders was found to produce a more regular interconnection pattern for the partial product reduction stage [5]. To this end, we decided to probe more on the interconnect pattern, by quantifying the parasitic capacitance presented at the inputs of half adders, full adders and 4:2 compressors. We realized that by using the parasitic capacitance, together with the transition probability of the signal lines,
a more efficient interconnection pattern can be achieved, which minimizes the overall power consumption in the partial product reduction stage.

1.2 Thesis Objectives and Scope

The main focus of this thesis is on power reduction techniques that will insure minimum power dissipation for a 16x16 Multiplier, by manipulating the interconnection patterns in the partial product reduction stage, based on a special parameter, called the total effective input capacitance. The techniques for power optimization will be done both at the Architectural Design Level and Device Level, which will be justified theoretically and verified by simulation.

The scope of this thesis is mainly on understanding the basics of a multiplier unit, especially half adders, full adders, multiplexers, and building arithmetic units using them. In addition, this thesis includes the basics of CMOS circuit design and power dissipation calculations. Moreover, the design and simulations will be done using 50nm CMOS technology using Electric VLSI tools and LTspice.

1.3 Thesis Approach

The objective of this research is to device an interconnection strategy for the logical modules used in the partial product reduction stage of a 16x16 Multiplier, which will minimize the overall power dissipation of the entire unit. We formulated an analytical approach for power reduction based on a probabilistic model on the input of the multiplier. In our design, the partial product reduction stage is implemented using three logic modules: half adder, full adder, and 4:2 compressor.
Our theoretical approach uses a probabilistic technique to calculate the probability of each input and output of the partial product reduction stage of the 16x16 Multiplier. This in conjunction with the total effective input capacitance at each input of the 4:2 compressor, full adder, and half adder, will be used to provide a metric for power consumption. To accomplish this, we took into consideration the unit size capacitance for each input in the 4:2 compressor, full adder, and multiplexer circuits, and used those numbers for calculating the total effective input capacitance for each input. This allows us to make the best decision about how to connect the outputs from a block at stage N, to the inputs of the next block at stage N+1, which is one level below it. Thus, the output of the logic block at stage N that has the largest switching activity gets connected to the input with the lowest effective capacitance at stage N+1.

Our simulation approach is aimed to verify the expected values of the power consumption of the partial product reduction stage of the 16x16 Multiplier using our proposed design against four other designs. The four designs that we will compare our design with are: Nageshwar’s design [6], and three other designs with random interconnection patterns used in the partial product reduction stage. This will verify whether our proposed design has lower power consumption compared to the other designs or not. It may be noted that Nageshwar’s design is already optimized, while the other random designs are interconnected in a random fashion.

We will use Electric VLSI tools to design the partial product reduction stage of the 16x16 Multiplier and all its internal circuits, such as 4:2 compressors, full adders, half adders, and multiplexers. LTspice was used for simulation purposes. This will help us to determine which input has the largest effect in power dissipation and which input has the smallest. More importantly, we will be able to validate our claim by simulation.
1.4 Thesis Organization

The overall workflow of the thesis will be as follows:

Chapter 2 presents an overview of the basics of power dissipation in CMOS circuits. In addition, it will cover the power related issues for multipliers and the techniques that could be applied to reduce power dissipation.

Chapter 3 covers the basic structure of multipliers. Moreover, it will illustrate the partial product reduction stage in multipliers, where many techniques can be used to reduce the partial product reduction stage. In addition, this chapter introduces Booth encoding techniques for both radix-2 and radix-4 algorithms. Finally, different hardware designs for implementing Booth encoding will be discussed.

Chapter 4 presents the emphasis of this research on deriving a technique that will be used for reducing the power dissipation of the partial product reduction stage of the 16x16 Multiplier. Switching activity, input capacitance, and effective capacitance parameters are discussed in detail in this chapter. A technique for the interconnection of logic modules in the partial product reduction stage to minimize power dissipation is also presented in this chapter.

Chapter 5 includes two parts. The first part covers all the designs that have been made for supporting this research. Logic units such as, AND, XOR, MUX, half adder, full adder, and 4:2 compressor are designed using both schematic and layout views. The software that we are going to use for these logic designs is Electric VLSI. The second part, however, will be the simulation of all these logic devices using LTspice. In addition, power measurements and testing will be illustrated in this chapter as well.

Chapter 6 presents the results of power dissipation measurements of our design. In addition, a comparison will be made between our proposed design and four other designs, which will
justify our claim about reducing the overall power dissipation of the partial product reduction stages of the 16x16 Multiplier.

Chapter 7 presents the conclusions of this research and how the research can be extended in the future.
Chapter 2

Power Dissipation

In today’s technology, power reduction is a critical aspect for many different electronic devices, such as laptops, cell phones, smart wrist watches, and so on. Since each device has its own battery for powering the device, the time the device will stay on without recharging its battery is indirectly proportional to the power dissipation of the device. In other words, the more power dissipation the device has, the less time it will stay on without recharging its battery. Hence, it is very important to apply power reduction techniques in the design of these devices to obtain better performance.

2.1. CMOS Power Dissipation

Power dissipation is the primary factor in today’s technology. In the field of electrical engineering, power dissipation is defined as the amount of energy that is drawn from the power supply by an electrical circuit per unit time, which is given by the following equation.

\[ P = \frac{E}{t} \]  

Equation 2.1

where \( P \) is the power in watts, \( E \) is the energy in joules, and \( t \) is the time in seconds.

Nowadays, the use of Complementary Metal Oxide Semiconductor (CMOS) transistor has become the major building block in manufacturing portable electronic devices, such as laptops, cell phones, smart wrist watches, and many other electronic devices. The total average power dissipation in these circuits is divided into three main categories: dynamic power, short circuit power and leakage power. The total power dissipation \( P \) is therefore given as:

\[ P = P_{\text{dyn}} + P_{\text{short}} + P_{\text{leak}} \]  

Equation 2.2
where $P_{\text{dyn}}$ denotes the dynamic power dissipation, $P_{\text{short}}$ denotes the short circuit power dissipation, and $P_{\text{leak}}$ denotes the leakage power dissipation.

### 2.1.1. Dynamic Power Dissipation

The dynamic power dissipation in CMOS circuits is related to the charging and discharging of the capacitors that are present at each node in the circuit [1]. This can be clearly explained for an inverter circuit, using Figures 2.1(a) and 2.1(b). During a low to high transition (logic 0 to logic 1) at the output of the inverter, the load capacitor charges to $V_{\text{DD}}$. At this time energy is drawn from the power source. Half of the energy is saved in the capacitor and the other half is dissipated in the PMOS transistor (Figure 2.1(a)). In the next half cycle, when the output switches from high to low (logic 1 to logic 0), the energy from the capacitor is dissipated in the NMOS transistor (Figure 2.1(b)). The average dynamic power drained from power source is given by:

$$P_{\text{dyn}} = \alpha C_L V_{\text{DD}}^2 f$$  \hspace{1cm} \text{Equation 2.3}$$

where $\alpha$ is the switching activity that refers to the number of rising transitions at the output of the device in an interval $[0,T]$, where $f = 1/T$, $C_L$ is the load capacitance, $V_{\text{DD}}$ is the supply voltage, and $f$ is the frequency. Since the dynamic power is proportional to the square of supply voltage, supply voltage reduction is an excellent way to minimize dynamic power. However, since the supply voltage is limited by the threshold voltage of the transistors, the supply voltage cannot be reduced to very low values.

Furthermore, the average transition energy can be defined as the amount of energy consumed when a switching event occurs. This is also useful when comparing the power dissipation of
various designs. In addition, the dynamic power can be adjusted depending on the desired capacitances and delay specifications [1].

Moreover, when considering only the switching component of power dissipation, we can view the transition energy as:

\[
\text{Energy Per Transition} = \frac{p_{\text{total}}}{f} = C_{\text{eff}} \times V_{DD}^2
\]

where \( C_{\text{eff}} \) is the effective capacitance that is being switched, which is given by:

\[
C_{\text{eff}} = \alpha_{0,1} \times C_L
\]

where \( \alpha_{0,1} \) denotes the probability of a low to high transition at the output.

### 2.1.2. Short Circuit Power Dissipation

The short circuit power dissipation occurs when both NMOS and PMOS transistor networks are conducting simultaneously, which creates a direct path from the power supply (Vdd) to ground, as shown in Figure 2.2. This happens whenever there is a transition at the output node. The amount of short circuit power dissipated depends on the time interval during which both NMOS and PMOS transistors are conducting simultaneously. In fact, this depends on the input
signal rise and fall times, and also the transistor sizing used in the design. The short circuit power is given by:

\[ I_{sc} = \frac{K}{V_{DD}(V_{DD}-2V_{th})\tau NF} \]  
Equation 2.6

where \( K \) is a constant that depends on the transistor sizes, as well as on technology, \( V_{th} \) is the threshold voltage of the NMOS and PMOS transistors, \( \tau \) is the rise or fall time of the input signal, \( N \) is the average number of transitions in the inverter’s output, and \( f \) is the frequency [7].

Moreover, the short circuit current becomes dominant whenever the load capacitance is small, and/or when the rise and fall time of the input signals are large. However, we can approximate the time-averaged short circuit current, which is drawn from the power supply along with the corresponding short circuit power dissipation of the CMOS inverter, by assuming symmetric rise and fall time delays for the input signal, and symmetric threshold voltages for the transistors, which gives us the following equation for the short circuit power dissipation:

\[ P_{sc} = \frac{\beta}{12} (V_{DD} - 2V_{th})^3 \tau f \]  
Equation 2.6

where \( \beta \) is the transconductance of the transistor, and \( V_{th} \) corresponds to the threshold voltage of the PMOS and NMOS transistors [7].
2.1.3. Leakage Power Dissipation

Leakage power dissipation is strongly related to the technology used for the design and implementation of CMOS circuits. As the technology scales down in size, it improves the speed, area, power dissipation, and therefore the performance of the circuit [7]. But leakage power is a big problem for nanometer devices. This leakage current increase is due to many factors such as reverse biased diode junction leakage current, Gate leakage, and sub threshold leakage, as shown in Figure 2.3.

![Figure 2.3. Inverter, leakage power dissipation](image)

The reverse biased diode junction leakage current, however, can be illustrated in the inverter shown in Figure 2.3, where the leakage current can occur from the source or drain to the substrate via the reverse biased diode when the transistor is OFF [7]. In addition, there are two main components that are related to the reversed biased pn junction leakage, the first one is the minority carrier diffusion, which is located near the edge of the depletion region. The second component is due to the generation of electron-hole pairs that take place inside the depletion
region of the reversed biased junction. For example, the inverter is fed an input signal that has low voltage, say 0 V, then the NMOS transistor is going to be OFF while the PMOS transistor is ON, which in effect, give a high voltage at the output node of the inverter. While the inverter is in this mode, the voltage between the drain and the substrate of the NMOS transistor is going to be equal to the supply voltage, thereby, leakage current occurs.

