LOW POWER MULTIPLIER DESIGN USING GATE DIFFUSION INPUT CMOS LOGIC

Thesis report submitted towards the partial fulfillment of

requirements for the award of the degree of

Master of Technology

In

VLSI Design & CAD

Submitted by

Amit Jindal

Roll No. 600861004

Under the Guidance of

Ms. Sakshi

Assistant Professor, ECED

Department of Electronics and Communication Engineering

THAPAR UNIVERSITY

PATIALA-147004, INDIA

JULY - 2010

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, **"Low Power Multiplier Design using Gate Diffusion Input CMOS Logic"** in partial fulfillment of the requirement for the award of degree of M.Tech (VLSI Design & CAD) at Electronic and Communication Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms Sakshi, Assistant Professor, ECED.

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

Date: 30-6-10

Amit Jindal
Amit Jindal
Roll No. 600861004

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

Sakshi
Ms. Sakshi
Assistant Professor
ECED, Thapar University

Counter signed by:

13.7.10
(Dr. A. K. Chatterjee)
Professor and Head
Electronics and Communication Engineering Department,
Thapar University, Patiala

13.7.10
(Dr. R. K. Sharma)
Dean (Academic Affairs)
Thapar University,
Patiala.

ACKNOWLEDGMENT

The quest for knowledge is a journey that is long and difficult but equally rewarding. It is a journey not many people would endeavour for fear of failure. That is why it is a necessity to have strong support from the people around you to make this journey a success. Without the support from many people, I would not have completed my graduate work. It is pleasant aspect that I have now the opportunity to express my gratitude for all of them.

I would like to thank *Ms. Sakshi*, Assistant Professor, my thesis advisor and mentor at Thapar University, under whose inspiration, encouragement and guidance I have successfully completed this thesis work. She let me work on my thesis in complete freedom while strongly supporting my academic endeavours, no matter where they took me. I would like to thank her for introducing me to the problem and providing invaluable advice throughout the course of the work. I truly admire her perseverance, depth of knowledge and strong dedication to students and research that has made her one of the most successful professors ever. Her mastery at any topic is amazing, but yet she is such a humble and down-to-earth person. I'm glad that I was given the opportunity to work with her. She brings out the best in her students and I'd like to thank her for all the support, encouragement and guidance given to me during my graduate years. Any student should consider himself or herself extremely fortunate to find a gem of an advisor like Sakshi Madam.

Next, I wish to express my gratitude to *Professor (Dr.) A. K. Chatterjee*, Head, Department of Electronics and Communication Engineering and *Ms. Alpana Agarwal*, P.G. Coordinator, for their excellent guidance and encouragement right from the beginning of this course. I am especially indebted to them for critical reviewing of my compilation to bring it in the submitted presentable form.

I am also grateful for their trust in the choices I made to complete this thesis work.

I am also thankful to all the faculty and staff members of ECED for providing me all the facilities required for the completion of this thesis work. It has a pleasure working at Thapar University and this is mostly due to the wonderful people who have sojourned there over the past years.

I am also thankful to all my friends especially Vaibhav, Deependra, Gopal , Hem, Simran and Venus who were always there at the need of hour and provided me motivation and inspiration in my tough times.

Next, I'd like to thank *Mr. B. K. Hemant* in charge VLSI Design Lab for making available access to the tools and lab at any time we required. Next I would like to thank all my friends. All the good times at the lab and there help, criticisms, suggestions, and friendship which makes everyday a pleasant one. Thanks so much to all of you for the fun, frolic and great memories here at T.U.

Finally, and above everyone else, I would like to thank *My Family* for standing by me through all the joys and sorrows that life had to offer. My heartfelt thanks and life-long gratitude go to my *Dearest Mother* and my *Loving Father* for all the love and affection that they have showered upon me. *You both* are the *Best and Most Loving Parents* that anyone can hope to have in this entire universe. If not for your constant support, encouragement and sacrifices I would never have made it to this stage in life.

My acknowledgements would not be complete without expressing my gratitude towards *Almighty God*. I feel very fortunate to come to know *Him* during all these years of my life and have continually been blessed by *His endless love* ever since. He is the true shepherd of my life.

Amit Jindal

ABSTRACT

The main objective of this thesis is to provide new low power solutions for Very Large Scale Integration (VLSI) designers. Especially, this work focuses on the reduction of the power dissipation, which is showing an ever-increasing growth with the scaling down of the technologies. Various techniques at the different levels of the design process have been implemented to reduce the power dissipation at the circuit, architectural and system level.

Furthermore, the number of gates per chip area is constantly increasing, while the gate switching energy does not decrease at the same rate, so the power dissipation rises and heat removal becomes more difficult and expensive. Then, to limit the power dissipation, alternative solutions at each level of abstraction are used.

The dynamic power requirement of CMOS circuits is rapidly becoming a major concern in the design of personal information systems and large computers. In this thesis work, a new CMOS logic style called *Gate Diffusion Input* is discussed (GDI). GDI basic cell is very similar to basic CMOS logic style but it enables us to design complex functions with fewer gates as compared to CMOS, which further reduces the power consumption of the circuit.

Multiplier is one of the most commonly used circuits in the digital devices. Multiplication is one of the basic functions used in digital signal processing. Most high performance DSP systems rely on hardware multiplication to achieve high data throughput. There are various types of multipliers available depending upon the application in which they are used. Full Adder is the main block of power dissipation in multiplier. So reducing the power dissipation of full adder ultimately reduces the power dissipation of multiplier. Dynamic component of power is reduced in GDI techniques as source of pMOS is not permanently connected to Vdd. It also reduces the latency of the circuit.

To my wonderful & beloved

Parents

for all the love they tried to give me

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

CMOS	Complementary Metal Oxide Semiconductor
CPTL	Complementary Pass Transistor Logic
CVSL	Cascaded Voltage Swing Logic
DCVS	Differential Cascode Voltage Swing
DRC	Design Rule Check
GDI	Gate Diffusion Input
IC	Integrated Circuit
NMOS	N-type Metal Oxide Semiconductor
PTL	Pass Transistor Logic
PMOS	P-type Metal Oxide Semiconductor
RC	Resistance Capacitor
TGL	Transmission Gate Logic
VLSI	Very Large Scale Integration

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Arithmetic circuits, like adders and multipliers, are one of the basic components in the design of communication circuits. Recently, an overwhelming interest has been seen in the problems of designing digital systems for communication systems and digital signal processing with low power at no performance penalty. Designing low power high-speed arithmetic circuits requires a combination of techniques at four levels; algorithm, architecture, circuit and system levels. This thesis presents layout and simulations of a multiplication algorithm, which is suitable for high-performance and low–power applications [2]. Digital multipliers are the most commonly used components in many digital circuit designs. They are fast, reliable and efficient components that are utilized to implement any operation. Depending upon the arrangement of the components, there are different types of multipliers available. Particular multiplier architecture is chosen based on the application. The power dissipation in a multiplier is a very important issue as it reflects the total power dissipated by the circuit and hence affects the performance of the device.

Most digital signal processing (DSP) systems incorporate a multiplication unit to implement algorithms such as correlations, convolution, and filtering and frequency analysis. In many DSP algorithms, the multiplier lies in the critical delay path and ultimately determines the performance of the algorithm. The speed of multiplication operation is of great importance in DSP as well as in the general

processors today, especially since the media processing took off. In the past, multiplication was implemented generally with a sequence of addition, subtraction and shift operations. Recently, many multiplication algorithms have been invented and developed, each having pros and cons in different fields. The multiplier is a fairly large block of a computing system. The amount of circuitry involved is proportional to the square of its resolution; i.e. a multiplier of size n bits has O (n^2) gates [1]. For multiplication algorithms performed in DSP applications, latency and throughput are the two major constraints from delay perspective. Latency is the real delay of computing a function, a measure of how long after the inputs to a device are stable, is the final result available on outputs. Throughput is the measure of how many multiplications can be performed in a given period of time. Multiplier is not only a high-delay block but also a significant source of power dissipation. That's why, if one also aims to minimize power consumption, it is of great interest to identify the techniques to be applied to reduce delay by using various delay optimizations.

Minimizing power consumption for digital systems involves optimization at all levels of the design. This optimization includes the technology used to implement the digital circuits, the circuit style and topology, the architecture for implementing the circuits and at the highest level the algorithms that are being implemented. The most important technology consideration is the threshold voltage and its control, which allows the reduction of supply voltage without significant impact on logic speed. Since energy is consumed only when capacitance is being switched, power can be reduced by minimizing this capacitance through operation reduction, choice of number representation, exploitation of signal correlations, resynchronization to minimize glitching, logic design, circuit design, and physical design.

1.2 Design Considerations in Integrated Circuits

After guaranteeing correct digital functionality, the primary consideration for system designers has always been speed. A circuit is specified to operate at a particular delay, otherwise the entire system may not work; further reduction is

beneficial but not strictly necessary. Other factors may have equal or greater importance than power dissipation; area of implementation and reliability issues are subjects which designer must take into account. It's worth to note that power reduction techniques are not necessarily negatively correlated to delay reduction. For example, one method to reduce delay in a circuit's critical path is to upsize the driving strength of gates, which results in increased power reduction. However, reducing interconnect capacitance, which is another way to lower delay, reduces both power and delay. Generally, great power savings can be achieved if delay is not an issue, but optimizing power without delay consideration is insignificant.

1.3 Why Low Power?

Power dissipation limitations come in two ways. The first is related to cooling considerations when implementing high performance systems. High-speed circuits dissipate large amounts of energy in a short amount of time, generating a great deal of heat. This heat needs to be removed by the package on which integrated circuits are mounted. Heat removal may become a limiting factor if the package cannot sufficiently dissipate this heat or if the required thermal components are too expensive for the application.

The second failure of high-power circuits relates to the increasing popularity of portable electronic devices. Laptop computers, portable video players and cellular phones all use batteries as a power source. These devices provide a limited time of operation before they require recharging. To extend the battery life, low power operation is desirable in integrated circuits.

1.4 Thesis Outline

This thesis focuses on an algorithm, called Wallace Tree Multiplication [3], which is suitable for high-speed and low-power applications. The algorithm, which is proposed by W. Wallace, is symmetric so it's very applicable for binary multiplication, due to the interchangeability of the multiplicand and the

multiplier. In theory, it is proven that the algorithm is comparably faster than array multiplier and much simpler than the others by means of circuit complexity. Logic style used to implement this algorithm is Gate Diffusion Input (GDI) CMOS logic. The implementation of this algorithm is performed by designing a 8-bit x 8-bit multiplier block in 0.18µ CMOS technology using Cadence Design Framework tools.

This thesis is organized as follows:

CHAPTER I: INTRODUCTION. This chapter introduces power consumption issues in the area of VLSI. This chapter also summarizes the need of low power design in the today's era of scaling down of technologies and nanotechnology. Finally, this thesis chapter explains organization of the thesis.

CHAPTER 2: MULTIPLIER BASICS. This chapter explains what multiplication is? How multiplication is performed in digital domain? It also talks about various multiplier architectures in detail.

CHAPTER 3: CIRCUIT DESIGN. This chapter explains the need of circuit design. It discusses various options in logic styles available to us as per our requirements. Finally it discusses the adder design using the above logic styles.

CHAPTER 4: MULTIPLIER DESIGN. This chapter focuses on the design of multiplier. It explains the each step involved in designing a multiplier from a designers point of view and also shows the simulated waveforms and their validation by comparing with truth tables. It also discusses about the power aspects of the digital design.

CHAPTER 5: PHYSICAL LAYOUT DESIGN AND POST-LAYOUT SIMULATIONS. This chapter discusses the designs of different layouts for all the proposed structures, which are designed in Cadence Assura UMC 0.18 micron Technology and the Layout versus Schematic (LVS) program was executed to perform a comparison of the schematic to the physical layout.

Finally we have the conclusion which compares multiplier design of three logic styles and future scope.

CHAPTER 2

MULTIPLICATION BASICS

2.1 INTRODUCTION

Multiplication is a mathematical operation that at its simplest is an abbreviated process of adding an integer to itself a specified number of times. A number **(multiplicand)** is added to itself a number of times as specified by another number **(multiplier) to** form a result **(product).** In elementary school, students learn to multiply by placing the multiplicand on top of the multiplier. The multiplicand is then multiplied by each digit of the multiplier beginning with the rightmost, Least Significant Digit (LSD). Intermediate results **(partial-products)** are placed one atop the other, offset by one digit to align digits of the same weight. The final product is determined by summation of all the partial-products. Although most people think of multiplication only in base 10, this technique applies equally to any base, including binary. Figure 2.1 shows the data flow for the basic multiplication technique just described. Each black dot represents a single digit.

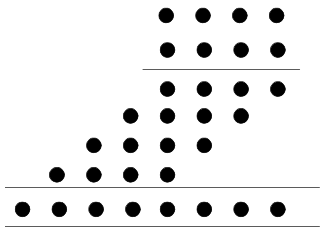


Figure 2.1: Basic Multiplication

Here, we assume that MSB represent the sign of the digit. The operation of multiplication is rather simple in digital electronics. It has its origin from the classical algorithm for the product of two binary numbers. This algorithm uses addition and shift left operations to calculate the product of two numbers. Based upon the above procedure, we can deduce an algorithm for any kind of multiplication which is shown in Figure 2.2. We can check at the initial stage also that whether the product will be positive or negative or after getting the whole result, MSB of the results tells the sign of the product

Digital multiplication is a series of bit shifts and bit additions, where two numbers, the multiplicand and the multiplier are combined into the result. Considering the bit representations of the multiplicand $X_0X_1\ X_{2\ldots\ldots}X_{n-1}$ and the multiplier $Y_0Y_1Y_2\ldots\ Y_{n-1}$, in order to form the product, up to n shifted copies of the multiplicand is to be added for unsigned multiplication. The entire process consists of three steps, partial product generation, partial product reduction and final addition.