The gate leakage current, however, is affected by two other current sources, the gate induced drain leakage current ($I_{GIDL}$) and the gate direct tunneling leakage current ($I_{Gate\ Tunnel}$), which are illustrated in Figure 2.4. The gate leakage current is given by:

$$I_G = I_{GIDL} + I_{Gate\ Tunnel}$$

![Figure 2.4. Gate Leakage current in a CMOS transistor](image)

where $I_{GIDL}$ is the leakage current caused by the occurrence of a high field effect at the drain junction of the MOS transistor, and $I_{Gate\ Tunnel}$ is the leakage current that flows from the gate to the oxide insulation layer, and finally to substrate.
Lastly, the subthreshold leakage current ($I_{sub}$), also called weak inversion current, flows between the source and drain of the transistor. The effect of the subthreshold current, however, comes into play inside the diffusion current of the minority carriers located on the channel of the MOS device whenever the gate voltage is below the threshold voltage ($V_{Th}$) [7]. For example, in the case of an inverter that is fed a low voltage input signal, we assume that no current is flowing. In fact, this assumption is not accurate, because although $V_{GS}$ is at 0V, there will be a small current flowing in the gate of the NMOS transistor, which is OFF. This can be explained by the potential of the $V_{DS}$, which in this case is equal to $V_{DD}$.

There are several factors that are related to the subthreshold leakage current such as, the device size, supply voltage, temperature, and the parameters that contributes to $V_{Th}$. Furthermore, since the subthreshold leakage current is the highest of all leakage currents, it is preferred to keep the subthreshold current to a minimum as much as possible.

In conclusion, we can express the total leakage current $I_{Leak}$ as the sum of all three leakage currents. Thus, the average leakage power $P_{Leak}$ is given by:

$$P_{Leak} = I_{Leak} \cdot V_{DD}$$

Equation 2.7
Chapter 3

Multipliers

In this section, we will discuss the basics of digital multiplication and how it can be implemented in hardware. Digital multiplication is one of the most frequently used functions in today’s technology, especially in Digital Signal Processing, Digital Image Processing, Digital Filters etc. A multiplier, multiplies two numbers and outputs the result. However, when it comes to the design of a multiplier, there are many different designs available in literature. In terms of their size, they are classified as: 2x2 Multiplier, which means that it multiplies two 2-bits numbers together. Similarly, 4x4 Multiplier, 8x8 Multiplier, 16x16 Multiplier, and so on. However, the major challenge in designing an efficient multiplier lies on the partial product reduction stage. Algorithms by Wallace, Dadda, and Oskuii, have been developed for the purpose of minimizing the addition process that will be performed on the partial product [3] [4] [8].

In general, digital multiplication can be divided it into three steps: the first step is the partial product generation stage. In this step the entire multiplicand is first multiplied by the first bit of the multiplier (LSB). The result is called a partial product. Then the entire multiplicand is multiplied by the second bit of the multiplier, while shifting the whole partial product by 1-bit to the left. This process is repeated until all partial products are generated.

The second step is partial product reduction, where different techniques are used to add the partial products and reduce them into two operands. Finally, the two operands are adder together to get the final result.
For example, assume a 4x4 Multiplier, which multiplies two 4-bit numbers together \( A_3 \ A_2 \ A_1 \ A_0 \) and \( B_3 \ B_2 \ B_1 \ B_0 \). The partial product can be produced by multiplying each bit of the multiplier \( A_0 \) by the entire multiplicand \( B \), and then shift 1-bit to the left, and continue multiplying \( A_1 \) by the multiplicand \( B \). In general, for \( nxn \) Multiplier, there will be \( n \) rows of partial products generated. Thus, for the 4x4 Multiplier, 4 partial product rows will be generated, as shown in Figure 3.1. After all partial products are generated, a reduction technique will take place, such as the use of Carry Save Adder in Wallace and Dadda approaches, which will reduce the partial products into two operands. Finally, an \( n \)-bit carry propagate adder will be used to sum these two operands to produce the final result.

Figure 3.1. Digital Multiplication Flow

3.1 Partial Product Generation

In the partial product generation step, shifted copies of the partial products will be generated. This is done by using multiple two-input AND gates. To illustrate this, the partial product array (PPA) for a 6x6 Multiplier is shown in Figure 3.2. We can view each dot as a logical AND gate that takes the corresponding multiplier and multiplicand bits for \( p \) and \( c \), respectively. The generated columns, however, are to be added together to produce the final result.
Note that the bits of the PPA are generated in parallel; this means that the static delay for each bit is the same. In addition, the dimensions of the PPA depends on the size of both multiplier and multiplicand, where the height of the PPA is proportional to the size of the multiplier, and the width of the PPA is proportional to the size of the multiplicand. Last but not the least, for each column to be added, full adders can be used to add all the bits that are in one column. Moreover, it depends on how many bits are to be added inside the column, in order to determine how many full adders are needed. However, besides the full adder, 4:2 compressors and half adders are also used to reduce the addition process. Furthermore, by looking at the partial product reduction stage in Figure 3.2, we can observe that more additions will be required on the middle part of that partial product stage than on the sides. However, the left side, which is high-order side, requires a little more addition than the right side, or the low-order side. This is because of the
carry propagation that occurs during addition, which propagates from the low-order side to the high-order side.

3.2 Partial Product Reduction

The essence of the partial product reduction stage is to speed up and/or to minimize the addition process, which takes place in the PPA. There are many different techniques available for the reduction of the partial product bits. The most basic technique, however, uses two-dimensional arrays that consist of full adders and half adders, which are arranged in a ripple carry fashion. However, since the ripple carry arrangement produces large delays for carry propagation, this technique is considered to be very slow. Therefore, many smart techniques have been proposed to reduce addition time. We will discuss some of these techniques in the coming sections.

3.2.1 Partial Product Reduction using Ripple Carry Adder

This is the simplest structure used for partial product reduction. The ripple carry adder (RCA) technique uses both full adders and half adders to construct the partial product reduction stage. Figure 3.3 shows the partial product reduction stage for a 4x4 Multiplier using this technique. For the first two bits to be added in each row, half adders are used because there are no carries to
be added at this stage. After that full adders are used, which produce a carry-in to the next full adder and so on.

Despite the low speed of the RCA technique, it illustrates the basic concept for the addition of partial products. They also fit very well for VLSI implementation because of their regular structure and simplified testing techniques [9]. Nonetheless, if we are to add m shifted copies of an n-bit multiplicand, using the ripple carry adder technique, the delay is $O(m+n)$, which is proportional to the width of the multiplier and multiplicand. This will eventually produce large delays. The main reason for this large delay is that each full adder will wait for the carry-out to be generated from the previous full adder, which means that the results will be produced from the last full adder after all carry-outs are generated and propagated to the last full adder.

### 3.2.2 Partial Product Reduction using Carry Save Adder

The ripple carry problem in RCA, however, is solved by using a carry save adder (CSA) structure, where the carry-out of the full adder does not go directly to the carry-in of the next full adder in the same level. Instead, it goes to the carry-in of the next full adder in the lower level.
This way, all full adders that are in the first level will produce their outputs instantly, without waiting for a carry-in from any other full adder in the same level. However, all carry-out’s from the first level will be connected to the next full adder to the left, one level below.

This is illustrated with a 6x6 Multiplier, shown in Figure 3.4, where in (a), we sum the first three 6-bit vectors inside the PPA, by using six full adders connected in a Carry Save Adder fashion, as illustrated in (b). Next, the outputs from this addition will be combined with the last three 6-bit vectors that are left in the PPA, as shown in (c), which will produce another set of vectors that needs to be added using another set of full adders. The complete design of the Carry Save Adder structure for a 6x6 Multiplier is shown in (d). The delay of the CSA structure for an nxn Multiplier is still going to be linear, as in RCA, but is $O(n)$, which is less than the delay in the RCA structure.
3.2.3 Partial Product Reduction using Wallace/Dadda Approach

Wallace proposed this approach in 1964, when he noticed that the later stages of the CSA array structure must always wait for the outputs produced from the earlier stages. In addition, he also noticed that PPA of multipliers contains the highest number of transistors because of the large number of half adders and full adders employed [3]. However, since each set of n-bit vectors are added individually, each addition stage is considered to be independent. Thus, by performing more than one add operation for two or more sets individually, in a parallel fashion,
the entire structure will have much lesser delay than if we were to add only one set together and 
wait for its outputs to be combined with the next set, and so on. This is illustrated in Figure 3.5, 
where it shows the Wallace reduction structure for a 6x6 Multiplier by employing two CSA sets.

![Wallace Reduction of 6x6 PPA using CSA](image)

The essence of parallelizing the add operations in the CSA structure is that it has a much 
shorter delay than using the sequential series of operations in the PPA. While the delay of the 
sequential operation is $O(n)$, the delay of the parallel operation is $O(\log_{3/2} n)$.

The parallel structure of the CSA’s is called Wallace tree, which significantly reduced the 
delay of the PPA stage. Moreover, in the final add operation in the array structures is of width $n$, 
while in Wallace tree, the final adder will have a width of approximately $(2n - \log_{3/2} n)$. This is 
because each CSA reduces the number of operands by a factor of 1.5. The smallest height $h(n)$ 
of an $n$-input Wallace tree can be estimated by:

$$h(n) = 1 + h\left[\frac{2n}{3}\right]$$

Equation 3.1
Furthermore, we can express the relationship between the number of inputs and the tree height by calculating the maximum number of inputs, which can be eventually reduced into two operands for a \( h \)-level tree. Thus, the equation for the maximum number of inputs \( n(h) \) is:

\[
n(h) = \left\lfloor \frac{3n(h-1)}{2} \right\rfloor
\]

Equation 3.2

On the other hand, there is a disadvantage in Wallace tree that lies in its irregular structure, where the array structure might require more wiring, which makes it complex in terms of wiring and connectivity. This brings us to Dadda’s approach, which is based on Wallace tree structure. However, in Dadda’s tree, the reduction of the number of operands is based on the values shown in Table 3.1, in which it reduces the operands below the next lower \( n(h) \) value [4].

### 3.2.4 Partial Product Reduction using 4:2 Compressors

Both Wallace and Dadda trees use only full adders and half adders in their design, which requires much wiring complexity in the parallel process. This wiring complexity, however, results in more power consumption for the design, which in turn, decrease the performance of the multiplier, especially for large multipliers.

To solve this problem, compressors such as, 4:2, 5:2, 7:4, and 9:2 have been designed, in which the wiring complexity for parallel connection in the partial product reduction stage have been reduced, while also reducing the height of the partial product reduction tree [10] [11].
Table 3.1. The maximum number $n(h)$ of inputs for h-level CSA tree

<table>
<thead>
<tr>
<th>$h$</th>
<th>$n(h)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>42</td>
</tr>
<tr>
<td>9</td>
<td>63</td>
</tr>
<tr>
<td>10</td>
<td>94</td>
</tr>
</tbody>
</table>

A good example of using compressors in the partial product reduction stage is Goldvsky’s design, which uses 3:2, 4:2, and 7:4 compressors for adding the partial products [12]. The use of compressors in the partial product reduction stage was also found to decrease the overall power consumption of the multiplier [13]. The 4:2 compressor, however, is frequently used in multipliers for achieving better performance. In general, to construct a 4:2 compressor, we can use two cascaded full adders, as shown in Figure 3.6, where five inputs are being added together to generate three outputs, sum (S), and two carries: carry (Co) and carry-out (Cout). They are given by:
\[ S = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{in}. \]  
\[ C_{out} = (X_1 \oplus X_2).X_3 + \overline{(X_1 \oplus X_2)}.X_1, \] and
\[ C_o = (X_1 \oplus X_2 \oplus X_3 \oplus X_4).C_{in} + \overline{(X_1 \oplus X_2 \oplus X_3 \oplus X_4)}.X_1 \]  

Equation 3.3
Equation 3.4
Equation 3.5

Nonetheless, one of the best implementations that have been developed for the 4:2 compressors is Jiang’s design, shown in Figure 3.7 [14]. This design has improved and simplified the signal flows, whereas it contains four XOR gates and two 2x1 multiplexers, which in turn decreases both the delay and the overall power consumption of the 4:2 compressor.

![Figure 3.6. 4:2 compressor using two full adders](image)

The use of 4:2 compressors in both Wallace and Dadda trees can simplify the connectivity in the partial product reduction stage, along with reducing the height of the tree [15][16]. Nevertheless, many different designs for different compressors are available, in which most of them have been designed for the purpose of higher speed of operation [11].
3.3. Booth Encoding

In 1950, Andrew D Booth, has devised an algorithm for the purpose of multiplying two signed numbers together using a uniformly formatted algorithm, which can repeat the same process in order to obtain the correct pattern for its outputs by encoding the bits of the multiplier operand. This is also another technique for speeding up multiplication by reducing the number of partial products. In the previous sections, we discussed the partial product reduction methods by reducing the number of subsequent calculations of the generated partial products. Booth encoding, however, uses a different approach that can generate a reduced version of the partial products themselves before the addition process takes place [17]. The main strategy for Booth encoding is to subdivide the multiplier into individual bits, and then encode each bit in such a
way so as to produce a reduced pattern for the partial product, which later will be added together to produce the final result.

Booth encoding was initially based on Radix-2 multiplication [17], in which by encoding the multiplier bits, it can skip the addition process if the partial product bit is 0, thereby eliminating an addition operation. This ultimately reduces the number of add operations for all the 0’s in the partial product, while implementing shift operations instead, which in turn speeds up the process and reduces the overall power consumption. Table 3.2, shows Radix-2 Booth Coding. In general, the process starts by scanning the first two bits of the multiplier \((x_{i-1} \text{ and } x_i)\) from right to left, and based on these two values, the encoded multiplier bit \(y_i\) corresponding to \(x_i\) can be obtained. \(y_i\) determines the type of operation that will be performed in each cycle, whether it is addition, subtraction or no-operation, where 1 indicates the addition operation, \(\bar{1}\) indicates the subtraction operation, and 0 indicates that no-operation is performed.

<table>
<thead>
<tr>
<th>(x_i)</th>
<th>(x_{i-1})</th>
<th>(y_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>(\bar{1})</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Furthermore, multiplication becomes slower in Radix-2 Booth encoding if alternate 1’s and 0’s exist in the binary representation of the multiplier operand. However, if a consecutive group
of 1’s exists, then it can be replaced by a subtraction at the least significant end, and an addition operation to the bit that is on the left of its most significant bit, which in effect will speed up the process. Thus, the longer consecutive 1’s there are, the more saving the circuit will have.

On the other hand, if more alternating sequences of 1’s and 0’s exist, this will increase the number of add operations, which leads to an increase in the delay and the overall power consumption. To overcome this problem, similar techniques have been developed using Radix-4 Booth encoding, in which it scans each three bit groups of the multiplier from right to left, and based on their values, a value for the recorded digit Di is obtained, which determine the operation to be performed [18] [19] [20]. The operations determine whether the shifted and/or complementary copy of the multiplicand is to be used in the operation. Table 3.3, shows the truth table of the Radix-4 Booth coding [21], whereas the corresponding operation for the recoded digits are shown in Table 3.4.

<table>
<thead>
<tr>
<th>Xi+1i</th>
<th>Xi</th>
<th>Xi-1</th>
<th>Di</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 3.4. Radix-4 Booth Coding Operations

<table>
<thead>
<tr>
<th>Recorded Digit $D_i$</th>
<th>Operation on multiplicand X</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Shift left next partial product two positions</td>
</tr>
<tr>
<td>+1</td>
<td>Add X to the partial product</td>
</tr>
<tr>
<td>+2</td>
<td>Shift left X one position and add it to the partial product</td>
</tr>
<tr>
<td>-1</td>
<td>Add two’s complement of X to the partial product</td>
</tr>
<tr>
<td>-2</td>
<td>Shift left two’s complement of X one position and add it to the partial product</td>
</tr>
</tbody>
</table>

To observe the reduction in the number of partial products, let us consider implementing Radix-4 Booth encoding to a 8x8 Multiplier, shown in Figure 3.8, where it reduces the size of the PPA from eight to five.

![Figure 3.8. PPG for Radix-4 Booth encoding for 8x8 Multiplier](image-url)
3.4. Booth Encoding Hardware

There are many different encoding designs available that use Booth encoding for generating the partial products. Each design has its own advantages and disadvantages in terms of speed and power reduction. In the following three sections, however, we will discuss three of these designs proposed by Ohkubo, Cho, and Wu, respectively.

3.4.1 Ohkubo’s Encoder and Partial Product Generator

Ohkubo had proposed a new design for the encoding process using Radix-4 Booth encoding, where he took advantage of the use of pass logic multiplexer for the purpose of speeding up the process [22]. The truth table for Ohkubo’s encoder is shown in Table 3.2, which uses the three bits of the multiplier operand, \( Y_{i-1}, Y_i, \) and \( Y_{i+1} \), in order to generate one of the three output signals, \( a \), \( 2a \), and \( \text{NEG} \), where:

\[
a = Y_{i-1} \oplus Y_i \quad \text{Equation 3.6}
\]

\[
2a = (Y_{i-1} \cdot Y_i) \cdot \overline{Y_{i+1}} + (Y_{i-1} \cdot \overline{Y_i}) \cdot Y_{i+1}, \quad \text{and} \quad \text{Equation 3.7}
\]

\[
\text{NEG} = Y_{i+1} \quad \text{Equation 3.8}
\]

Figure 3.9(a), shows the logic circuit of Ohkubo’s Radix-4 Booth encoder, where the multiplier bits are given as inputs to the encoder. The logic circuit, however, is designed using two AND gates, one XOR gate, one 2x1 multiplexer, and three buffer gates. Note that the NEG
bit is nothing but a copied version of the $Y_{i+1}$ bit, which is also being used in the multiplexer for its inverted bit, without the need of adding additional inverter to the circuit.

Table 3.2. The truth table for Ohkubo’s radix-4 Booth encoder

<table>
<thead>
<tr>
<th>$Y_{i+1}$</th>
<th>$Y_i$</th>
<th>$Y_{i-1}$</th>
<th>Booth Operation</th>
<th>a</th>
<th>2a</th>
<th>NEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 X</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 X</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2 X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2 X</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1 X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1 X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-0 X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3.9(a). Ohkubo’s Radix-4 Booth Encoder

The design of the partial product generator proposed by Ohkubo is shown in Figure 3.9(b). The design consists of three NAND gates and one XOR gate, which produce the partial product bit.
3.4.2 Cho’s Encoder and Partial Product Generator

Cho’s encoder was proposed back in 2003, where he also based his technique on the modified Radix-4 Booth encoding algorithm. The advantage of Cho’s encoder is that it contains less number of components compared to the other designs such as Ohkubo’s encoder [20]. Table 3.3 shows the truth table for Cho’s Radix-4 Booth encoder, where he used three bits from the multiplier operand, $Y_{i-1}, Y_i, \text{ and } Y_{i+1}$, in order to produce three output signals, Direction ($D_m$), which determines whether the partial product is positive or negative, Shift ($S_m$), which determines whether the partial product is a left shifted copy of the multiplicand or not, and Addition ($A_m$), which determines whether the multiplicand will be added to the partial sum or not. They are given as:

\[
D_m = Y_m + 1, \quad \text{Equation 3.9}
\]

\[
S_m = Y_{m+1} \oplus Y_m, \quad \text{and} \quad \text{Equation 3.10}
\]

\[
A_m = Y_m \oplus Y_{m-1} \quad \text{Equation 3.11}
\]
The design of Cho’s Radix-4 Booth encoder consists of two XOR gates and three buffers, as shown in Figure 3.10(a), which contains fewer components compared to Ohkubo’s design. Furthermore, the number of transistors in each design is determined based on the implementation of the internal circuits for each logic gate inside each design.

Further, in order to construct a 16x16 Multiplier using Cho’s encoding technique, it requires nine Booth encoders in order to generate nine partial products. However, simple modifications in the logic structure, such as sharing signals can always be implemented to achieve an efficient process flow. Thus, the modified Cho’s Radix-4 Booth encoder will use 15 XOR gates and 25 buffer gates.

Figure 3.10(b) shows the design of the partial product generator proposed by Cho. As we notice in the design, the Partial Product Generator (PPG) consists of three 2x1 multiplexers connected in a cascade fashion, where all three multiplexers receive their select signals from the outputs $D_m, S_m$, and $A_m$, while the top multiplexer receives its input signals from the multiplicand bit $X_i$. 
Table 3.3. The truth table for Cho’s radix-4 Booth encoder

<table>
<thead>
<tr>
<th>$Y_{i+1}$</th>
<th>$Y_i$</th>
<th>$Y_{i-1}$</th>
<th>Booth Operation</th>
<th>Direction ($D_m$)</th>
<th>Shift ($S_m$)</th>
<th>Addition ($A_m$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 X</td>
<td>0</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 X</td>
<td>0</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2 X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2 X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1 X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1 X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-0 X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3.10(a). Cho’s Radix-4 Booth Encoder

Figure 3.10(b). Cho’s Partial Product Generator

3.4.3 Wu’s Encoder and Partial Product Generator

A third design implementing Booth encoding algorithm has been proposed by Wu, et al in 2005 [19]. Wu’s method, however, focused on both increasing the speed of the process and
reducing the hardware implementation. Moreover, by achieving these two properties, the design will in turn have lower power consumption. The truth table of Wu’s Radix-4 Booth encoder is shown in Table 3.4, where it uses a set of three bits from the multiplier and produces three output signals, Shift, a1, and a0.

Figure 3.11(a) shows the design of Wu’s Radix-4 Booth encoder. The design, however, is constructed of one XNOR gate, one 2x1 multiplexer, one inverter, and three buffers, where the number of transistors of the entire design is determined by the internal implementation of each logic gate inside the design.

Figure 3.11(b) shows Wu’s proposed design for the PPG, which consists of two cascaded sets of three 2x1 multiplexers, to give a total of six 2x1 multiplexers. Note that the inputs of top set, which contains three 2x1 multiplexers, comes from the individual bits of the multiplicand, while their outputs are the three select signals for the multiplexers that are in the lower set.