Digital multiplication consists of three basic steps, these are:-

1. Generation of Partial Product Array
2. Reduction of Partial Product Array
3. Final Addition

Figure 2.2 below represents the multiplication flow in detail.

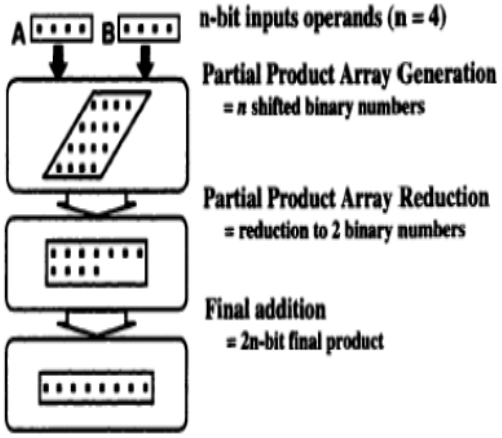


Figure 2.2: Multiplication Flow [4]

2.2 PARTIAL PRODUCT GENERATION

In digital multiplication, as an initial step, one needs to generate n shifted copies of the multiplicand, which may be added in the coming stage. The value of the multiplier bit determines whether the shifted copy is to be added or not: if the i^{th} bit ($0 \leq i \leq n-1$) of the multiplier is '1', then the shifted copy of the multiplicand is added. If the bit is '0', it's not added. A logical AND gate can implement this operation, by performing the function AND ($x_i y_j$) ($0 \leq i \leq n-1$ & $0 \leq y \leq n-1$). The resulting values are called partial products. Fig.2.3 shows a trapezoidal structure, called partial product array (PPA), where the partial product bits are arranged in columns to be added in order to form the product. This process is called product array generation [4].

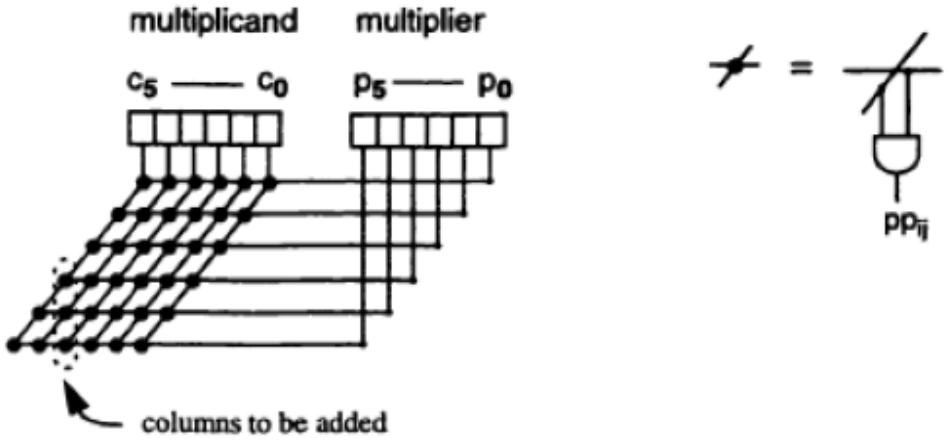


Figure 2.3: Partial Product Generation [4]

2.3 PARTIAL PRODUCT REDUCTION

Efficient implementation of a digital multiplier depends on the method of the addition of partial product array bits. Since each shifted version of the multiplicand will give a delay proportional to the width of the multiplicand, the multiplier block will require a large amount of time to perform the operation if conventional adders were used to implement this addition. Hence, the partial

products are reduced using a technique, called *carry-save addition*, which allows successive additions in one global step.

Considering the addition of two bits from two vectors, X and Y, where numerical bit vector representations are of the form X= $x_{n-1}x_{n-2}\ldots\ldots x_1 x_0$ and Y= $y_{n-1}y_{n-2}\ldots y_1 y_0$, conventional full-adder can be used, which takes in three bits and outputs a sum and a carry bit, so the block adds two bits at a given position with the carry in from the previous bit position. Considering the case of adding two bit vectors, two bits are added at the lowest bit position and the carry is propagated to the next bit position. At the higher positions, two inputs and the carry bit are to be combined and a carry out is generated. This rippling technique of adding two n-bit numbers requires O (n) sequential bit additions, hence a delay of $O(n)$. For the addition of three n-bit vectors X, *Y* and *Z*, this method can be used to add *X* and *Y*, then to add *Z* to the sum of *X +Y;* so the total number of bit additions of *n* shifted copies of an *n*-bit multiplicand is $O(n)$ where the total delay is $O(n^2)$, assuming that the add operations are dependent on the previous ones since the output of earlier operations are inputs to later operations [4].

Even though the result comes from the combination of all operations, a certain amount of independence exists between each operation, considering the addition on a particular column. All the bits in a column must be added together along with the carry in bits coming from the previous column. Carry save addition implies that addition in separate columns can be performed independently without concerning about carry from previous column. For example, in order to add three vectors of bits, full-adders can be used to perform the addition of three bits in each column. Except the lowest and the highest bit positions, the result is a carry and a sum bit in each bit position. So, the three bit vectors have been reduced to two bit vectors. Using carry save addition technique, a set of vectors, which are to be added together, can be reduced to two bit vectors. Carry save addition is one of the ways to make a multiplication faster than the conventional methods, considering the number of necessary additions There are several ways to implement addition of partial products in the trapezoidal array. This also forms the basis of classification among parallel multipliers. In the following section we would discuss the two most commonly used methods.

1. Array Multiplier

2. Tree Multiplier

2.3.1. ARRAY MULTIPLIER

Array multiplier [6] is well known due to its regular structure. In array multiplier, the counters and compressors are connected in a serial fashion for all bit slices of the Partial Product parallelogram. As can be seen in Figure 2.4. The array topology is a two-dimensional structure that fits nicely on the VLSI planar process [5]. There are several possible array topologies including simple, double and higher order arrays.

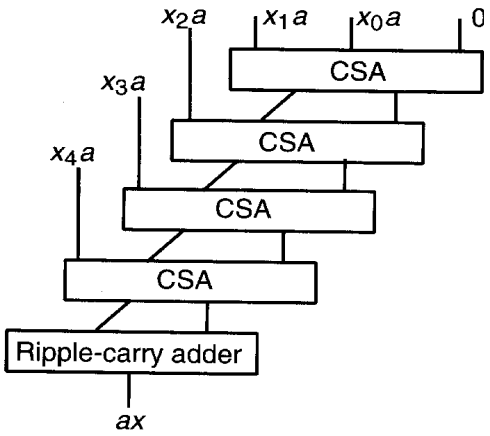


Figure 2.4: Array Multiplier Mechanism [6]

2.3.1.1 SIMPLE ARRAY MULTIPLIER

The array multiplier originates from the multiplication parallelogram. As shown in Figure 2.5, each stage of the parallel adders should receive some partial product inputs. The carry-out is propagated into the next row. The bold line is the critical path of the multiplier. In array multiplier, all of the partial products are generated at the same time. It is observed that the critical path consists of two parts: vertical and horizontal. Both have the same delay in terms of full adder delays and gate delays. For an n-bit by n-bit array multiplier, the vertical and the horizontal delays are both the same as the delay of an n-bit full adder. For a 16-bit multiplier, the worst-case delay is 32 τ where τ is the worst-case full adder delay.

In the simple array, each row of [3:2] compressors adds a partial product to the partial sum, generating a new partial sum and a sequence of carries. The delay of the array depends on the depth of the array. Therefore, the summing time for the simple array is (N-2) [3:2] compressor delays, where N is the number of partial products. The drawback of this type of array is the hardware is underutilized. The counters are used only once in the calculation of the result, for the remaining time, they are idle. This drawback can be diminished by pipelining the array so that several multiplications can occur simultaneously. Pipelining would increase the throughput of the multiplier, but would also increase the latency and area of the multiplier. A fully pipelined array is normally avoided, since the array would be faster than the clock of processor. Since the intermediate partial sum is kept in a redundant, carry-save form there is no carry propagation. This means that the delay of an array multiplier is only dependent upon the depth of the array, and is independent of the partial-product width.

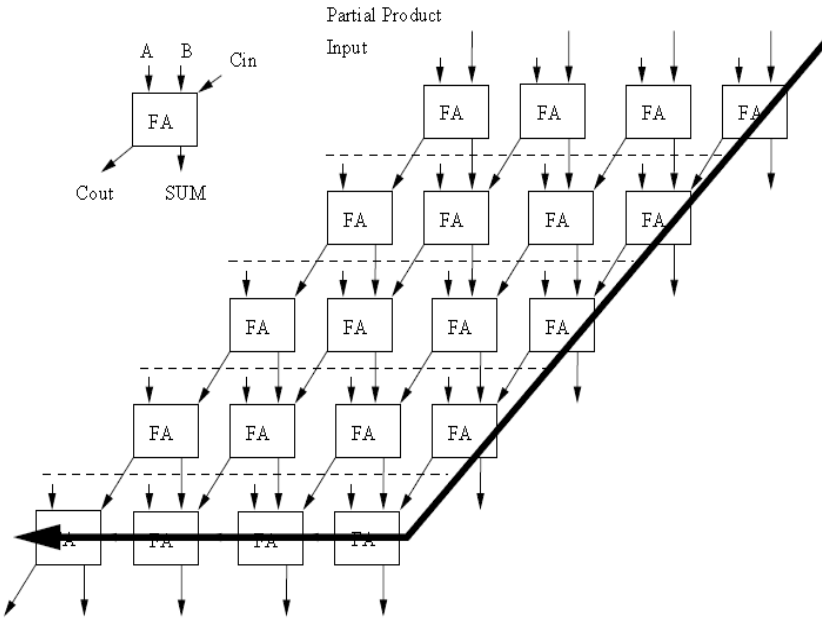


Figure2.5: 4 x 4 Array Multiplier [6]

2.3.1.2 ADVANTAGES OF ARRAY MULTIPLIER

One advantage of the array multiplier comes from its regular structure. Since it is regular, it is easy to layout and has a small size. The design time of an array multiplier is much less than that of a tree multiplier.

A second advantage of the array multiplier is its ease of design for a pipelined architecture. A fully pipelined array multiplier with a stage delay equal to the delay of a 1-bit full adder plus a register has been successfully designed for high-speed DSP applications. Also it can be easily pipelined by inserting latches after CSA (carry save adders) or after every few rows.

2.3.1.3 LIMITATIONS OF ARRAY MULTIPLIER

The biggest problem with full linear array multipliers is that they are very large. As operand sizes increase, linear arrays grow in size at a rate equal to the square of the operand size. This is because the number of rows in the array is equal to the length of the multiplier, with the width of each row equal to the width of multiplicand. The large size of full arrays typically prohibits their use, except for small operand sizes, or on special purpose math chips where a major portion of the silicon area can be assigned to the multiplier array.

Another problem with array multipliers is that the hardware is underutilized. As the sum is propagated down through the array, each row of CSA's computes a result only once, when the active computation front passes that row. Thus, the hardware is doing useful work only a very small percentage of the time. This low hardware utilization in conventional linear array multipliers makes performance gains possible through increased efficiency.

2.3.2 TREE MULTIPLIER

Trees are an extremely fast structure for summing partial-products. In a linear array, each row sums one additional partial product. As such, linear arrays require order N stages to reduce N partial-products. In contrast, by doing the additions in

parallel, tree structures require only order log N stages to reduce N partial products [1].

The result of the multiplication is obtained by first generating partial products and then adding the partial products. The critical path of a multiplier depends on the delay of the carry chain through all of the adders. Table 2.1 shows the truth table of a full adder, which is the basic addition process usually employed in a computer to add two numbers together. A and B are the adder inputs, and C is the carry input. The full adder produces a bit of summand and a bit of carry out. It can be observed that a full adder is actually a “one's counter". A, B and C can all be seen as the inputs of [3:2] compressor. The outputs, Carry and Sum, are the encoded output of the three inputs in binary notation. The tree multiplier is based on this property of the full adder. The addition of summands can be accelerated by adopting a [3:2] compressor [7]. A [3:2] compressor adds three bits and produces a two-bit binary number whose value is equal to that of the original three. The advantage of the [3:2] compressor is that it can operate without carry propagation along its digital stages and hence is much faster than the conventional adder. In any scheme employing [3:2]compressors, the number of adder passes occurring in a multiplication before the product is reduced to the sum of two numbers, will be two less than the number of summands, since each pass through an adder converts three numbers to two, reducing the count of numbers by one. To improve the speed of the multiplication, one must arrange many of these passes to occur simultaneously by providing several [3:2] compressors

Table 2.1: Truth Table of Full Adder

No. of Zeros	Carry	Sum	A	B	C
0	0	0	0	0	0
1	0	1	0	0	1
1	0	1	0	1	0
2	1	0	0	1	1
1	0	1	1	0	0
2	1	0	1	0	1
2	1	0	1	1	0
3	1	1	1	1	1

Thus, the best first step for a tree multiplier is to group the summands into threes, and introduce each group into its own [3:2] compressor, thus reducing the count of numbers by a factor of 1.5. The second step is to group the numbers resulting from the first step into threes and again add each group in its own [3:2] compressors. By continuing such steps until only two numbers remain, the addition is completed in a time proportional to the logarithm of the number of summands Figure 2.6 shows a 4-bit tree multiplier. It should be noted that [4:2] compression can also be used besides the [3:2] compression. A [4:2] compression can be achieved by combining two full adders. For a 16-bit multiplier, the worst-case delay is $7\tau_{fa} + \tau_{cpa}$ where τ_{fa} is the delay [8] of a 3-2 compressor, which is the delay of a full adder, and τ_{cpa} is the delay of the fast carry propagate adder.To improve the speed of the multiplication, one must arrange many of these passes to occur simultaneously by providing several [3:2] compressors. Thus, the best first step for a tree multiplier is to group the summands into threes, and introduce each group into its own [3:2] compressors, thus reducing the count of numbers by a factor of 1.5. The second step is to group the numbers resulting from the first step into threes and again add each group in its own [3:2] compressors. By continuing such steps until only two numbers remain, the addition is completed in a time proportional to the logarithm of the number of summands.

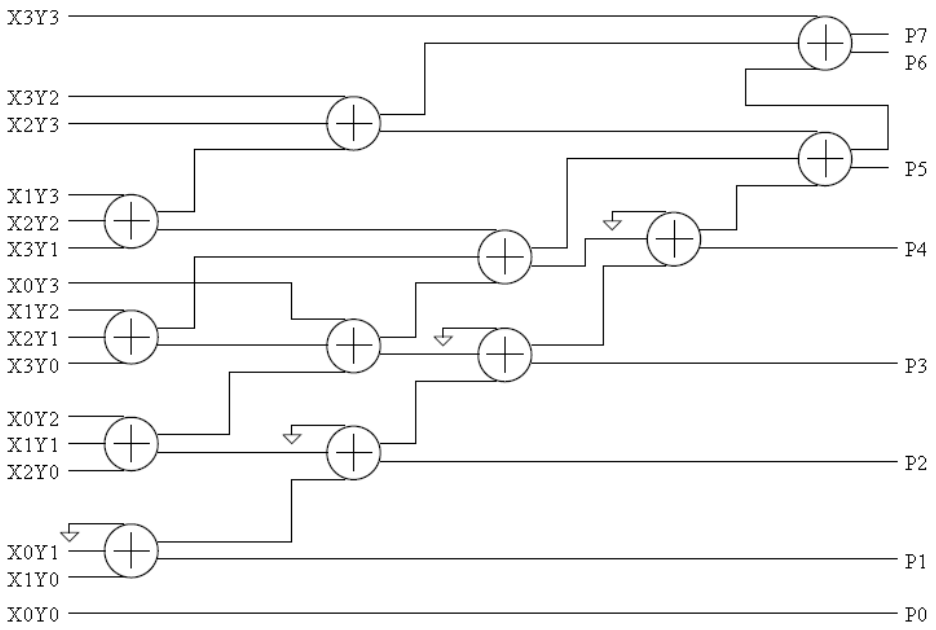


Figure 2.6: 4 x 4 Tree Multiplier [9]

The first tree structure was introduced by Wallace. Wallace showed that PPs can be reduced by connecting [3:2] compressors in parallel in a tree topology. The regular trees include binary, balanced-delay and overturned-staircase trees as well as [9:2] compressors.

2.3.2.1 WALLACE TREE MULTIPLIER

Several popular and well-known schemes, with the objective of improving the speed of the parallel multiplier, have been developed in past. In 1964, C.S. Wallace observed that it is possible to find a structure, which performs the addition operations in parallel; thus resulting in less delay. A Wallace tree [9] is an implementation of an adder tree designed for minimum propagation delay. Rather than completely adding the partial products in pairs like the ripple adder tree does, the Wallace tree sums up all the bits of the same weights in a merged tree.

A **Wallace tree** is an efficient hardware implementation of a digital circuit that multiplies two integers. The Wallace tree has three steps:

- Multiply (that is - AND) each bit of one of the arguments, by each bit of the other, yielding n^2 results. Depending on position of the multiplied bits, the wires carry different weights, for example wire of bit carrying result of a_2b_3 is 32 (see explanation of weights below).
- Reduce the number of partial products to two by layers of full and half adders.
- Group the wires in two numbers, and add them with a conventional adder.

The second phase works as follows. As long as there are three or more wires with the same weight add a following layer:

- Take any three wires with the same weights and input them into a full adder. The result will be an output wire of the same weight and an output wire with a higher weight for each three input wires.
- If there are two wires of the same weight left, input them into a half adder
- If there is just one wire left, connect it to the next layer.

- Wallace introduced a different way of parallel addition of the partial product bits using a tree of carry save adders, which is known as "Wallace Tree".

In order to perform the multiplication of two numbers with the Wallace method, partial product matrix is reduced to a two-row matrix by using a carry save adder and the remaining two rows are summed using a fast carry-propagate adder to form the product. This advantage becomes more pronounced for multipliers of bigger than 16 bits. Wallace tree flow can be seen in Figure 2.7.

In Wallace Tree architecture, all the bits of all of the partial products in each column are added together by a set of counters in parallel without propagating any carries. Another set of counters then reduces this new matrix and so on, until a two-row matrix is generated. Here a 3:2 counter is used. Then, a fast adder is used at the end to produce the final result. The advantage of Wallace tree is speed because the addition of partial products is now O (log N).

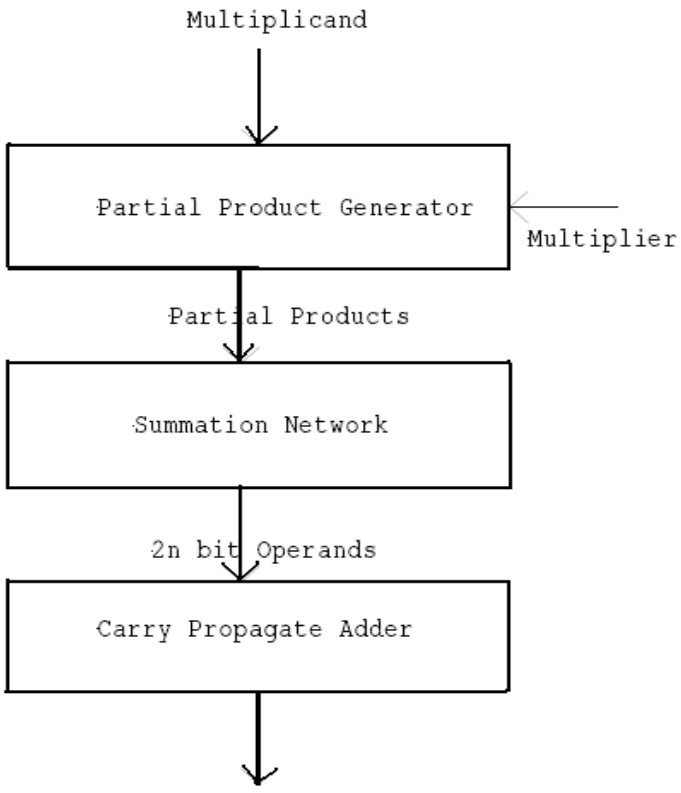


Figure 2.7: Flow of Wallace Tree Multiplier

Wallace method uses three-steps to process the multiplication operation

- Formation of bit products
- The bit product matrix is reduced to a 2-row matrix by using a carry-save adder
- The remaining two rows are summed using a fast carry-propagate adder to produce the product.

In general, its multiplication process can be summarized as follows:

1. After generating the partial products, a set of counters reduces the partial product matrix but it does not propagate the carries.
2. The resulting matrix is composed of the sums and carries of the counters.
3. Another set of counters then reduces this matrix and the whole process continues until a two- row matrix is generated.
4. The two rows get summed up with a final adder, preferably by a carry propagate adder. This method takes advantage of the carry save architecture in order to avoid the carry propagation until the final adder. In this scheme, the number of levels is crucial since they determine the speed of the multiplier.

The conventional Wallace tree algorithm reduces the propagation by incorporating [3:2] compressors.

The Figure 2.8 explains the various steps of Wallace tree multiplier. In stage 1 the partial products are reduced using compressors. The partial terms marked as red are kept as such, the one marked with dark green indicates that they are compressed with full adder, the one marked with light yellow indicate 3:2 compressors and the one with white box indicates4:2

compressors.

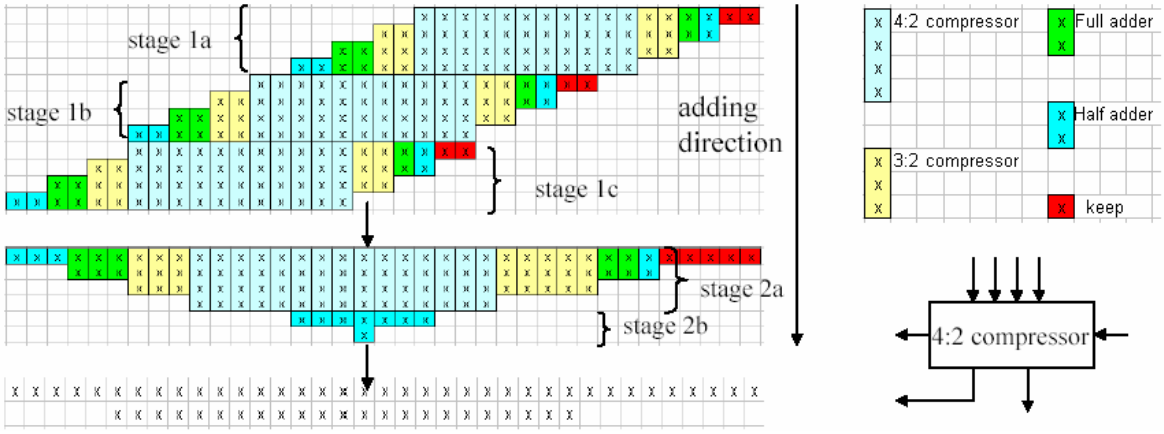


Figure 2.8: Wallace Tree Example [9]

2.4 COMPRESSORS

Compressors are mostly used in multipliers to reduce the operands while adding terms of partial products. A compressor Ci is a combinatorial device that compresses N input lines in the position i to 2 output lines i.e. sum and carry. In addition, there are L inputs lines coming to the compressor to different levels j. Figure 2.9 shows a simple compressor.

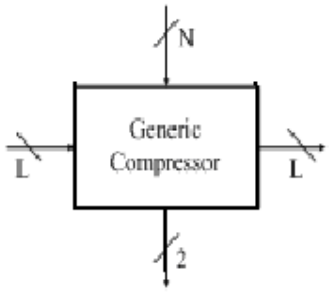


Figure 2.9: A Generic Compressor

2.4.1 [3:2] COMPRESSOR

A [3:2] compressor is basically a Full adder. It has 3 inputs A, B and C to be summed up and provides 2 outputs (sum and carry). Gate level diagram of [3:2] compressor is shown in Figure2.10.

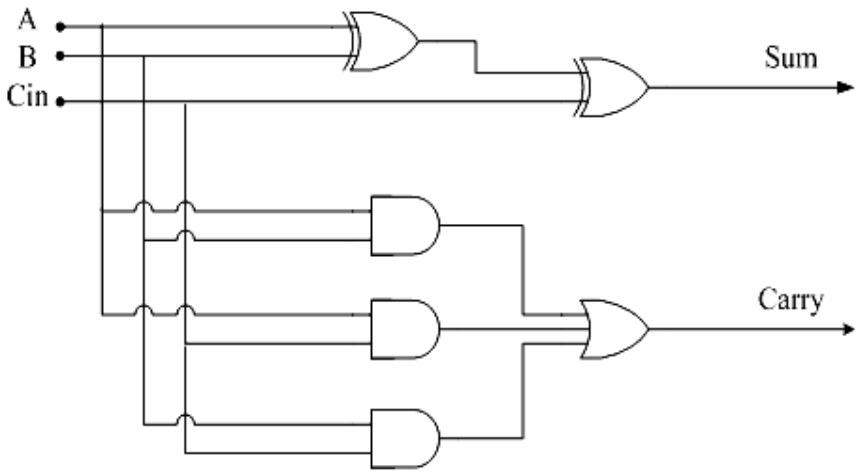


Figure 2.10: Gate level design of [3:2] compressor

2.4.2 [4:2] COMPRESSOR

A [4:2] compressor [10] has 4 input lines i1, i2, i3 and i4 that must be summed and has two output lines s and c, which are so called results of compression. The additional lines are input and output carries. In actual the structure compresses five partial product bits four input bits and one carry - in bit) into three. However it acts as a compressor reducing the four bits into compressor reducing the four bits into two, since carry-in and carry-out bits connect the adjacent 4:2 compressor. Thus, the number of partial product bits is reduced by half in one stage, making the efficiency higher. The truth table of compressor is given in table 2.2. A [4:2] compressor can also be designed using two [3:2] compressors as shown in Figure 2.11.