Table 3.4. The truth table for Wu’s radix-4 Booth encoder

<table>
<thead>
<tr>
<th>$Y_{i+1}$</th>
<th>$Y_i$</th>
<th>$Y_{i-1}$</th>
<th>Booth Operation</th>
<th>Shift</th>
<th>a1</th>
<th>a0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 X</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1 X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 X</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2 X</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2 X</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1 X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1 X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0 X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 3.11(a). Wu's Radix-4 Booth Encoder

Figure 3.11(b). Wu's Partial Product Generator
Chapter 4

Power Minimization Techniques

A theoretical framework for dynamic power minimization in the partial product reduction stage in a multiplier is presented here. Three different logical units are used in the partial product reduction stage. They are: half adder, full adder and 4:2 compressor, shown in Figures 4.1(a), 4.1(b), and 4.1(c), respectively. Only dynamic power consumption is considered in this work. The dynamic power consumption is directly proportional to the capacitive load presented at each and every node in a CMOS circuit. It is also a direct function of the switching (transition) probabilities at these nodes.

Figure 4.1(a). Half adder logic circuit  Figure 4.1(b). Full adder logic circuit
4.1. Switching Activity

As mentioned earlier, we are only going to focus on the partial product reduction stage in a 16x16 radix-4 Booth coded multiplier. The partial products, however, will have 16 bits in each row. Each successive partial product will be shifted left by 2-bits, since it is generated by Radix-4 Booth encoding technique. For a 16x16 multiplier, the partial product reduction stage can be arranged into three reduction stages as shown in Figure 4.1 [6]. The inputs to the logic modules in the first reduction stage are the partial product bits. Assuming a static probability of 0.5 for each operand bit, the static probability of each product bit will be 0.25. Hence, we will assume the same input probability for all inputs in the first reduction stage. The switching probability equations at the outputs of half adder, full adder and 4:2 compressor are given in Appendix A. Using these equations, and knowing the input probabilities, we can calculate the output probabilities (static) of half adder, full adder and 4:2 compressor. They are tabulated in Tables 4.1(a), 4.1(b) and 4.1(c) respectively.
Table 4.1(a). Half adder output probabilities

<table>
<thead>
<tr>
<th>Output</th>
<th>Boolean Equations</th>
<th>Static Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cout</td>
<td>$A, B$</td>
<td>0.0625</td>
</tr>
<tr>
<td>S</td>
<td>$A \oplus B$</td>
<td>0.375</td>
</tr>
</tbody>
</table>

Table 4.1(b). Full adder output probabilities

<table>
<thead>
<tr>
<th>Output</th>
<th>Boolean Equations</th>
<th>Static Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cout</td>
<td>$(A \oplus B).Cin + (A \oplus B).A$</td>
<td>0.1563</td>
</tr>
<tr>
<td>S</td>
<td>$A \oplus B \oplus C$</td>
<td>0.4375</td>
</tr>
</tbody>
</table>
Table 4.1(c). 4:2 Compressor output probabilities

<table>
<thead>
<tr>
<th>Output</th>
<th>Boolean Equations</th>
<th>Static Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cout</td>
<td>((X_1 \oplus X_2).X_3 + (X_1 \oplus X_2).X_1)</td>
<td>0.2266</td>
</tr>
<tr>
<td>Co</td>
<td>((X_1 \oplus X_2 \oplus X_3 \oplus X_4).C_{in} + (X_1 \oplus X_2 \oplus X_3 \oplus X_4).X_4)</td>
<td>0.1563</td>
</tr>
<tr>
<td>S</td>
<td>(X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{in})</td>
<td>0.4844</td>
</tr>
</tbody>
</table>

However, the outputs from the first reduction stage will be the inputs to the second reduction stage, therefore, the output probabilities from the second reduction stage will differ for the half adder, full adder, and 4:2 compressor blocks. Likewise, the outputs from the second stage will be the inputs to the third reduction stage, which will produce different switching probabilities at the outputs of the third reduction stage. Therefore, at each stage we will calculate the output probabilities from the half adder, full adder and 4:2 compressor blocks using the probability equations given in Appendix A. All three blocks, 4:2 compressor, full adder, and half adder, will be used in all three reduction stages to produce the final stage, which is applied to a 32-bit adder, as shown in Figure 4.2. Nevertheless, the first stage contains eight 4:2 compressors, seven full adders, and eight half adders. The second reduction stage, on the other hand, contains eighteen 4:2 compressors, four full adders, and three half adders while the last reduction stage contains twenty three full adders and two half adders. All outputs from the logic blocks in the first stage will be connected to the inputs of the logic blocks in the second reduction stage, and the rest of the unused partial products will be transferred to the second stage with no change.
In order to understand how the connections are made between the output of one stage and the inputs of the next stage, we need to study and analyze the weight of the output bits of each logic block. Figure 4.3(a) shows the 4:2 compressor block, which takes 5 inputs, X1, X2, X3, X4, and Cin, from the same column i, and produces three outputs S, Co, and Cout, where S is produced in the same column i, and Co and Cout are given to the next column to the left, i+1. Likewise, Figures 4.3(b) and 4.3(c) show the full adder and half adder blocks, respectively, both producing two outputs each, S and Cout, in which S is produced in the same column (column i), and Cout is produced in the column to the left (column i+1).
4.2. Input Capacitance

An earlier work on power minimization for the partial product reduction stage used the concept of effective capacitance [6]. The effective capacitance is defined as the product of switching activity and load capacitance at a node. Power reduction was achieved by minimizing the effective capacitance. In their work, they considered the capacitance presented at the input of a logic gate as one unit capacitance. They did not differentiate between different logic gates. For example, in the 4:2 compressor block shown in Figure 4.1(c), the input node $X_1$ is connected to an XOR gate and a MUX. So the capacitance seen by $X_1$ is 2 unit capacitance (denoted by $C_2$), the subscript denotes the number of unit capacitances. The same is true for $X_3$, $X_4$, and $C_{in}$. But $X_2$ is connected only to one XOR gate input and hence its value is denoted as $C_1$. This is shown in Figure 4.4(a).

A similar approach is used for the full adder shown in Figure 4.1(b). In the full adder, the inputs $A$ and $C$, both are connected to 2 logic gates, whereas $B$ has only one connection. This
corresponds to 2 unit capacitances for both A and C, and 1 unit capacitance for B, as illustrated in Figure 4.4(b).

![Figure 4.4(a). 4:2 Compressor capacitance](image)
![Figure 4.4(b). Full Adder capacitance](image)

In the presentation below we will show that the capacitance estimation used in the above manner will not give a correct estimate for CMOS circuits. First, in order to get the best estimate for capacitance, we will use the inverter as a base unit, since it uses the minimum transistor size for both NMOS and PMOS transistors. For simplification, we will assume that the minimum transistor size for the NMOS transistor is 1 and for the PMOS transistor is 2. This is marked in Figure 4.5. This assumption is valid considering the mobility ratios of the carriers in the two transistors. In addition, we will assume that the gate of the NMOS transistor presents 1 unit capacitance (1C), and the gate of the PMOS transistor presents 2 unit capacitance (2C), as shown in Figure 4.5. Therefore, the total capacitance at the input node of the inverter is equal to 3C.
Based on the above, we can now estimate the capacitance for each input in an XOR gate. Figure 4.6 shows the transistor schematics of a two-input XOR gate. The transistor sizes and the corresponding capacitances are marked in Figure 4.6. From Figure 4.6, it may be noted that both inputs A and B present the same capacitance, which is 9C. Similarly, Figure 4.7, shows a 2-to-1 MUX with transistor sizes and capacitance values. From Figure 4.7, it may be noted that the capacitance presented at each input A and B is 3C, while the selector lead (Sel) presents a capacitance equivalent to 9C. From the above discussion, we can conclude that by treating input capacitances of different logic devices as one unit capacitance contributes to large errors in capacitance estimation.
Figure 4.6. Capacitance of XOR gate
4.2.1. Effective Capacitance

To get a more accurate estimate on the capacitance for each input, we are going to introduce a new concept. We look at each input lead and find its effectiveness in making a transition at the output leads of logic devices. This can be done by calculating the probability of an output transition whenever the input makes a transition, and can easily be calculated by finding \( P\left(\frac{dy}{dx}\right)\), where \( \frac{dy}{dx} \) represents the Boolean difference of the output ‘y’ with respect to the input ‘x’. The
Boolean difference $\frac{dy}{dx}$, where $\frac{dy}{dx} = y_{x=0} \oplus y_{x=1}$, specifies under what conditions a transition on x induces a transition on y. $P\left(\frac{dy}{dx}\right)$ gives the probability of a transition at y due to a transition on x input. Now $TP(x) \times P\left(\frac{dy}{dx}\right)$ represents the total transition probability at y whenever there is a transition on x, where $TP(x)$ denotes the transition probability at x. If the output signal y drives a capacitance $c_y$, then $c_y TP(x)P\left(\frac{dy}{dx}\right)$ represents an effective capacitance which contributes to the dynamic power consumption due to the capacitance $c_y$ resulting from a transition on input x. Since power drawn from supply source occurs only when the output makes a low to high transition, only the transition probability value for either low to high or high to low is used for $TP(x)$. Hence, $TP(x) = P(x)P\left(\frac{dy}{dx}\right)$. The total effective capacitance with reference to dynamic power consumption for x can now be obtained by adding the individual effective capacitances at the different nodes along the path to the output, which are activated by x. The Boolean difference equations for the outputs of half adder, full adder and 4:2 compressor are given in Appendix B.

### 4.2.1.1. Total Effective Capacitances for Full Adder

A full adder is shown in Figure 4.8(a). First let us consider the effective capacitance at input A. The contribution comes from the direct capacitances seen at input A itself (12C from Figure 4.8(a)), and the capacitance at the output of the top XOR gate (node SS). These are highlighted in blue in Figure 4.8(a). Assuming static probabilities of 0.25 for all inputs of the full adder, $TP(A) = TP(B) = TP(C_{in}) = 0.1875$. 
\[
\frac{dSS}{dA} = 1, \text{ and hence } P\left(\frac{dSS}{dA}\right) = 1.
\]

The effective capacitance seen by A due to the capacitance at node SS is given by:

\[C_{ss} TP(A) P\left(\frac{dSS}{dA}\right) = 18C \times 0.1875 \times 1 = 3.375C,\]

where \(C_{ss}\) is the capacitance at node SS.

Hence, total effective capacitance at A = 12C + 3.375C = 15.375C. Input B has a direct capacitance of 9C as shown in Figure 4.8(b). The effective capacitance due to the capacitance at node SS is given by:

\[C_{ss} TP(B) P\left(\frac{dSS}{dB}\right) = 18C \times 0.1875 \times 1 = 3.375C\]

Hence, the total effective capacitance at B = 9C + 3.375C = 12.375C.

Lastly, if we choose input \(C_{in}\), we notice that it sees a direct capacitance of 12C, as shown in Figure 4.8(c). Table 4.2, shows the summary of the total effective capacitances at the inputs of the full adder.