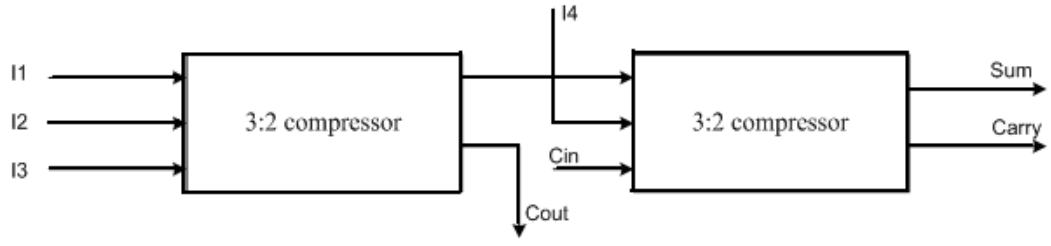


Figure 2.11: [4:2] compressor using [3:2] compressor

Table 2.2: Truth Table of [4:2] Compressor

INPUTS				**Cin = 0**		**Cin = 1**		**Cout**
A	**B**	**C**	**D**	**Carry**	**Sum**	**Carry**	**Sum**	
0	0	0	0	0	0	0	1	0
0	0	0	1	0	1	1	0	0
0	0	1	0					
0	1	0	0					
1	0	0	0					
0	0	1	1	0	0	0	1	1
0	1	0	1					
0	1	1	0					
1	0	0	1					
1	0	1	0					
1	1	0	0					
0	1	1	1	0	1	1	0	1
1	0	1	1					
1	1	0	1					
1	1	1	0					
1	1	1	1	1	0	1	1	1

CHAPTER

3 CIRCUIT DESIGN

3.1 INTRODUCTION

Circuit design plays an important role in the design of digital circuits like multiplier. First, to guarantee the multiplier to work at the desired clock rate, the designer has to know the delay of the critical path and the required time of inserting a pipeline stage. Second, to reduce the area of the multiplier, several architectures of adders are investigated. Circuit analysis helps the designer verify the functions and performances of the adders. The architecture of the adder has to be determined first. Then the number of the pipeline stages can be decided by the speed of the adder. The size of the multiplier should be as small as possible if all the requirements can be met.

Fast arithmetic requires fast circuits. Fast circuits require small size, to minimize the delay effects of wires. Small size implies a single chip implementation, to minimize wire delays, and to make it possible to implement these fast circuits as part of a larger single chip system to minimize input/output delays. The increasing demand for low-power VLSI asks, among others, for power efficient logic styles [2]. Performance criteria for logic styles are circuit speed, circuit size, power dissipation, and wiring complexity as well as ease-of-use and generality of gates in cell-based design techniques. Dynamic logic styles are often a good choice for high-speed, but not for low-power circuit implementations due to the high node activity and large clock loads [2]. This chapter focuses review of various logic styles suitable for low power.

3.2 CMOS LOGIC STRUCTURE

Today CMOS (Complementary Metal Oxide Semiconductor) [11] is the primary technology in the Semiconductor industry. Most high speed microprocessors are implemented using CMOS. Contemporary CMOS technology is characterized by:

- Small minimum sized transistors, allowing for dense layouts, although the interconnect limits the density.
- Low Quiescent Power - The power consumption of conventional CMOS circuits is largely determined by the AC power caused by the charge and discharge of capacitances:

  $$\text{Power} = CV^2f \tag{3.1}$$

  Where f is the frequency at which a capacitance is charged and discharged. As the circuits get faster, the frequency goes up as does the power consumption.
- Relatively simple fabrication process.
- Large required transistors - In order to drive wires quickly, large width transistors are needed, since the time to drive a load is given by :

  $$\Delta t = C\frac{\Delta V}{i} \tag{3.2}$$

  Where:

  Δt is the time to charge or discharge the load

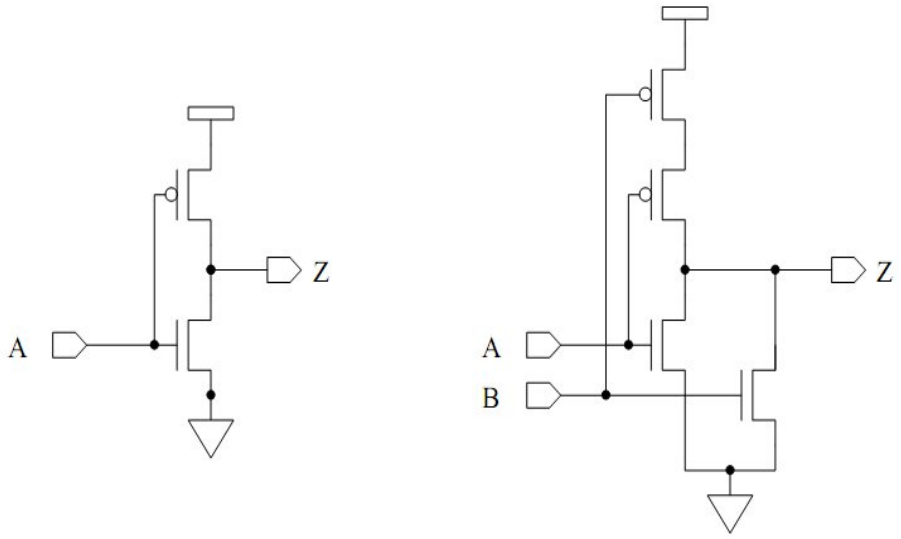
  C is the capacitance associated with the load

  ΔV is the load voltage swing

  i is the average current provided by the load driver
- Large voltage swings - Typical voltage swings for contemporary CMOS are from 3.3 to 5volts (with even smaller swings on the way). All other things being equal, equation 3.2 says that a smaller voltage swing will be proportionally faster.
- Good Noise Margin

Figure 3.1 (a) shows a CMOS inverter. There is a pull-up PMOS transistor and a pull-down NMOS transistor. The steady state output will be independent of the ratio of the pull-up and pull-down transistor sizes. Because of this, CMOS

complementary logic does not have to worry about signal degradation problems in pass-transistor logic. Because the power-to-ground path only closes during the transition, it almost consumes no static power. The CMOS complementary gate has two function determining blocks an n-block and a p-block. There are normally 2n transistors in an n-input gate.



(a) (b)

Figure 3.1: (a) CMOS Inverter (b) CMOS NOR Gate

3.3 PASS TRANSISTOR LOGIC

In pass transistor logic [12] style we use either NMOS or PMOS transistor to build a design. Figure 3.2 shows an XNOR implemented in pass-transistor logic. A is used as a control signal and B is used as a pass signal. In this circuit, when A is true, B is passed to the output. When A is false, the complement of B is passed. As can be seen this logic form is not always active, which limits its ability to be used in long chains. The advantage of pass-transistor logic is that it occupies much less area to construct complex Boolean function. The XNOR in CMOS complementary logic needs 3 2-input NAND gates, which consisted of 12 transistors. It only needs 2 transistors in pass-transistor logic as shown in Figure 3.2. Hence we see that pass transistor logic requires less number of transistors

than CMOS logic, but the output voltage swing is limited as neither of the transistor (n or p) is good conductor of both ‘0’ and ‘1’.

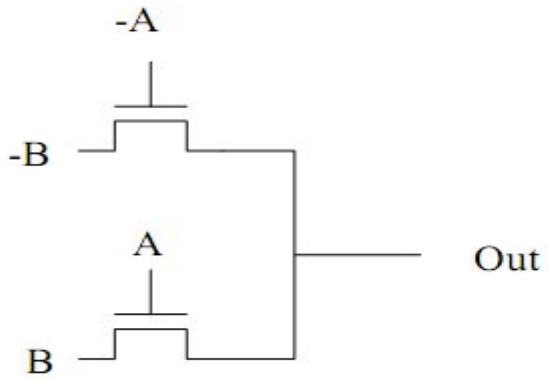


Figure 3.2: Two Input XOR Gate using Pass Transistor Logic

3.4 TRANSMISSION GATE LOGIC

The transistor connections for a complementary switch or transmission [13] are reviewed in Figure 3.3. It consists of an n-channel transistor and p-channel transistor with separate gate connections and common source and drain connection. The control signal $ is applied to the gate of n- device and its complement to gate of p-device. Operation can be well explained by considering the n and p device separately. When the control signal $ is low i.e. ‘0’ both n and p devices are off and output is high impedance. Similarly when $ is high i.e. ‘1’ both n and p devices are on and input is transferred to output node. The corresponding output characteristics are shown in Figure 3.4.

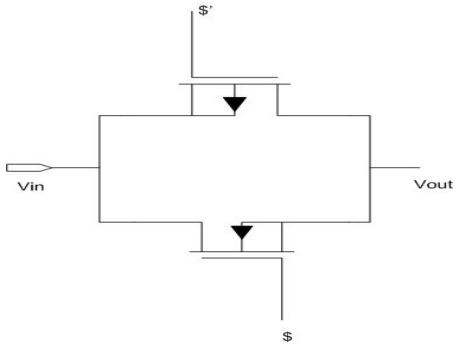


Figure 3.3: Transmission Gate Logic

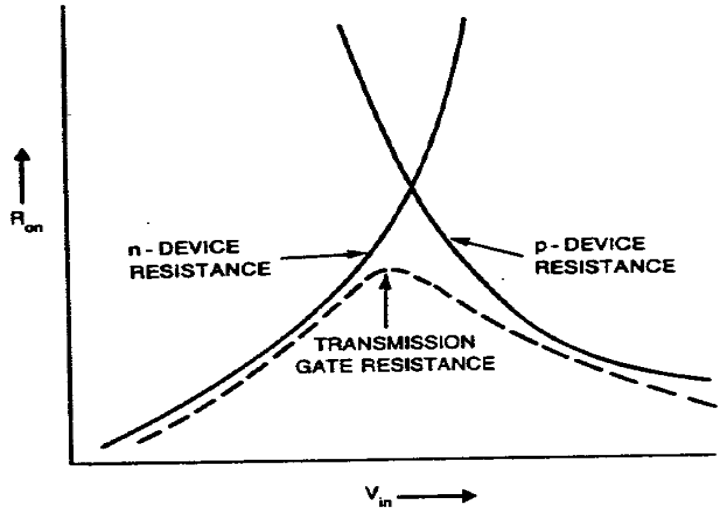


Figure 3.4: Transmission gate output characteristics[7]

3.5 COMPLEMENTARY PASS-TRANSISTOR LOGIC

Several differential CMOS logic families such as cascade voltage switch logic (CVSL) (Figure 3.5) have been proposed for CMOS circuit speed improvement. CVSL is a differential style of logic requiring both true and complementary signals to be routed to gates [15]. Two complementary NMOS switch structures are constructed and then connected to a pair of PMOS cross-coupled load, which together can reduce input capacitance, increase logic functionality, and sometimes eliminate inverter circuits. Therefore, these logic families can increase speed. However, the actual advantage of CVSL circuits is less than that anticipated in the original paper, as clarified in [8]. This is because the PMOS cross-coupled latch cannot easily be inverted due to the “fighting" to the NMOS pull-down trees. High-speed inversion of the PMOS latch is possible only when the gate width of the PMOS is sufficiently small. However, a small gate width severely degrades the pull-up transit time. The main concept behind CPL is the use of an NMOS pass-transistor network for logic organization, and elimination of the PMOS latch such as cascade voltage switch logic (CVSL), as shown in Figure 3.5. CPL consists of complementary inputs/outputs, an NMOS pass transistor logic network, and CMOS output inverters. The pass transistors and the

inverters function as pull-down and pull-up devices. Thus the PMOS latch can be eliminated, allowing the advantage of the differential circuits to be fully utilized.

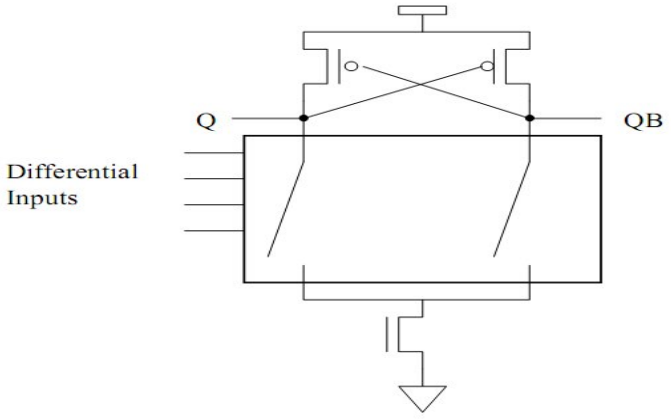


Figure 3.5: CVSL Schematic Structure

Because the high level of the pass-transistor outputs (nodes Q and Q) is lower than the supply voltage level by the threshold voltage of the pass transistors, the signal have to be amplified by the output inverters. At the same time, the CMOS output inverters shift the logic threshold voltage and drive the capacitive load . Figure 3.6 shows an AND/NAND circuit in CPL.

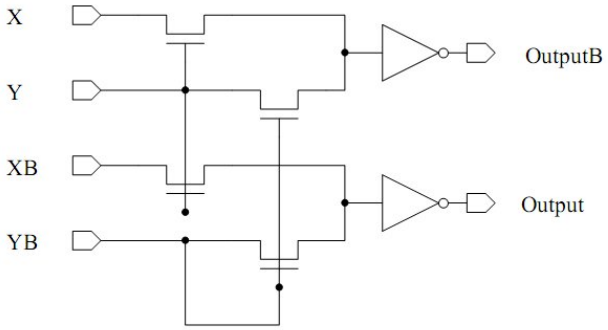


Figure 3.6: CPL AND/NAND Gate

3.6 GATE DIFFUSION INPUT (GDI)

Gate diffusion input (GDI) [16]—a new technique of low-power digital combinational circuit design. This technique allows reducing power consumption, propagation delay, and area of digital circuits while maintaining low complexity

of logic design. Pass-transistor logic has been presented for NMOS. They are based on the model, where a set of control signals is applied to the gates of NMOS transistors. Another set of data signals are applied to the sources of the n-transistors [2]. Some of the main advantages of PTL over standard CMOS design are

- High speed, due to the small node capacitances
- Low power dissipation, as a result of the reduced number of transistors
- Lower interconnection due to a small area.

However, most of the PTL implementations have two basic problems. They are:

- The threshold drop across the single-channel pass transistors results in reduced current drive and hence slower operation at reduced supply voltages; this is particularly important for low-power design since it is desirable to operate at the lowest possible voltage level.
- Since the “high input voltage level at the regenerative inverters is not VDD, the PMOS device in the inverter is not fully turned off, and hence direct-path static power dissipation could be significant

A new low-power design technique that allows solving most of the problems mentioned above—gate diffusion input (GDI) technique. The GDI approach allows implementation of a wide range of complex logic functions using only two transistors. This method is suitable for design of fast, low-power circuits, using a reduced number of transistors (as compared to CMOS and existing PTL techniques), while improving logic level swing and static power characteristics and allowing simple top-down design by using small cell library.

The GDI method is based on the use of a simple cell as shown in Figure. 3.7 At first glance, the basic cell reminds one of the standard CMOS inverter, but there are some important differences.

- The GDI cell contains three inputs (common gate input of nMOS and pMOS), P (input to the source/drain of pMOS), and N (input to the source/drain of nMOS).
- Bulks of both nMOS and pMOS are connected to N or P (respectively), so it can be arbitrarily biased in contrast with a CMOS inverter.

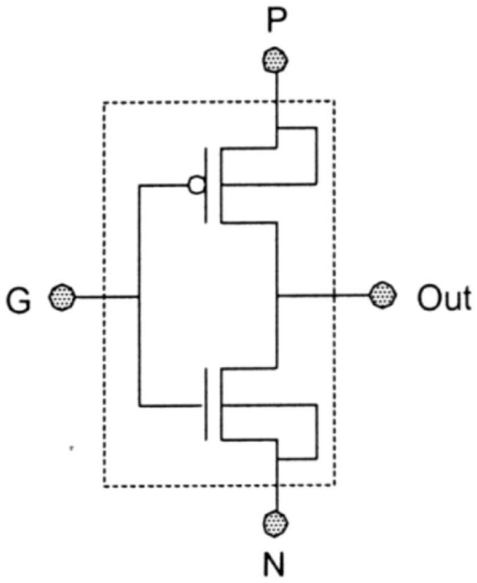


Figure 3.7: GDI Basic Cell [16]

It must be remarked that not all of the functions are possible in standard p-well CMOS process but can be successfully implemented in twin-well CMOS or silicon on insulator (SOI) technologies. Table 3.1 shows how a simple change of the input configuration of the simple GDI cell corresponds to very different Boolean functions. Most of these functions are complex (6–12 transistors) in CMOS, as well as in standard PTL implementations, but very simple (only two transistors per function) in the GDI design method.

Table 3.1: Various Logic Functions of GDI cell for different Input configurations

N	P	G	OUT	FUNCTION
‘0’	B	A	A’B	F1
B	‘1’	A	A’ + B	F2
‘1’	B	A	A + B	OR
B	‘0’	A	AB	AND
C	B	A	A’B + AC	MUX
‘0’	‘1’	A	A’	NOT

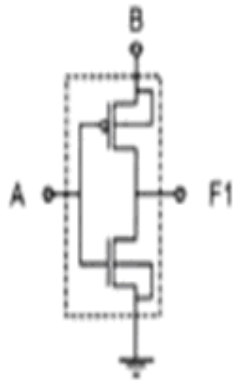
3.6.1 OPERATIONAL ANALYSIS OF GDI CELL

As mentioned in Section I, one of the common problems of PTL design methods is the low swing of output signals because of the threshold drop across the single-channel pass transistors. In existing PTL techniques, additional buffering circuitry is used to overcome this problem. To understand the effects of the low swing problem in a GDI cell, we suggest the following analysis, based on the example of F1 function, and can be easily extended to use in other GDI functions. Table 3.2 presents a full set of logic states and related functionality modes of F1.

As can be seen from Table 3.2, the only state where low swing occurs in the output value is A= '0' and B='0'. In this case, the voltage level of F1 is V_{tp} (instead of the expected 0 V) because of the poor high-to-low transition characteristics of the PMOS pass transistor [4]. It is obvious that the only case (among all the possible transitions) where the effect occurs is the transition from A= '0' and B= Vdd to A= '0' and B='0'.

The fact that demands special emphasis is that in about 50% of the cases (for), the GDI cell operates as a regular CMOS inverter, which is widely used as a digital buffer for logic-level restoration. In some of these cases, when VDD = '1' without a swing drop from the previous stages, a GDI cell functions as an inverter buffer and recovers the voltage swing. Although this feature allows a self-swing restoration in certain cases, in this thesis the worst case is assumed and additional circuitry is used for swing restoration in the implemented circuits.

Table 3.2: Input logic states versus functionality and output swing of F1 function



A	B	Functionality	F1
0	0	pMOS Trans Gate	VTp
0	1	CMOS Inverter	1
1	0	nMOS Trans Gate	0
1	1	CMOS Inverter	0

3.7 ADDER

The adder is the basic element in computer arithmetic. It is also the critical element of the multiplier. The truth table of a one-bit full adder is shown on Table 3.1. We can derive the Boolean function in sum of products from this truth table as the following:

$$SUM = ABC + \overline{AB}C + A\overline{BC} + B\overline{AC} \quad (3.3)$$

$$CARRY = AB + BC + AC \quad (3.4)$$

A and B are the adder inputs, C is the carry input, SUM is the sum output, and CARRY is the carry output. The generate signal, G, occurs when a carry output (CARRY) is internally generated within the adder. When the propagate signal, P, is true, the carry-in signal (C) is passed to the carry output (CARRY) when C is true [13]. Because the carry-in of the current bit is determined by lower bits of two operands, the delay of the adder depends on the generation of the carry.

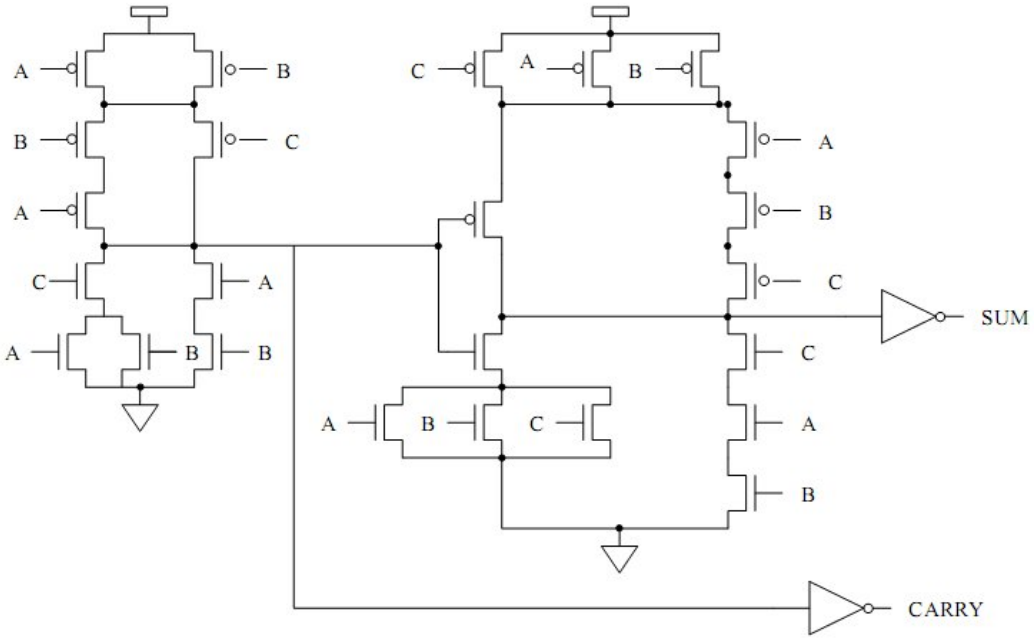


Figure 3.8: CMOS Full Adder

3.7.1 CMOS FULL ADDER

Equation 3.4 may be factored as follows:

$$CARRY = AB + C(A + B) \quad (3.5)$$

$$\overline{CARRY} = \overline{AC} + \overline{BC} + \overline{AB} \quad (3.6)$$

From Equation 3.2 and Equation 3.6, SUM can be expressed as follows:

$$SUM = ABC + (A + B + C)\overline{CARRY} \quad (3.5)$$

This implementation is shown in Figure 3.8.

3.7.2 TRANSMISSION-GATE ADDER

Figure 3.9 shows a transmission gate [15] adder which consists of 18 transistors. The inputs A and B are connected to an XNOR gate. The transmission gates and the inverters on the output from two XOR gates. The outputs may not be active. The transmission gate between C and SUM may form a long chain of the transmission gates for a wide adder. The same can be applied to the transmission gate between C and CARRY. The long chain of the transmission gates will act as a RC circuit. Because the resistance and capacitance of the chain becomes large, the speed will slow down. So it becomes necessary to insert some buffers in a wide adder.

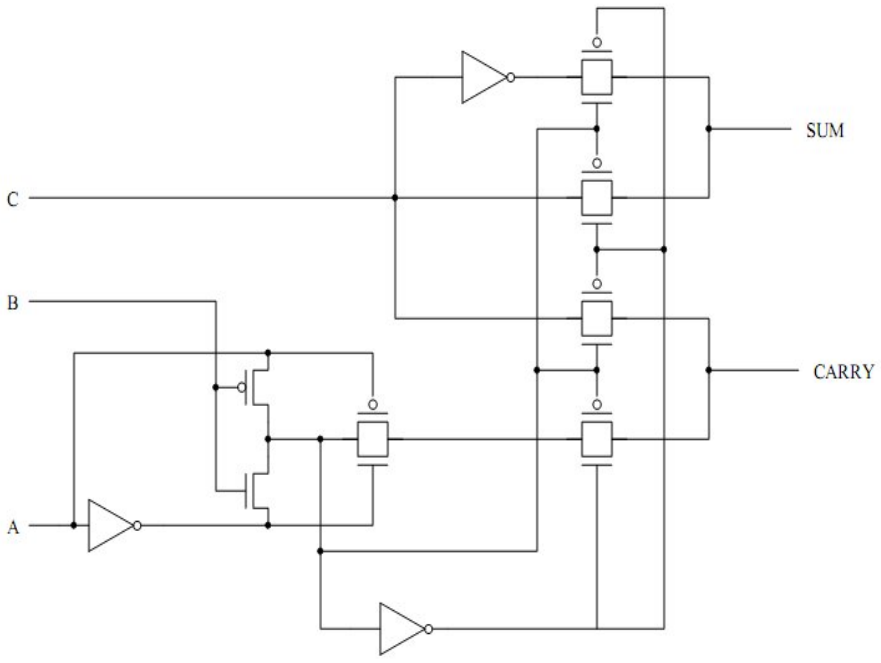


Figure 3.9: Full Adder using Transmission Gates

3.7.3 GDI ADDER

Gate Diffusion Induced technique is a novel low power technique used to build logic circuits. It requires fewer transistors as compared to other CMOS techniques. Extra circuitry required in this is the precharge circuitry. Adder Circuit using this technique is implemented in Figure 3.10. As mentioned, one important requirement of full adder cells especially at low voltage is to provide enough driving power to the following circuits. The drivability is ensured by the full signal swing and decoupling of inputs and outputs (at least one inverter per cell) so that the adder cell can be cascaded arbitrarily and work reliably in any circuit configuration. So the second stage of full adder cells which generate Sum and Carry must have enough drivability. In addition, there are several choices of circuits to generate signal Sum. We use a similar circuit as that of TFA, but fully exploit the available XOR and XNOR outputs from stage I to allow a single inverter to be attached at the last stage. The output inverter guarantees sufficient drive to the cascaded cells.