![Figure 4.8(a). Effective capacitance for A](image1)

![Figure 4.8(b). Effective capacitance for B](image2)
Figure 4.8(c). Total effective capacitance for Cin

<table>
<thead>
<tr>
<th>Input</th>
<th>Total effective input capacitance</th>
<th>Number of affected units</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>15.375C</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>12.375C</td>
<td>3</td>
</tr>
<tr>
<td>Cin</td>
<td>12C</td>
<td>2</td>
</tr>
</tbody>
</table>

### 4.2.1.2. Total Effective Capacitances for 4:2 Compressor

The 4:2 compressor is shown in Figure 4.9(a), where it shows the path of the input X1 inside the 4:2 compressor block (blue color). The internal nodes are SS1 and SS2. The total effective capacitance at X1 is given by:

\[
12C + C_{ss_1} \times TP(X_1) \times P \left( \frac{dSS_1}{dX_1} \right) + C_{ss_2} \times TP(X_1) \times P \left( \frac{dSS_2}{dX_1} \right)
\]

\[
=12C + 18C \times 0.1875 \times 1 + 18C \times 0.1875 \times 1
\]

\[
=18.75C
\]
Figure 4.9(b), shows the capacitance for input $X_2$ (blue color). The total effective capacitance is estimated as 15.75C. Similarly, for $X_3$ it is 17.06C (Figure 4.9(c)), for $X_4$ the effective capacitance is 17.06C (Figure 4.9(d)) and for $C_{in}$ it is 12C (Figure 4.9(e)). Table 4.3, shows the summary of the total effective capacitance at the inputs of the 4:2 compressor.

![Figure 4.9(a). Effective capacitance for X1](image1.png)

![Figure 4.9(b). Effective capacitance for X2](image2.png)

![Figure 4.9(c). Effective capacitance for X3](image3.png)

![Figure 4.9(d). Effective capacitance for X4](image4.png)
4.3. Logic Module for Interconnection of Low Power

In the first part, we calculated all probabilities for the output nodes of the 4:2 compressor, the full adder, and the half adder blocks, shown in Tables 4.1(a), 4.1(b), and 4.1(c). In the second part, we calculated the total effective capacitances for each input for the 4:2 compressor and the full adder blocks, as shown in Tables 4.2 and 4.3. Based on this information, we can make the
best decision for the interconnection between the outputs from the first reduction stage and the inputs of the second reduction stage. To achieve minimum power, we connect the output with the highest probability to the input with the smallest total effective capacitance. This will reduce the overall switching power in the logic modules. The following example illustrates our interconnection strategy based on the above principle.

Example 4.1: Consider the interconnections between three 4:2 compressors, two in the first stage and one in the second stage. We want to connect the outputs of the two 4:2 compressors in the first reduction stage to the inputs of the 4:2 compressor in the second reduction stage. We feed 10 input signals, with the same frequency but different initial delays, to all ten inputs of the two 4:2 compressors that are in the top level. This is shown in Figure 4.10. The output probabilities and the effective input capacitance values are also shown in the figure. The best way to insure minimal power dissipation for the connections between the outputs of the two 4:2 compressors and the inputs of one 4:2 compressor at the bottom level is to connect the output with the highest probability to the input with the smallest total effective capacitance, as illustrated in Figure 4.10.

The output probabilities of the 4:2 compressors in the first stage are: 0.4844, 0.2266 and 0.1563 for S, C_O, and C_OUT respectively. Hence, there are two outputs with the highest probability of 0.4844. There are only five inputs in the second level 4:2 compressor. So one of the highest probability outputs (S output of the 4:2 compressor on the right in Figure 4.10) is passed straight down to the third level. The other higher probability output (S output of 4:2 compressor on the left) is connected to the lowest effective capacitance input C_in of the bottom 4:2 compressor. One of the lowest probability outputs (C_OUT of 4:2 compressor on the left in Figure 4.10) is connected to the highest effective capacitance input X_1. The other lowest
Probability output ($C_{OUT}$ of 4:2 compressor on right) is connected to $X_3$ input. The $C_O$ outputs from the top 4:2 compressor are connected to $X_2$ and $X_4$ inputs. We can define a metric for power by taking the product of output probability and the effective input capacitance for each interconnection, and summing all of them. This gives a metric for power equal to:

$$0.1563 \times 18.75C + 0.2266 \times 15.75C + 0.2266 \times 17.06C + 0.4844 \times 12C + 0.1563 \times 17.06C = 18.8C.$$ 

The power metric is an indication of the power dissipated in the device. Larger power metric means more power dissipation. Any other interconnection will have a higher power metric than the one obtained for the above interconnection.
Figure 4.10. Connection between two 4:2 compressor in the first stage and one 4:2 compressor in the second stage
Chapter 5

Design and Simulation

The design and simulation of each logic unit, such as XOR, multiplexer, full adder, half adder, and 4:2 compressor will be done using Electric VLSI Design System along with LTspice. The Electric VLSI Design System is a free open-source Electronic Design Automation (EDA) system that is provided by Static Free Software, and it can handle different forms of circuit designs such as, Schematic Capture, IC Layout, Textual Languages, and more. However, we will be using Electric VLSI to design both schematic and layout for each logic unit, in order to obtain the netlist of the design, which contains the SPICE code that will be simulated using LTspice. Moreover, LTspice, which stands for Linear Technology Spice, is a high performance SPICE simulator that can be interfaced with Electric VLSI, which simulates the extracted netlist and displays the resulting waveforms for verification. The design and simulation for each logic gate will be further explained in the following sections.

5.1. Design Specifications

First we list all of the transistor parameters and input signal parameters that will be used throughout in all our designs in this thesis. The test vectors, however, were chosen carefully in order to get the best estimate for the average power for each design:

- Lambda ($\lambda$) = 25 nm
- Channel Length (L) = 50 nm
- Minimum NMOS Transistor Width = 90 nm
- Minimum PMOS Transistor Width = 180 nm
• Power Supply Voltage = 1.2 V
• Input Pulse Rise time = 10 ps
• Input Pulse Fall time = 10 ps
• Pulse width (Input Pulse) = 10 ns
• Period (Input signal) = 20 ns
• \( f = \frac{1}{\text{Period}} = \frac{1}{20 \text{ ns}} = 50 \text{ MHz} \)
• A Fan-out-of-Four (FO4) load will be used at all outputs to mimic realistic conditions.

The parameter \( \lambda \) is the MOSIS design parameter, which relates directly to the technology used. We are using 50nm technology for our designs. It uses a 3 metal process with two polysilicon layers. For 50nm technology, \( \lambda = 25 \text{nm} \). All transistors in a particular technology use the same channel length.

5.2 Basic Logic Units

In this section, we will briefly discuss eight logical devices that will be designed and eventually integrated together to build our proposed partial product reduction stage. They are: Inverter, Fan-out-of-Four (FO4), AND_2, XOR_2, MUX, HA, FA, and 4:2_compressor.

5.2.1 CMOS Inverter

The inverter is the simplest and most popular CMOS circuit. Minimum size transistors are used in its design. A PMOS transistor width of 180nm, and an NMOS transistor width of 90nm is used in the design. The schematic view of the inverter in Electric VLSI is shown in Figure 5.1(a). The transistor parameters 3.6 and 2 shown in Figure 23 for the NMOS transistor refer to
the width and channel length of the transistor in units of $\lambda$. So for our chosen technology (50nm), they evaluate to 90nm and 50nm respectively. Similarly, the PMOS transistor width is 180nm. The PMOS transistor size is doubled to compensate for the mobility ratios of holes and electrons in these devices.

![Schematic Design of Inverter](image)

Figure 5.1(a). Schematic Design of Inverter

The layout design of the inverter will have the same transistor sizes as in the schematic design. The designed layout view is shown in Figure 5.1(b).
5.2.2. Fan-out-of-Four (FO4) Unit

The Fan-out-of-Four (FO4) is nothing but four inverters with their inputs tied together. To mimic a realistic environment during simulation, all logic devices are simulated with an FO4 load at its output. The schematic and layout design of FO4 load is shown in Figures 5.2(a) and 5.2(b) respectively.
Figure 5.2(a). FO4 Schematic Design in Electric VLSI

Figure 5.2(b). FO4 Layout Design in Electric VLSI
5.2.3. Two-input AND Gate

The two input AND gate is labeled as AND_2. It generates the logical AND of the two inputs A and B. The truth table of logical AND operation is shown in Table 5.1.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

We will use the conventional design for the AND_2 gate, which is simply a NAND unit with an inverter connected at its output node. Both schematic and layout design of the AND_2 gate are shown in Figures 5.3(a) and 5.3(b), respectively. The transistor sizing used is shown in the figure. The NMOS transistor sizes are doubled since there are two of them in series.
Figure 5.3(a). Schematic Design of the AND-2 Unit

Figure 5.3(b). Layout Design of the AND-2 Unit
5.2.4. Two-input XOR Gate

The XOR gate will be widely used in our designs, such as in the half adder, full adder, and the 4:2 compressor blocks. For our purpose, we will use the conventional design for the XOR_2 gate. Table 5.2, shows the truth table of the XOR operation. Both schematic and layout designs are shown in Figures 5.4(a) and 5.4(b) respectively. Once again the transistors are sized appropriately. When two transistors are connected in series, their widths will be doubled. This is done to provide the same current drive as in an inverter.

Table 5.2. Truth Table of XOR Operation

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 5.4(a). Schematic Design of the XOR_2 Unit

Figure 5.4(b). Layout Design of the XOR_2 Unit
5.2.5. 2x1 Multiplexer (MUX) Unit

The MUX unit takes thee inputs, A, B and Sel, such that if the Sel input is equal to 0, input A will be passed to the output. In contrast, if the Sel input is equal to 1, then input B will be passed to the output. Figure 5.5, shows the logic circuit for the MUX unit. The MUX unit will be used to design both full adder and 4:2 compressor. Furthermore, we will use the conventional design for the MUX unit. Figures 5.6(a) and 5.6(b), show the schematic and layout designs of the MUX unit, respectively.

Figure 5.5. Internal Circuit of MUX Unit
Figure 5.6(a). Schematic Design of the MUX Unit

Figure 5.6(b). Layout Design of the MUX Unit
5.2.6. Half Adder

The half adder adds two bits A and B, and produces two outputs, sum S and a carry C. The truth table for the half adder is shown in Table 5.3.

Table 5.3. Truth Table of the Half Adder

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Nonetheless, the half adder logic circuit consists of one XOR and an AND gate. The XOR gate will produce the sum output S, while the AND unit will produce the carry output C. Moreover, since we already designed both AND and XOR gates, we can combine them together to build a half adder. Figures 5.7(a) and 5.7(b), show the schematic and layout designs for the half adder.
Figure 5.7(a). Schematic Design of the Half Adder

Figure 5.7(b). Layout Design of the Half Adder
5.2.7. Full Adder

The full adder is designed in the same manner as the half adder, except that the full adder has a third input called carry-in (Cin), and it adds three bits A, B, and Cin, and produces two outputs, sum S and a carry-out Cout.

The internal circuit of the full adder can be designed in many ways. For example, the conventional full adder is designed using AND and OR gates. However, for our purposes, we will use a different design for the full adder, which is designed using two XOR gates and one MUX as shown in Figure 5.8(a). The reason behind choosing this design for our full adder is because it uses less number of transistors, which effectively reduces the power consumption of the entire adder unit. Figure 5.8(b) shows the layout design of the full adder.