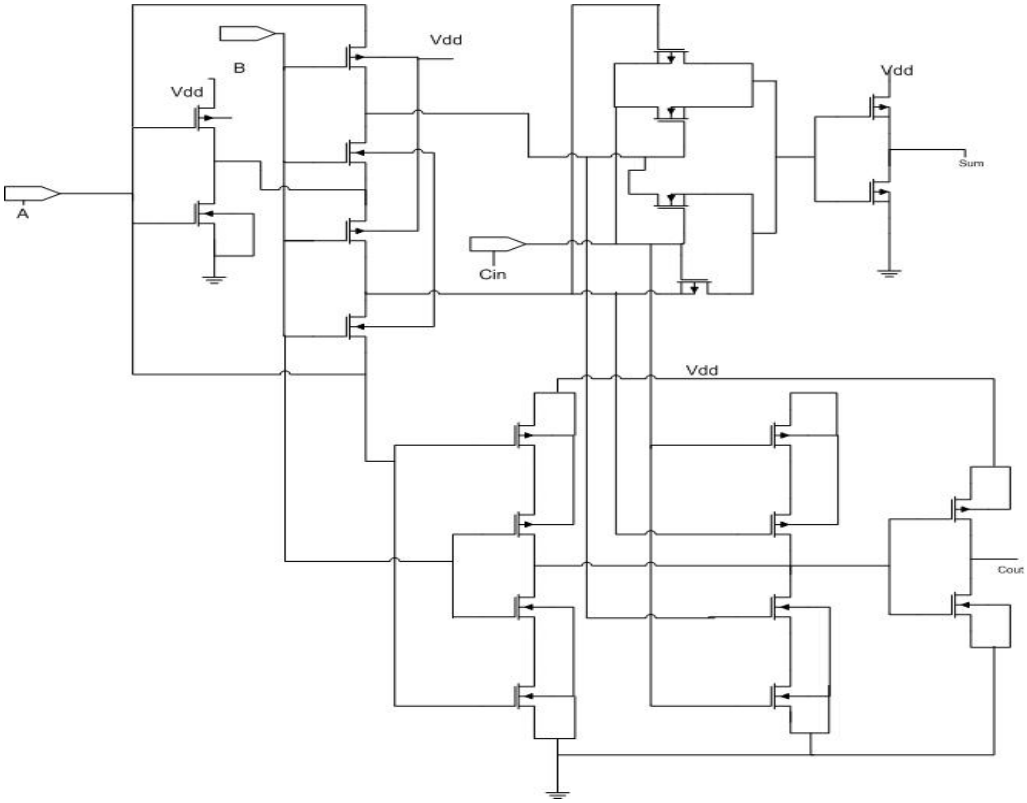


Figure 3.10: GDI full adder

3.5.3.1 ADVANTAGE OF GDI ADDER

GDI adder has following benefits over other power reduction techniques

- It requires least number of transistors to implement a full adder.
- Output swing is rail to rail as compared to Pass Transistor Logic style which has lowered output swing.
- It has least static power dissipation as compared to CMOS and other logic styles.
- Reduced dynamic component of power consumption as source of pmos is not tied to vdd permanently.

3.5.3.2 LIMITATIONS OF GDI ADDER

GDI adder has only one limitation that it requires additional circuitry to restore its full swing.

CHAPTER 4

MULTIPLIER DESIGN

4.1 INTRODUCTION

Multiplier design can be divided into two blocks. These are

1. Partial Product Generation
2. Partial product Addition

This chapter discusses in detail the design steps, verify the truth tables with simulations results and power aspects associated with the above mentioned blocks.

4.2 PARTIAL PRODUCT GENERATION

As discussed in chapter 2, partial product is basically an AND operation of the i^{th} bit of multiplier with k^{th} bit of multiplicand. So we design an AND gate using GDI CMOS logic design. For n-bit multiplication we require n^2 AND gates, hence for our design of 8-bit multiplier we require 64 and gates. Figure 4.1 shows the schematic of AND gate, and Figure 4.2 shows the wave simulation waveforms for it and table 4.1 shows truth table.

.Table 4.1: Truth Table of AND Gate

A	B	VOUT
0	0	0
0	1	0
1	0	0
1	1	1

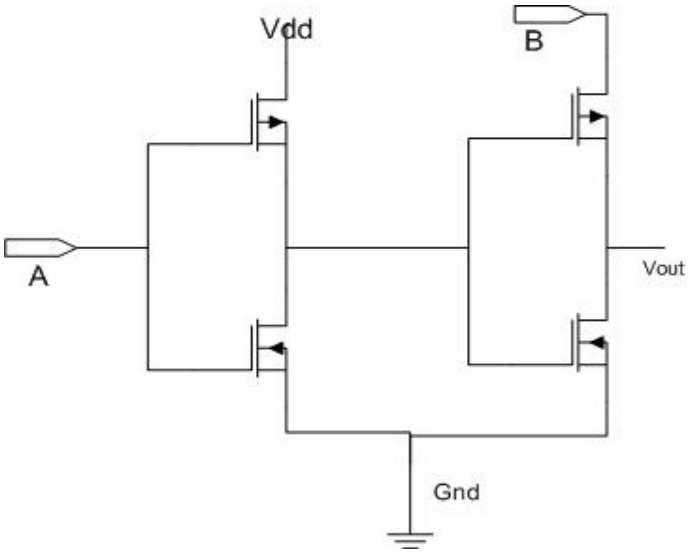


Figure 4.1: Schematic of AND Gate

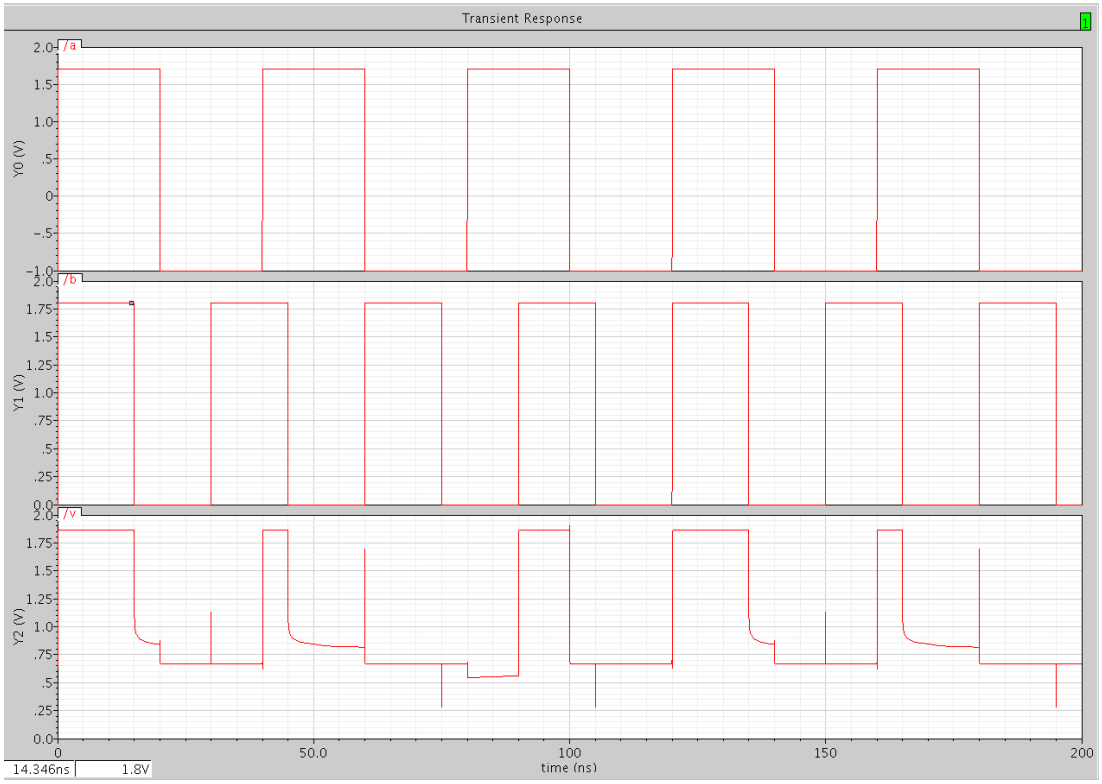


Figure 4.2: Simulation for AND Gate

4.3 PARTIAL PRODUCT ADDITION

The second step in the design of multiplier is to add the partial products generated in previous step. Various algorithms are present to add these partial terms, here we would use Wallace tree algorithm. So addition of partial product terms in this algorithm can be divided into two steps:-

1. Reduction of Partial Products
2. Final addition of Reduced Product

Reduction of partial products can be done with the use of compressors; we have discussed two compressors in chapter 2, now we would discuss their design.

4.3.1 [3: 2] COMPRESSOR

As explained in chapter 2, [3:2] compressor is basically a 1-bit full adder considering that A and B are the input bits to be added, C is the carry input, SUM is the sum output and CARRY is the carry output, the truth table of the full-adder cell is shown in Table 4.2.

Table 4.2: Truth Table of Full Adder

A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Following Boolean functions that full adder has to perform can be depicted from table 4.1

$$SUM = A \oplus B \oplus C \quad (4.1)$$

$$CARRY = AB + BC + AC \quad (4.2)$$

As discussed in chapter 3, we can design this adder using any of the logic style like CMOS, PTL, TGL, CPL or GDI, but as the comparison shows that adder designed using gate diffusion input requires least number of transistors without compensating on the output voltage swing. We have used GDI logic style to design our basic cell. GDI is a logic style similar to CMOS but consumes low power and requires less number of transistors as compared to CMOS style. So we can design a full adder using GDI and transmission gates using 24 transistors as shown in Figure 4.3.

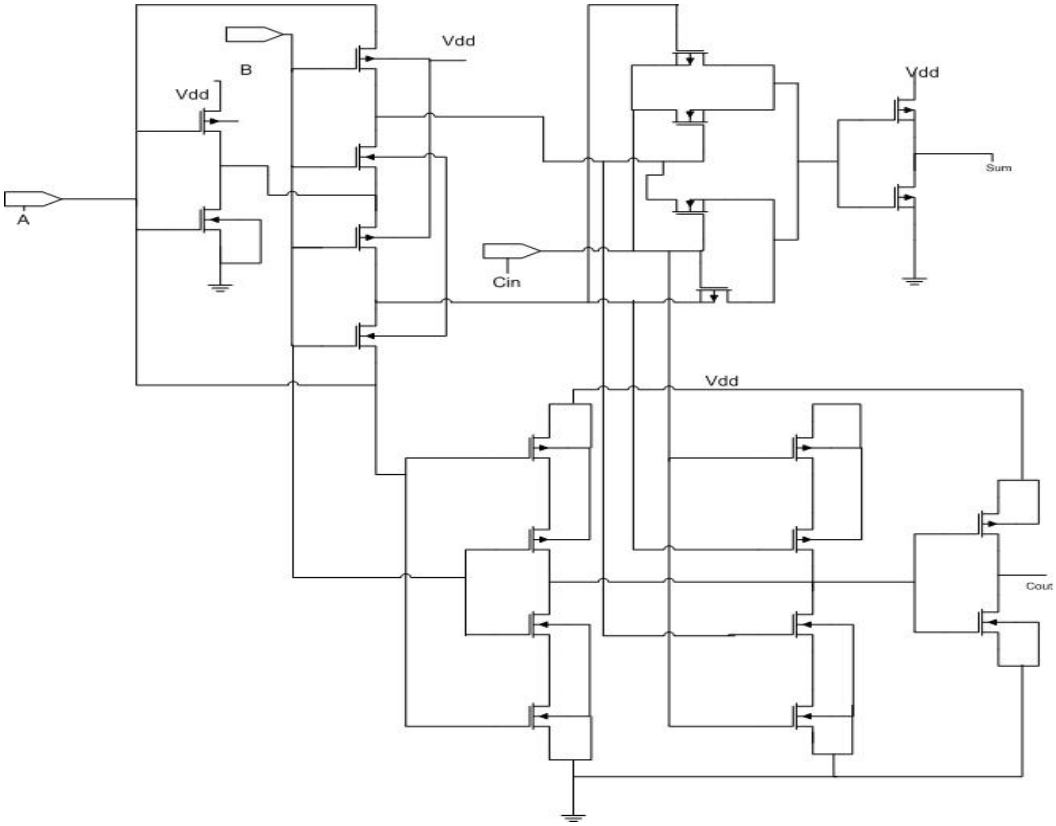


Figure 4.3: GDI Full Adder

As mentioned, one important requirement of full adder cells especially at low voltage is to provide enough driving power to the following circuits. The drivability is ensured by the full signal swing and decoupling of inputs and outputs (at least one inverter per cell) so that the adder cell can be cascaded arbitrarily and work reliably in any circuit configuration. So the second stage of

full adder cells which generate Sum and Carry must have enough drivability. In this full adder cell, a circuit based on complementary CMOS Logic style is used. Its robustness against voltage scaling and transistor sizing enables it to operate reliably at low voltage. Also, the output inverter guarantees sufficient drive to the cascaded cells. Figure 4.4 shows the simulation results of full adder, the design is simulated using analog design environment by cadence in 0.18µm technology.

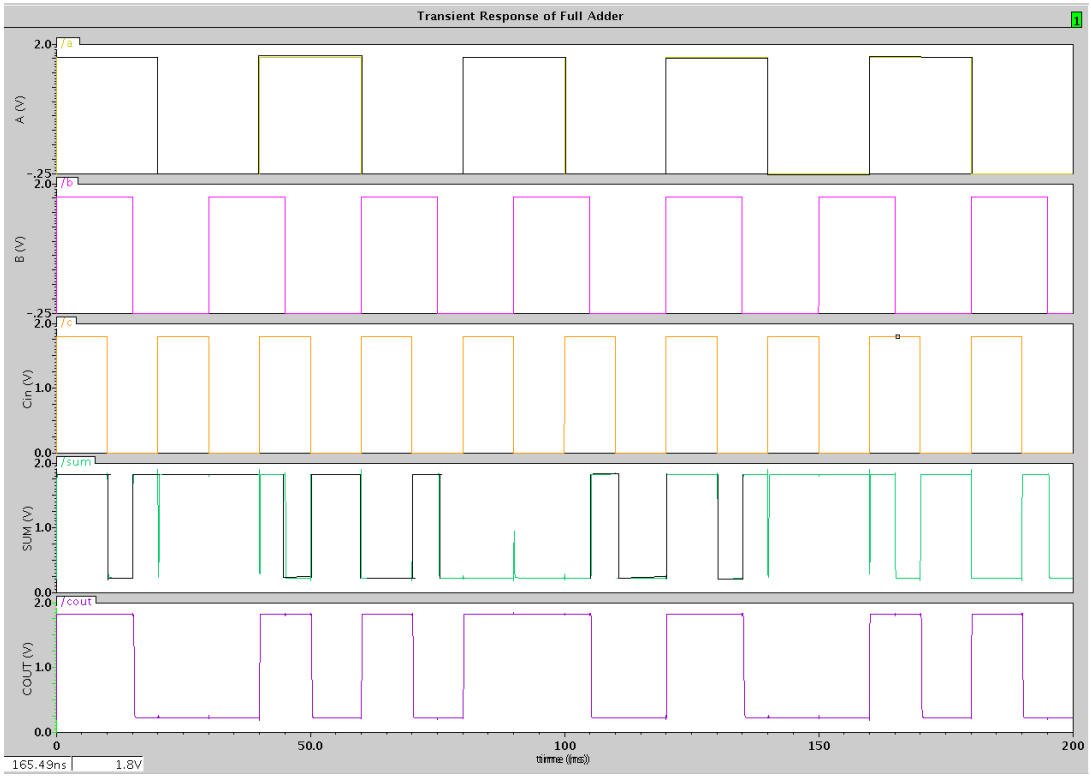


Figure 4.4: Simulation of GDI Full adder

4.3.2 [4:2] COMPRESSOR

[4:2] compressor is mainly a [5:3] compressor. It takes five inputs and produces three outputs. Among the five inputs four are the terms to be added and fifth one is the carry of previous stage and among the output it produces a sum bit and two carry bits both of equal weight. Hence we see that it also plays an important role in optimizing the partial products. Talking about design of this compressor it consist of two 1-bit full adders placed in series and OR gate, first full adder takes in the three inputs to be compressed and the next full adder takes in the fourth input and sum of the first adder, carryin bit is grounded. The carry out from two

adders are fed into an OR gate to evaluate the final carry signal. Sum of the second adder is treated as output of the compressor. Figure 4.5 & 4.6 shows block diagram of [4:2] compressor and simulation waveforms respectively.

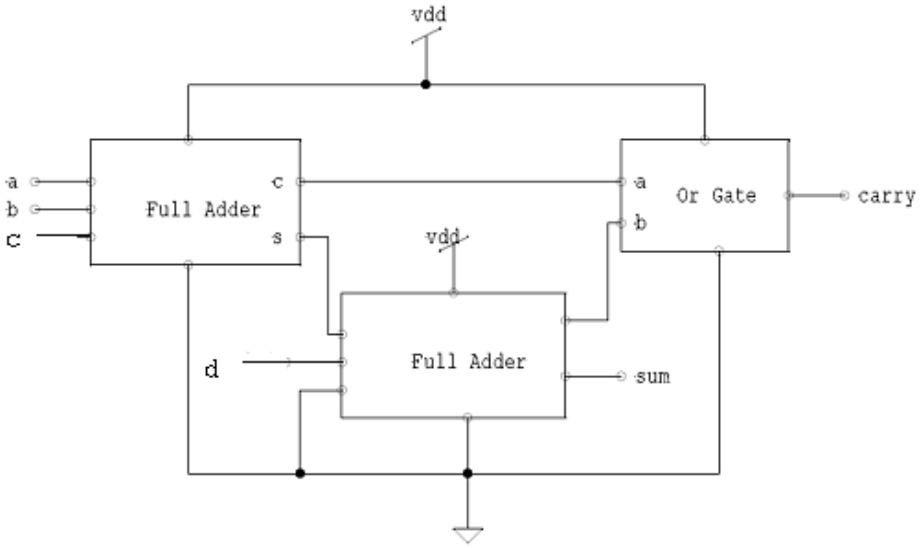


Figure 4.5: Schematic of [4:2] compressor

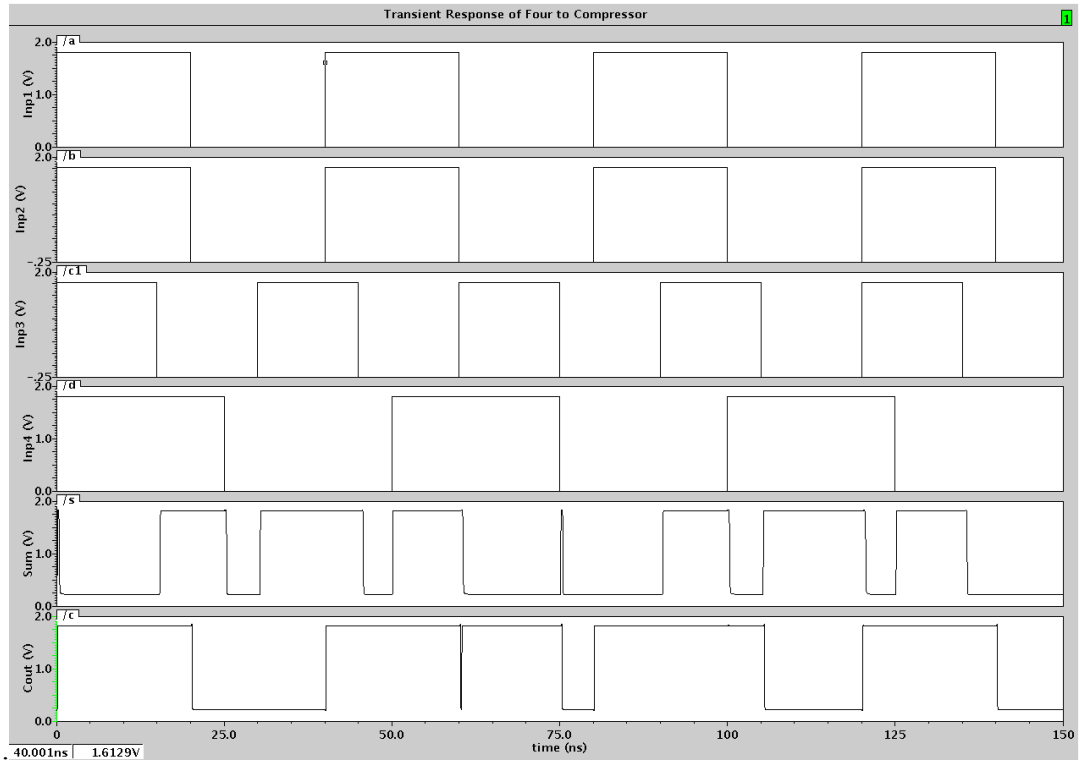


Figure 4.6: Transient response of [4:2] Compressor

After reduction of partial products as per Wallace tree multiplier algorithm, discussed in chapter 2, we would finally reduce the partial products to two bits in a column. Finally we would add up these bits to get the final product. Now to add these bits we can use any of the following adders of required width.

1. Ripple Carry Adder
2. Carry Propagate Adder
3. Carry Select Adder

Choice of adder differs from designer to designer depending upon his specifications. If a designer is aiming at low complexity and minimum area he goes for ripple carry adder but it lacks in performance in terms of speed. Similarly carry select adder is good at speed but requires large area and huge complexity in design. If we compensate on some aspects than we find that carry propagate adder is a good choice with high speed and design complexity moderately less than carry select adders. Following sections discusses the design of carry propagate adder.

4.3.3 CARRY PROPAGATE ADDER

The last stage of design consist of a carry propagate adder, it adds up all the two bits of partial product to give us the final product term. CPA consists of two following blocks:

1. Carry Propagate Block
2. Full Adder

4.3.3.1 CARRY PROPAGATE BLOCK

Carry propagate block is the one that decides whether the carry has to propagate, generate or killed. It takes in the two numbers to be added and carry in from any previous stage and produces carry_out signal as per the conditions. This block internally creates three signals propagate, generate and kill.

Propagate (Pi) = Ai xor Bi

Generate (Gi) = Ai and Bi

Kill (Ki) = Ai nor Bi

Now depending on the condition of these three signals carry_out is fed with proper value, i.e. if propagate is high carry_out is fed with carry_in, if generate is high carry_out is fed with high signal and if kill is high carry_out is made low. This can be modeled at transistor level as shown in Figure 4.7.

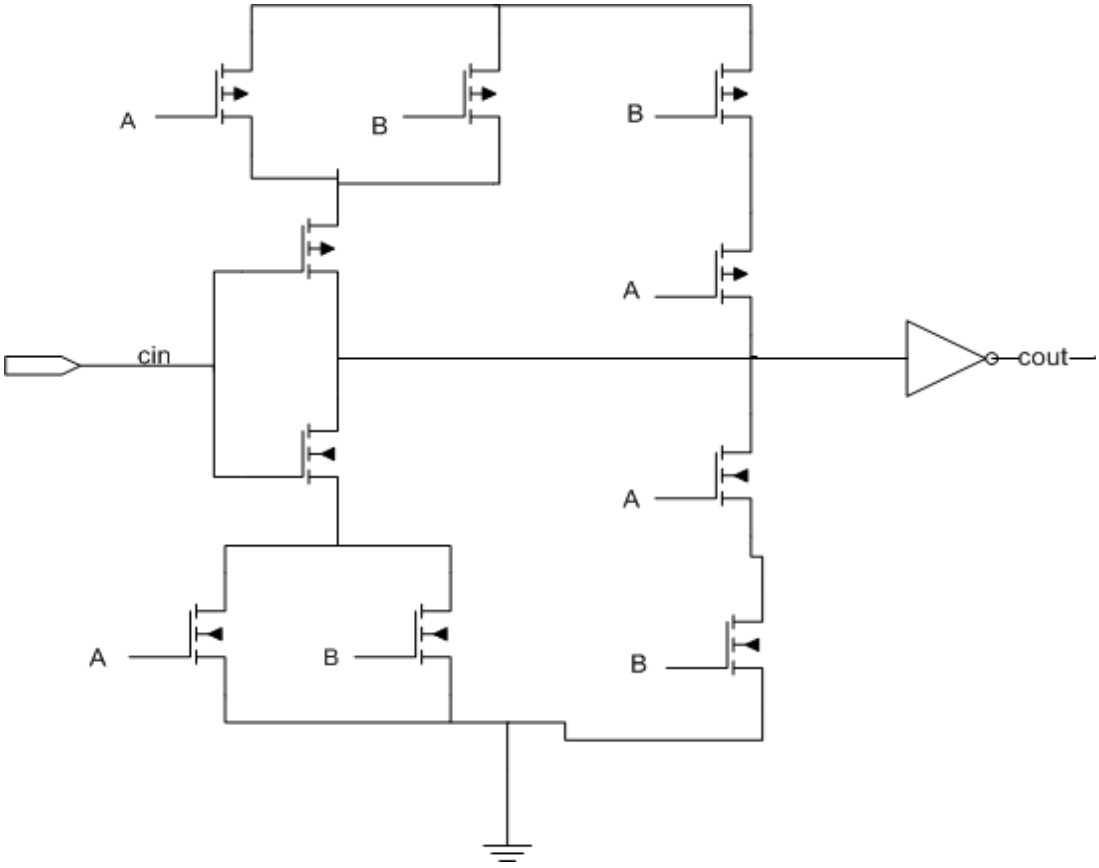


Figure 4.7: Schematic of Carry Propagate Block

As a result carry propagate block speeds up the addition process as each satge is not dependent on the previous for carry in signal. We can understand it better from the waveform of carry propagate block, so Figure 4.8, shows the simulated waveforms carry propagate block.

Design of 4-bit carry propagate adder is shown in Figure 4.9. We can design any n-bit CPA adder by moving on the same lines. In this design a 12-bit carry

propagate adder has been designed for 8-bit multiplication. Figure 4.10 shows the simulation waveforms for carry propagate adder.