![Figure 5.8(a). Schematic Design of the Full Adder](image)
5.2.8. 4:2 Compressor

The 4:2 compressor, also called 5:3 counter, is designed for the purpose of adding five bits to produce three outputs, sum S, and two carries Cout and Co. Figure 5.9(a), shows the design of a 4:2 compressor using four XOR gates and two MUXes. The previously designed XOR and MUX are used to build the 4:2 compressor. Figure 5.9(b), shows the layout design of the 4:2 compressor.
Figure 5.9(a). Logic Schematics of the 4:2 Compressor

Figure 5.9(b). Layout Design of the 4:2 Compressor
5.3 Simulation

Once the designs are completed for each individual logic device, the next step is to simulate and verify their performance. LTspice is used for simulation. Each logic device is loaded with an FO4 load to measure its performance under realistic conditions.

5.3.1. Inverter Simulation

Simulations were carried out for both schematic and layout designs. The resulting waveforms for the inverter is shown in Figure 5.10

![Figure 5.10. Input output waveforms of Inverter](image)

The schematic and layout design of the inverter with FO4 is shown in Figures 5.11(a) and 5.11(b), respectively. At this time, we are ready to measure the extracted CL of the inverter.
Figure 5.11(a). Inverter + FO4 Schematic Design in Electric VLSI

Figure 5.11(b). Inverter + FO4 Layout Design in Electric VLSI
5.3.1.1. Inverter Power Dissipation

By completing the layout design of the inverter with FO4, we can view the netlist of the design on LTspice. This can be done in Electric VLSI by choosing (Tools -> Simulation (Spice) -> Write Spice Deck). Note, in Electric VLSI, we must select ‘Conservative RC’ for the ‘Parasitic’ option in (File -> Preferences -> Tools -> Spice/CDL) window, in order to show the extracted capacitors and resistors for the layout design in the netlist file. The extracted netlist of the inverter with the FO4 is shown in Figure 5.12. Therefore, the extracted load capacitance for the inverter is \( C_L = 1.066 \text{ fF} \).

![Figure 5.12. Netlist of the inverter design with FO4](image)

For calculating the average power dissipation of the logic gate, which depends on load capacitance, our concern will be only in obtaining the value of extracted \( C_L \) at the output node of the logic gate after connecting the output with the FO4 load. Note that the extracted \( C_L \) is placed at the output node of the logic gate, which is to be tested, and also to the input gate of the FO4. Which is connected to the input gates of the four inverters. After we obtain the value of the
extracted C_L, we return to the schematic design of the inverter, with no FO4 connected to its output, and run Spice simulation to get the netlist of the inverter. Inside the netlist, we will add a load capacitor to the design with the same value of the extracted C_L that we have obtained from the netlist of the inverter with FO4 layout design, as shown in Figure 5.13.

![Figure 5.13. Netlist of the inverter design with extracted capacitance value](image)

The reason behind using the schematic view of the inverter and add to it the value of the extracted C_L is to only focus on the power on the effect of the load capacitance to get a ratio of the average power consumed by the circuit regardless of the extra power dissipated from the internal capacitors and resistors, which can be obtained from the extracted netlist of the layout design.

Therefore, the power consumption of the entire circuit can be measured in LTspice by showing the waveform of the following product, $V_{dd} \times I(V_{dd})$. This will show the waveform of the power consumed by the inverter. In addition, in LTspice, by pressing the ‘Ctrl’ button and
clicking on the label of the waveform, which says $V(v_{dd} \cdot I(V_{dd}))$, at the same time, we can view the average power consumption of the inverter between the intervals 0 to 1000 ns, as shown in Figure 5.14. In this case, the average power dissipation of the inverter unit is almost 151 nW.

![Figure 5.14. Average power dissipation of the inverter in LTspice](image)

5.4. The Effect of the 4:2 Compressor’s Inputs on Power Dissipation

Our main goal to test and verify the effect of each input of the 4:2 compressor on the power dissipation of the entire 4:2 compressor unit. By doing so, we will tabulate the five inputs of the 4:2 compressor from the highest effect on the power dissipation to the lowest, in which it will verify our conclusion from our theoretical approach in Section 4.

The testing technique that we will use for testing all inputs is explained as follows. Since, the 4:2 compressor unit has five inputs (X1 X2 X3 X4 Cin), we will test each input individually by feeding the input to be tested a pulse signal that has 50 MHz, according to our Design
Specifications, while feeding the other four inputs a constant DC voltage. For example, if we are to test the effect of the input X1 on the power dissipation of the entire 4:2 compressor, we will feed it a pulse signal with a frequency of 50 MHz, while feeding the other four inputs, X2, X3, X4, and Cin a constant DC voltage of 0, in which the input sets will be (X1 0 0 0 0). Thus, all five inputs of the 4:2 compressor will have a constant 0 DC voltage, except the input X1, which has a pulse signal. In other words, by changing the value of X1 only, the three outputs from the 4:2 compressor, S, Cout, and Co, will change according to X1. In addition, by measuring the average power dissipation of entire 4:2 compressor unit, while only changing the X1 value, we can predict the effect of X1 on power dissipation of the entire 4:2 compressor unit. Nonetheless, since the 4:2 compressor unit will have four inputs with constant voltages, then to cover all cases we need 2 to the power of 4 different cases, which is equal to 16 different cases. The 16 cases will range from (X1 0 0 0 0), (X1 0 0 0 1), (X1 0 0 1 0), ..., (X1 1 1 1 1). Therefore, we will test all 16 different cases while measuring the power consumption for each case. The power dissipation results of all different cases for the input X1 are shown in Table 5.1. After we obtains all 16 different values from all cases, we take the average number, which will be the final power consumption produced from the 4:2 compressor due to the change of X1. Similarly, we perform the same procedures for the inputs X2, X3, X4, and Cin, respectively. After obtaining all average values for the power consumption of the 4:2 compressor unit due to each input change, see Table 5.2, thus, we can order the inputs according to their effect on the power dissipation of the 4:2 compressor unit from largest to smallest, in which the order will be X1, X2, X3, X4, and Cin.
Table 5.1. Effect of X1 on power dissipation of the 4:2 Compressor

<table>
<thead>
<tr>
<th>Testing Input</th>
<th>Inputs</th>
<th>Power Dissipation (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cin X4 X3 X2 X1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0</td>
<td></td>
<td>3.39</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td></td>
<td>3.32</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td></td>
<td>3.62</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td></td>
<td>3.27</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td></td>
<td>3.76</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td></td>
<td>3.74</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td></td>
<td>4.00</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td></td>
<td>3.66</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td></td>
<td>3.80</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td></td>
<td>3.70</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td></td>
<td>4.04</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td></td>
<td>3.67</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td></td>
<td>3.45</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td></td>
<td>3.41</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td></td>
<td>3.75</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td></td>
<td>3.34</td>
</tr>
<tr>
<td>Average Power</td>
<td></td>
<td>3.62</td>
</tr>
</tbody>
</table>

Table 5.2. Summary of the Power Dissipation of the 4:2 Compressor due to each input

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Average Power dissipation (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>3.62</td>
</tr>
<tr>
<td>X2</td>
<td>3.45</td>
</tr>
<tr>
<td>X3</td>
<td>3.23</td>
</tr>
<tr>
<td>X4</td>
<td>3.16</td>
</tr>
<tr>
<td>Cin</td>
<td>1.34</td>
</tr>
</tbody>
</table>

5.5. Effect of the Full Adder’s Inputs on Power Dissipation

For testing the effect of the three inputs of the full adder unit individually, we will follow the same technique we have used for the inputs of the 4:2 compressor unit. However, when testing the first input, A, we are only left with two inputs, B and Cin. Therefore, we only need four different cases to test the effect of input A on the full adder unit. Those cases are, (A 0 0), (A 0 1), (A 1 0), and (A 1 1). In the same manner, we repeat the procedure for the inputs B and Cin.
Tables 5.3, shows the results of the power dissipation of the full adder unit due to input A, where the summarized results for the effect of all inputs are shown in Table 5.4, we conclude that the effect on the power dissipation of the full adder unit from largest to smallest, in which the order will be A, B, and Cin.

**Table 5.3. Effect of A on power dissipation of the Full Adder Unit**

<table>
<thead>
<tr>
<th>Testing Input</th>
<th>Inputs</th>
<th>Power Dissipation (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cin</td>
<td>B</td>
</tr>
<tr>
<td>A</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Average Power</td>
<td></td>
</tr>
</tbody>
</table>

**Table 5.4. Summary of the Power Dissipation of the Full Adder Unit due to each input**

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Average Power dissipation (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2.26</td>
</tr>
<tr>
<td>B</td>
<td>1.90</td>
</tr>
<tr>
<td>Cin</td>
<td>1.10</td>
</tr>
</tbody>
</table>

**5.6. Conclusion of Simulation Approach**

In the simulation part, we have measured the effect of each input on the power dissipation of both the 4:2 compressor and the full adder units. This have been accomplished by feeding the input to be tested a pulse signal that has a frequency of 50 MHz, while feeding the remaining inputs a constant DC voltage values. For the 4:2 compressor unit, each input is tested against the 16 different cases that correspond to the remaining four inputs. After tabulating the results for the power consumption of the 4:2 compressor due to each input, see Table 5.2, we concluded that
the order of the inputs of the 4:2 compressor from the largest power dissipation to the lowest are, X1, X2, X3, X4, Cin. Similarly, for the full adder unit, we test each input individually to obtain the effect of each input on the power dissipation of the full adder unit. After obtaining the results of all power dissipation due to all inputs, we conclude that the order of the inputs of the full adder unit from the largest power dissipation to the lowest are, A, B, and Cin.

Based on this information, we can make the best decision for the interconnection between the outputs from the first reduction stage and the inputs of the second reduction stage, where the output with the highest probability is connected to the input with the smallest total effective capacitance. This will be concluded in the following example.

5.7. Simulation Example

For verification purposes, we will repeat the same example that we have done in the theoretical example in section 4.4, where we will be connecting the outputs of two 4:2 compressors in the first reduction stage to the inputs of one 4:2 compressor in the second reduction stage. Next, we will feed all inputs with the same pulse signal that has a frequency of 50 MHz, but with different initial delays. After that, we test the design and measure the average power dissipation of the entire design. Moreover, we will compare the resulted power dissipation of the three 4:2 compressor units using our proposed interconnection against the same design but with different interconnection. This way, we can distinguish the effectiveness of our proposed interconnection from the others by observing the overall power dissipation of the two designs. For our purpose, we will compare the our proposed interconnection for the 4:2 compressor against Nageshwar’s pattern interconnection for the 4:2 compressor from [6], which orders the inputs of the 4:2 compressor according to their effective input capacitances. The order of the
inputs from highest to lowest is, X1, X3, X4, Cin, X2. While in our case, the order of the inputs of the 4:2 compressor depends on the total effective input capacitances, and the order from highest to lowest according to Table 4.3 is, X1, X2, X3, X4, and Cin. The design of the three 4:2 compressor using our proposed interconnection is shown in Figure 5.15, while Nageshwar’s interconnection is shown in Figure 5.16. Finally, the resulted power dissipation from both design are shown in Table 5.5.

Table 5.5. Power Dissipation for the proposed interconnection for three 4:2 Compressors

<table>
<thead>
<tr>
<th>Design</th>
<th>Average Power dissipation (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>90.73</td>
</tr>
<tr>
<td>Nageshwar’s design</td>
<td>94.75</td>
</tr>
<tr>
<td>Power Reduction (%)</td>
<td>5 %</td>
</tr>
</tbody>
</table>

Figure 5.15. Proposed interconnection for the 4:2 Compressor Unit
Figure 5.16. Nageshwar’s Proposed interconnection for the 4:2 Compressor Units
Chapter 6

Results

We formulated a set of new rules, presented in Section 4.3, for the interconnection patterns for different adder modules in the partial product reduction stage of a multiplier. Based on these rules we designed the partial product reduction stage for the 16×16 Multiplier, presented in Section 5.

The complete design includes 26 4:2 compressors, 34 full adders, and 13 half adders. We implemented five different designs for the interconnection between the blocks of the partial product reduction stage of the 16x16 Multiplier. The first one is based on our proposed interconnection pattern. The second one is based on Nageshwar’s design for the partial product reduction stage of the 16x16 Multiplier [6]. The remaining ones used random interconnection patterns.

Note that the randomness of the interconnection in the partial product reduction stage will have a variation in their effects on the power consumption of the entire design. These designs with random interconnection patterns were made without paying any attention to their effective capacitance. The different designs are shown in Figures 6.1, 6.2 and 6.3. Figure 6.1 shows our proposed design. Figure 6.2 shows the design based on Nageshwar’s technique. Figure 6.3 shows a randomly interconnected design, however, only one of the random designs is shown here.
We simulated all five designs to estimate their power consumption. The results obtained are shown in Table 6.1. All simulations were carried out at 1GHz frequency. The average power dissipation varied from 1.49mW to 1.76mW. Our design showed the lowest power dissipation of 1.49mW. Nageshwar’s design had 1.62mW. One of the randomly interconnected design consumed only 1.6mW of power, while others had 1.65mW and 1.76mW of power respectively. Hence, the overall power reduction ranges between 7-15% for our design compared to others. This validates our claim of low power interconnection strategy for the partial product reduction stage for a 16×16 Multiplier.
Figure 6.2. Nageshwar’s Proposed Interconnection for the Partial Product Reduction Stages
Figure 6.3. Random Interconnection for the Partial Product Reduction Stages

Table 6.1. Power Dissipation of the Partial Product Reduction Stage

<table>
<thead>
<tr>
<th>Design</th>
<th>Average Power dissipation (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposed</td>
<td>1.49</td>
</tr>
<tr>
<td>Nageshwar’s design</td>
<td>1.62</td>
</tr>
<tr>
<td>Random Interconnection 1</td>
<td>1.60</td>
</tr>
<tr>
<td>Random Interconnection 2</td>
<td>1.65</td>
</tr>
<tr>
<td>Random Interconnection 3</td>
<td>1.76</td>
</tr>
<tr>
<td>Power Reduction (%)</td>
<td>7-15 %</td>
</tr>
</tbody>
</table>
Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this research, we have studied and analyzed the interconnection pattern for the partial product reduction stage of a 16x16 Multiplier, solely for the purpose of reducing the power consumption in the partial product reduction stage. We focused on the partial product reduction stage because it is the dominant power consuming module with in the multiplier. Large power dissipation occurs in this module because of the very large number of adder stages used in it.

Our approach for achieving the best pattern for the interconnection between the half adder, full adder, and 4:2 compressors inside the partial product reduction stage of the 16x16 Multiplier was to study the behavior of the inputs and outputs in each block and their effects on power dissipation. We used both switching probabilities and input capacitance estimates to determine the effect of each input on the power dissipation of a particular logic block. However, since the half adder block has a symmetrical structure for its two inputs, its interconnection pattern does not affect power. On the other hand, the full adder and the 4:2 compressor blocks have asymmetric structure for their inputs and outputs, which resulted in different values for the switching probabilities and their effective capacitances. Thus, by a selective ordering of the effective capacitance of each input in the full adder and 4:2 compressor blocks, from highest to lowest, we derived the best pattern for interconnection between the logic blocks that will result in minimum power dissipation for the partial product reduction stage of the 16x16 Multiplier.

Five different designs were implemented using Electric VLSI tools. We used 50nm CMOS technology for all our designs. Out of the five designs, three of them were based on random interconnection patterns. One was based on a recent work by using the effective capacitance
approach. All five designs were simulated at a frequency of 1GHz using LTspice. Our design used the lowest power of 1.49mW. The highest power was consumed by one of the randomly interconnected designs, and it dissipated 1.76mW. Hence, the power reduction for our design varied between 7-15% compared to the others.

7.2. Future Work

Our research was an extension of other similar works that have been done on reducing the power dissipation of a 16x16 Multiplier. However, our main focus was only on the partial product reduction stage, which is the part in the multiplier that has the highest power dissipation because of the large adder stages used in it. Our design included three randomly interconnected partial product reduction stages. They were designed by mixing the wires that connect each reduction stage with the next one. Therefore, a large variation is found in the final value of the power dissipation of the three random cases. Further research could extend it by finding all different interconnection patterns and use each pattern for the entire design. This will further verify and order all interconnection patterns from high to low based on their power dissipation.

Furthermore, since we are only focusing on the partial product reduction stage, the research can be further extended to study the behavior of other parts of the multiplier, such as the design of Booth encoder and the last carry propagate adder stage, and reduce their power dissipation.
References


Appendix A

Probability derivations for Logic Modules

The essence of calculating the probabilities for logic modules comes into a play when analyzing the signals and the expected values for the logic gate to be logic 1. Our approach in calculating the probabilities is focused on analyzing the static input probabilities that are entering the partial product reduction stage. All inputs, however, will have a probability of 0.25 since they are produced by AND gates, which only give an output of logic 1 whenever both input are equal to logic 1, before they enter the partial product stage. Based on that, we calculate the static output probabilities for three logic gates, half adder, full adder, and 4:2 compressor, as follows:

A.1. Half Adder Static Probabilities

The sum and carry out expressions are given by:

\[ S = A \oplus B, \text{ and} \]

\[ \text{Cout} = A \cdot B \]

Thus, to calculate the probability of each output, we replace the each input with its static probability value and then perform the required operation according to each equation.

\[ Pr(S) = Pr(A') \cdot Pr(B) + Pr(A) \cdot Pr(B'), \text{ and} \]

\[ Pr(\text{Cout}) = Pr(A) \cdot Pr(B) - Pr(A) + Pr(b) \]

A.2. Half Adder Switching Probabilities

To obtain the switching probability for the outputs of the half adder, S and Cout, we simply multiply the resulted static probability by (1 - the static probability), which is illustrated below:
Pr(S)sw = Pr(s) * (1 – Pr(S)), and
Pr(Cout)sw = Pr(Cout) * (1 – Pr(Cout))

A.3. Full Adder Static Probabilities

The sum and carry out expression of the full adder are given by:

\[ S = A \oplus B \oplus C, \quad \text{and} \]
\[ \text{Cout} = (A \oplus B).C + A.B \]

Thus, the static probabilities for the outputs of the full adder are given by:

\[ \text{Pr}(S) = \Pr[(A \oplus B) \cdot \overline{C} + (A \oplus B) \cdot C], \quad \text{and} \]
\[ \text{Pr}(\text{Cout}) = \Pr[(A \oplus B).C + A.B] \]

A.4. Full Adder Switching Probabilities

Similar to the half adder, we can obtain the switching probability for the full adder outputs by performing the following computation:

\[ \text{Pr}(S)sw = \Pr(s) * (1 – \Pr(S)), \quad \text{and} \]
\[ \text{Pr}(\text{Cout})sw = \Pr(\text{Cout}) * (1 – \Pr(\text{Cout})) \]

A.5. 4:2Compressor Static Probabilities

The sum and carry out expression of the full adder are given by:

\[ S = A \oplus B \oplus C, \quad \text{and} \]
\[ \text{Cout} = (A \oplus B).C + A.B \]

Thus, the static probabilities for the outputs of the full adder are given by:

\[ \text{Pr}(S) = \Pr[X1 \oplus X2 \oplus X3 \oplus X4 \oplus Cin], \]
Pr(Cout) = Pr\([(X1 \oplus X2).X3 + (X1 \oplus X2).X1]\], and

Pr(Co) = Pr\([(X1 \oplus X2 \oplus X3 \oplus X4).Cin + (X1 \oplus X2 \oplus X3 \oplus X4).X4]\]

**A.6. 4:2 Compressor Switching Probabilities**

Similar to the half adder, we can obtain the switching probability for the full adder outputs by performing the following computation:

\[Pr(S)_{sw} = Pr(s) \times (1 - Pr(S))\], and

\[Pr(Cout)_{sw} = Pr(Cout) \times (1 - Pr(Cout))\]

\[Pr(Co)_{sw} = Pr(Co) \times (1 - Pr(Cout))\]

**A.7. Probability calculation for the partial product reduction stage of the 16x16 Multiplier**

The values for all probabilities that are calculated using the above equations are shown in Figure A.1. The figure, however, illustrates the probability values for each input inside the partial product reduction stage.
Figure A.1. Complete probabilities for the partial product reduction stage of the 16x16 Multiplier

APPENDIX B
Boolean Difference for Logic Modules

B1. Boolean Difference

Boolean Difference is a technique that differentiates a function with regard to a single variable. This allows us to study and calculate the change that occurs to the function with respect to the selected variable $x_i$.

Definition 1: Let $f(x) = f(x_1, \ldots, x_i, \ldots, x_n)$ be a logic function of $n$ variables, then

The Boolean difference $\frac{\partial f}{\partial x_i}$ is defined as: $\frac{\partial f}{\partial x_i} = f(x_1, \ldots, x_i, \ldots, x_n) \oplus f(x_1, \ldots, \overline{x_i}, \ldots, x_n)$. It is obvious that $\frac{\partial f}{\partial x_i} = 1$ if and only if $f(x_1, \ldots, x_i, \ldots, x_n) = \overline{f(x_1, \ldots, \overline{x_i}, \ldots, x_n)}$. This allows us to study the change occurring in $f$ due to a change in $x_i$. $\frac{\partial f}{\partial x_i} = 1$, implies that for every transition in $x_i$, there will be a corresponding transition in $f$.

For example, if we want to study when a change occurs on the following AND function $f = X_1 \cdot X_2$ with a change in $X_1$, then we can find the Boolean difference

$\frac{\partial f}{\partial X_1}$.

Now $\frac{\partial f}{\partial X_1} = [X_1 X_2]_{X_1=0} \oplus [X_1 X_2]_{X_1=1} = 0 \oplus X_2 = X_2$. Hence, $f$ makes a transition every time $X_1$ makes a transition if $X_2 = 1$.

The Boolean difference can be effectively used to find the switching probability of $f$ with respect
to \( X_1 \), which is \( P\left( \frac{\partial f}{\partial X_1} \right) \). The switching probability (transition probability) of \( X_1 \) is given by:

\[
TP(X_1) = P(X_1)P(\bar{X}_1).
\]

Therefore, the total switching probability of due to \( X_1 \) is: \( TP(X_1)P\left( \frac{\partial f}{\partial X_1} \right) \), and for the case of AND gate above, simplifies to \( TP(X_1)P(X_2) \).

**B.2. Half Adder Boolean Difference**

A half adder with inputs \( A \) and \( B \) and outputs \( S \) and \( C_{OUT} \) are shown in Figure B1.

![Figure B1. Half Adder Logic Diagram.](image)

The sum and carry out expressions are given by:

\[
S = A \oplus B \quad \text{and} \quad C_{OUT} = A \cdot B
\]

**B.2.1. The Boolean difference of \( S \):**

\[
\frac{dS}{dA} = (\bar{A} \cdot B + A \cdot \bar{B})_{A=0} \oplus (\bar{A} \cdot B + A \cdot \bar{B})_{A=1} = B \oplus \bar{B} = 1,
\]

Similarly \( \frac{dS}{dB} = 1 \).
B.2.2. Boolean Difference of Cout:

\[ \frac{dC_{OUT}}{dA} = (AB)_{A=0} \oplus (AB)_{A=1} = 0 \oplus B = B, \text{ and } \]

\[ \frac{dC_{OUT}}{dB} = A \]

B.3. Full Adder Boolean Difference

The full adder logic diagram is shown in Figure B2.

![Figure B2. Full Adder Logic Diagram.](image)

The logic equations for sum and carry are given by

\[ S = A \oplus B \oplus C_{IN}, \text{ and } \]

\[ C_{OUT} = AB + AC + BC \]

B.3.1. The Boolean difference of S:

\[ \frac{dS}{dA} = (A \oplus B \oplus C)_{A=0} \oplus (A \oplus B \oplus C)_{A=1} = \overline{B \oplus C} \oplus \overline{B \oplus C} = 1. \]
Similarly, $\frac{dS}{dB} = 1$, and $\frac{dS}{dC_{IN}} = 1$

### B.3.2. The Boolean difference of Cout:

$$\frac{dC_{OUT}}{dA} = (AB + BC + AC)_{A=0} \oplus (AB + BC + AC)_{A=1} = (BC) \oplus (B + C) = BC \oplus (B + C) + BC \overline{B + C}$$

$$= (B + C)(B + C) + BC \overline{B + C} = B \oplus C$$

Similarly, $\frac{dC_{OUT}}{dB} = A \oplus C$, and $\frac{dC_{OUT}}{dC_{IN}} = A \oplus B$

### B.4. 4:2 Compressor Boolean Difference

The Boolean equations for the 4:2 compressor is shown in Figure B.3

![Figure B3. Full Adder Logic Diagram.](image)

The Boolean expressions are given by:

$$S = X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{IN}$$
\[ C_o = (X_1 \oplus X_2 \oplus X_3 \oplus X_4)C_{IN} + \overline{X_1 \oplus X_2 \oplus X_3 \oplus X_4} \times X_4, \text{ and} \]
\[ C_{OUT} = (X_1 \oplus X_2)X_3 + \overline{(X_1 \oplus X_2)} \times X_1 \]

**B.4.1. The Boolean difference of S:**

\[ \frac{dS}{dX_1} = (X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{IN})_{X_1=0} \oplus (X_1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{IN})_{X_1=1} \]
\[ = (X_2 \oplus X_3 \oplus X_4 \oplus C_{IN}) \oplus \overline{(X_2 \oplus X_3 \oplus X_4 \oplus C_{IN})} = 1, \]

Similarly, \[ \frac{dS}{dX_2} = \frac{dS}{dX_3} = \frac{dS}{dX_4} = \frac{dS}{dC_{IN}} = 1 \]

**B.4.2. The Boolean difference of Cout:**

\[ \frac{\partial C_{out}}{\partial X_1} = [(X_1 \oplus X_2).X_1 + (X_1 \oplus X_2).X_3]_{X_1=0} \oplus [(X_1 \oplus X_2).X_1 + (X_1 \oplus X_2).X_3]_{X_1=1} \]
\[ = [(0 \oplus X_2).0 + (0 \oplus \overline{X_2}).X_3] \oplus [(1 \oplus X_2).1 + (1 \oplus \overline{X_2}).X_3] \]
\[ = [X_2.X_3] \oplus [X_2 + \overline{X_2}.X_3] \]
\[ = [X_2.X_3] \oplus [X_2 + X_3] \]
\[ = X_2 \oplus X_3 \]
\[ = \overline{X_2}.X_3 + X_2.\overline{X_3} \]

\[ \frac{\partial C_{out}}{\partial X_2} = [(X_1 \oplus X_2).X_1 + (X_1 \oplus X_2).X_3]_{X_2=0} \oplus [(X_1 \oplus X_2).X_1 + (X_1 \oplus X_2).X_3]_{X_2=1} \]
\[ = [(X_1 \oplus 0).X_1 + (X_1 \oplus 0).X_3] \oplus [(X_1 \oplus 1).X_1 + (X_1 \oplus 1).X_3] \]
\[ = [X_1 + \overline{X_1}.X_3] \oplus [\overline{X_1}.X_1 + X_1.X_3] \]
\[ = [X_1 + X_3] \oplus [X_1.X_3] \]
\[ = X_1 \oplus X_3 \]
\[ = \overline{X_1}.X_3 + X_1.\overline{X_3} \]
\[
\frac{\partial C_{\text{out}}}{\partial X_3} = [ (X_1 \oplus X_2).X_1 + (\overline{X_1} \oplus X_2).X_3]_{X_3=0} \oplus [(X_1 \oplus X_2).X_1 + (\overline{X_1} \oplus X_2).X_3]_{X_3=1}
\]

\[
= [(X_1 \oplus X_2).X_1 + (\overline{X_1} \oplus X_2).0] \oplus [(X_1 \oplus X_2).X_1 + (\overline{X_1} \oplus X_2).1]
\]

\[
= [(X_1 \oplus X_2).X_1] \oplus [(X_1 \oplus X_2).X_1 + (\overline{X_1} \oplus X_2)]
\]

\[
= [(X_1 \oplus X_2).X_1] \oplus [X_1 + (\overline{X_1} \oplus X_2)]
\]

\[
= X_1 \oplus X_2
\]

\[
= X_1 \cdot \overline{X_2} + X_1 \cdot \overline{X_2}
\]

**B.4.3. The Boolean difference of Co:**

Before we apply the Boolean Difference on the function Co, we will take X1 out of the XOR and XNOR functions in Equation #, as shown below:

**Probability of Co due to X1, X2, X3, :**

\[
C_o = [X_1 \cdot (X_2 \oplus X_3 \oplus X_4) \cdot X_4 + X_1 \cdot (X_2 \oplus X_3 \oplus X_4) \cdot X_4] + [X_1 \cdot (X_2 \oplus X_3 \oplus X_4) \cdot C_{in} + X_1 \cdot (X_2 \oplus X_3 \oplus X_4) \cdot C_{in}]
\]

This will make the equation simpler in order to test for when \( X_1 = 0 \) and when \( X_1 = 1 \). Thus, by applying the Boolean Difference now, we get the following equation:

\[
\frac{\partial C_o}{\partial X_1} = [(X_2 \oplus X_3 \oplus X_4) \cdot X_4 + (X_2 \oplus X_3 \oplus X_4) \cdot C_{in}] \oplus [(X_2 \oplus X_3 \oplus X_4) \cdot X_4 + (X_2 \oplus X_3 \oplus X_4) \cdot C_{in}]
\]

For simplification, we will make \( A = (X_2 \oplus X_3 \oplus X_4) \), and \( \overline{A} = (X_2 \oplus X_3 \oplus X_4) \). Now, we can neatly re-write the equation as shown below:

\[
\frac{\partial C_o}{\partial X_1} = [\overline{A} \cdot X_4 + A \cdot C_{in}] \oplus [A \cdot X_4 + \overline{A} \cdot C_{in}]
\]

Thus,

\[
\frac{\partial C_o}{\partial X_1} = [\overline{A} \cdot X_4 + A \cdot C_{in}] \cdot [A \cdot X_4 + \overline{A} \cdot C_{in}] + [\overline{A} \cdot X_4 + A \cdot C_{in}] \cdot [A \cdot X_4 + \overline{A} \cdot C_{in}]
\]
\[= (A + X4). (\bar{A} + \bar{C}_{m}). (A. X4 + \bar{A}. C_{in}) + (\bar{A}. X4 + A. C_{in}). (\bar{A} + X4). (A + \bar{C}_{m}).\]

\[= (A. \bar{C}_{in} + \bar{A}. X4 + X4. \bar{C}_{in}). (A. X4 + \bar{A}. C_{in}) + (\bar{A}. X4 + A. C_{in}). (\bar{A}. \bar{C}_{m} + A. X4 + X4. \bar{C}_{in})\]

\[= A. X4. \bar{C}_{in} + \bar{A}. X4. C_{in} + \bar{A}. X4. \bar{C}_{in} + A. \bar{X}4. C_{in}\]

Taking \(C_{in}\) and \(\bar{C}_{in}\) as a common factor, gives the following:

\[
\frac{\partial C_o}{\partial X1} = C_{in}. [X4. (\bar{A} + A)] + \bar{C}_{in}. [X4. (\bar{A}. A)]
\]

\[= C_{in}. X4 + X4. \bar{C}_{in}\]

\[= X4 \oplus C_{in}\]

Now, we replace \(\bar{A}\) with its original value \((X2 \oplus X3 \oplus X4)\), we get:

\[
\frac{\partial C_o}{\partial X1} = C_{in} + (X2 \oplus X3 \oplus X4). X4
\]

\[= C_{in} + [X4. X4. (X2 \oplus X3) + [X4. X4. (X2 \oplus X3)]\]

\[= C_{in} + X4. (X2 \oplus X3)\]

We observe that the effect of \(X1, X2,\) and \(X3\) on the function \(C_o\) is the same for all three inputs. Hence, we can say that the probability of \(C_o\) due to \(X1\), is equivalent to the probability of \(C_o\) due to \(X2\) and \(X3\):

\[P(\text{Cout due to } X1) = P(\text{Cout due to } X2) = P(\text{Cout due to } X3) = 0.08 = 8\%\]
**Probability of Co due to X4:**

Before we apply the Boolean Difference on the function Co, we will take X4 out of the XOR and XNOR functions in Equation #, as shown below:

\[
C_o = [\overline{X}4. X4. (X1 \oplus X2 \oplus X3) + X4. X4. (X1 \oplus X2 \oplus X3)] + [\overline{X}4. C_{in}. (X1 \oplus X2 \oplus X3)
+ X4. C_{in}. (X1 \oplus X2 \oplus X3)]
\]

\[
\frac{\partial C_o}{\partial X4} = [C_{in}. (X1 \oplus X2 \oplus X3)] \oplus [(X1 \oplus X2 \oplus X3) + C_{in}. (\overline{X}1 \oplus \overline{X}2 \oplus \overline{X}3)]
\]

Hence,

\[
\frac{\partial C_o}{\partial X4} = X1 \oplus X2 \oplus X3 \oplus C_{in}
\]