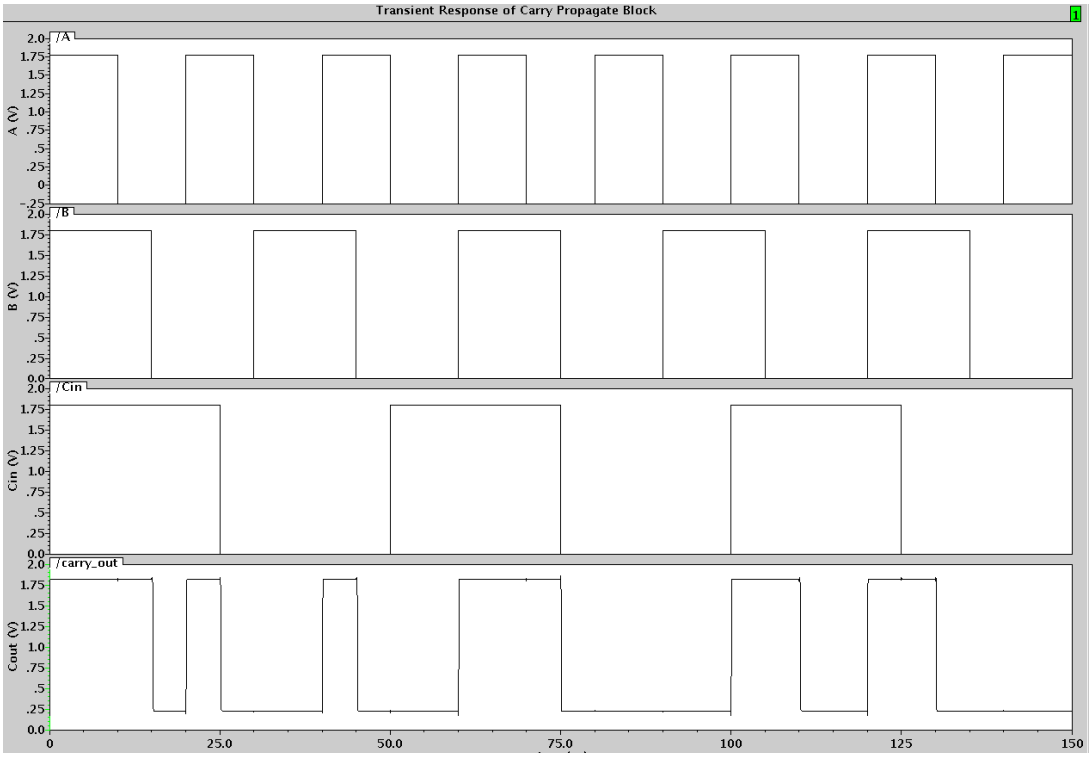


Figure 4.8: Simulated waveform for Carry Propagate Block

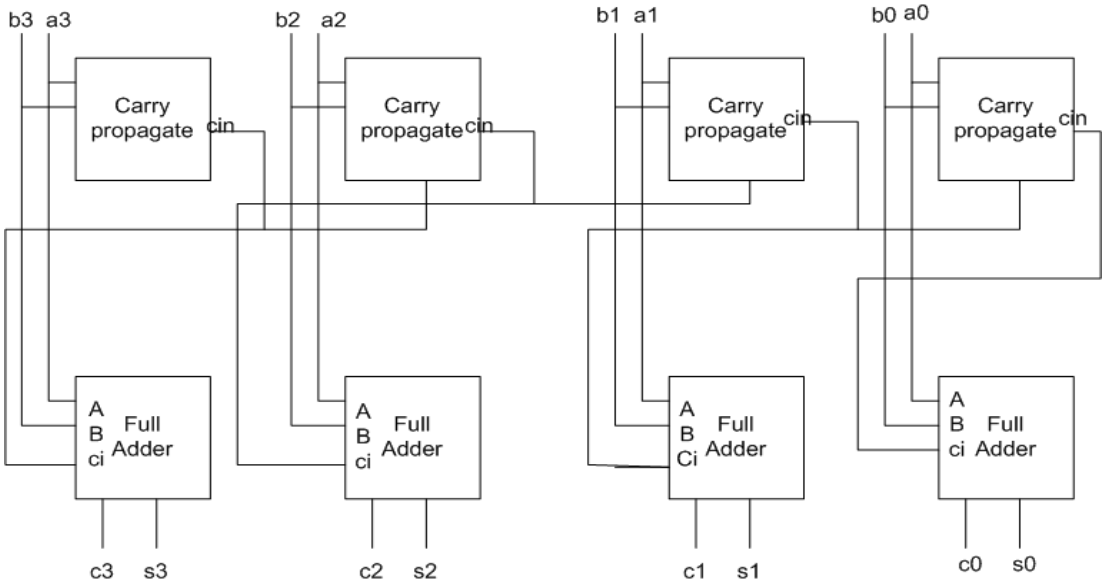
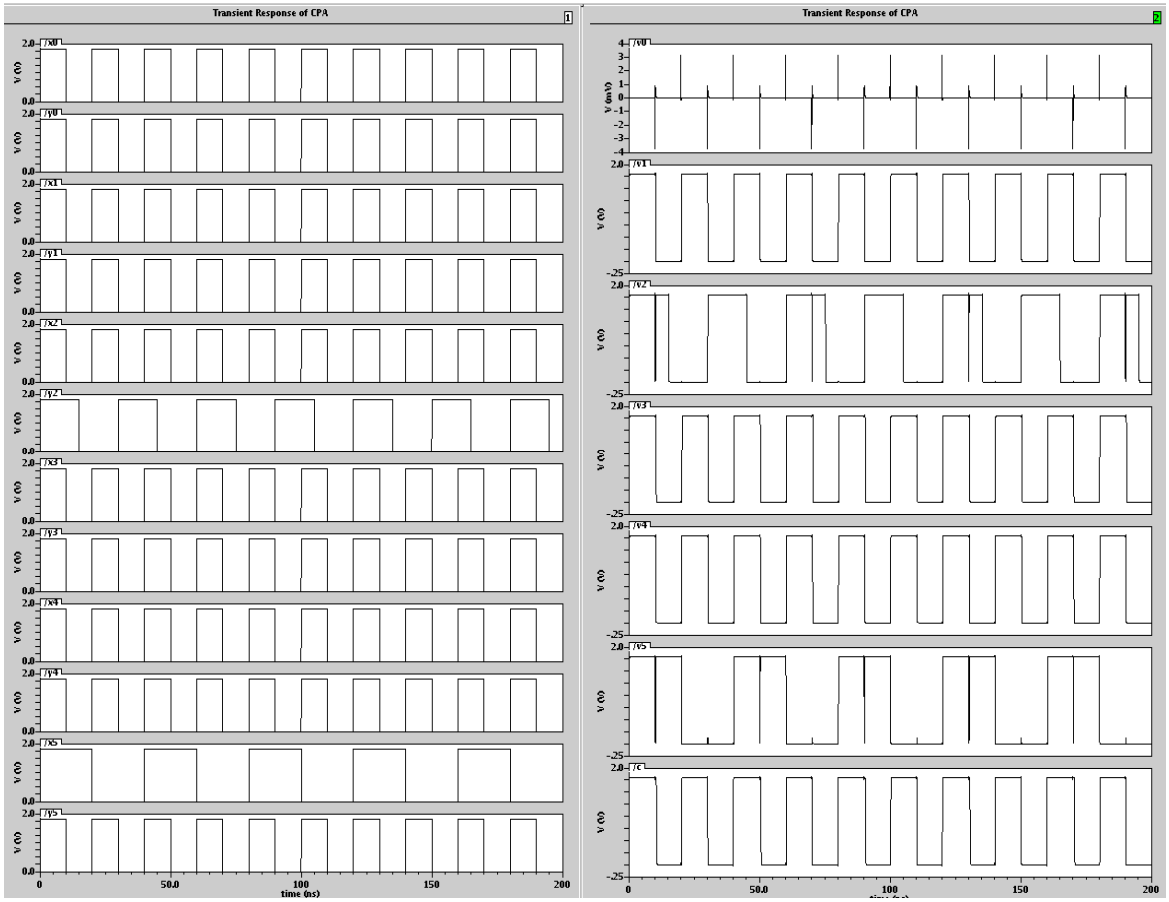


Figure 4.9: Schematic of Carry Propagate Adder



(a) **(b)**

Figure 4.10: (a) Carry Propagate Adder Input waves
(b) Carry Propagate Adder Outputs waves

So now after designing each block we now can design a multiplier as per our requirement, we discuss the design of 8-bit multiplier further though the schematic and waveform results for 4-bit are also shown.

As per the first step we would generate partial products, so 64 AND gates are place as shown in Figure 4.11, as a part of second step compressors are placed and feed them with the required partial product input, this reduction would be completed in two steps as shown in block diagram below. Finally Carry Propagate Adder is used to add up the reduced partial products, it require a 12-bit CPA in last stage. Figure 4.12 shows the simulated waveforms form 8-bit multiplier.

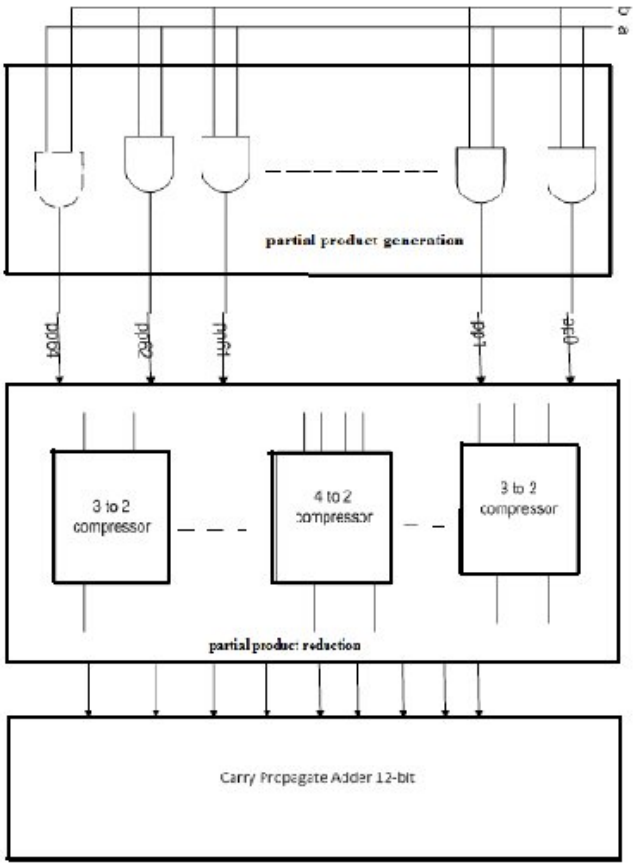


Figure 4.11: Block Diagram of Multiplier

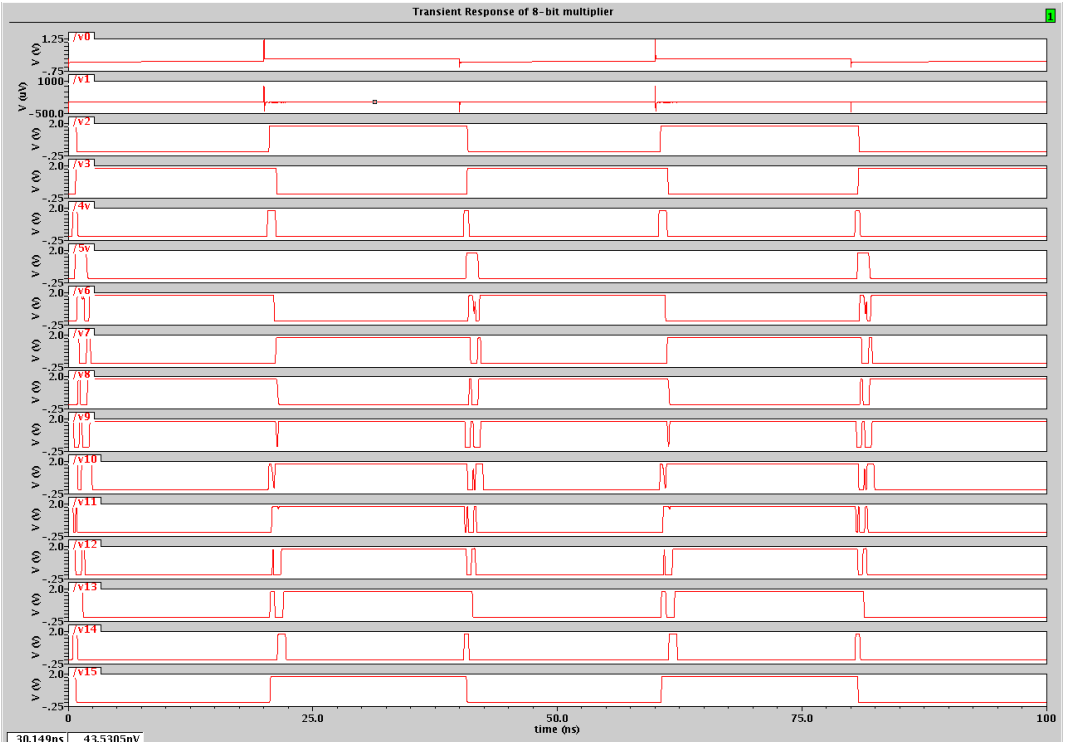


Figure 4.12: Simulation Waveform of 8-bit multiplier

4.4 POWER ANALYSIS

Power consumption is one of the basic parameters of any kind of integrated circuit (IC). Power and performance are always traded off to meet the system requirements. Power has a direct impact on the system cost. If an IC is consuming more power, then a better cooling mechanism would be required to keep the circuit in normal conditions. Otherwise, its performance is degraded and on continuous use it may be permanently damaged.

4.4.1 POWER AND ENERGY DEFINITIONS

It is important at this point, to distinguish between energy and power. The power consumed by a device is, by definition, the energy consumed per unit time. In other words, the energy (E) required for a given operation is the integral of the power (P) consumed over the operation time (T_{op}), hence,

$$E = \int_0^T P(t)dt \tag{4.3}$$

Here, the power of digital CMOS circuit is given by

$$P = C\, V_{DD}\, V_S f \tag{4.4}$$

Where, C is the capacitance being recharged during a transition. V_{DD} is the supply voltage, V_s is the voltage swing of the signal, and f is the clock frequency. If it is assumed that an operation requires n clock cycles, T_{op} can be expressed as n / f. Hence, Equation (4.3) can be rewritten as

$$E = n\, C\, V_{DD}\, V_S \tag{4.5}$$

It is important to note that the energy per operation is independent of the clock frequency. Reducing the frequency will lower the power consumption but will not change the energy required to perform a given operation [1]. Since the energy consumption is what determines the battery life, it is imperative to reduce the energy rather than just the power. It is, however important to note that the power is critical for heat dissipation considerations.

4.4.2 OVERVIEW OF POWER DISSIPATION

It is more convenient to talk about power dissipation of digital circuits at this point. Although power depends greatly on the circuit style, it can be divided, in general, into static and dynamic power. The static power is generated due to the DC bias current, as is the case in transistor-transistor-logic (TTL), emitter-coupled logic (ECL), and N-type MOS (NMOS) logic families, or due to leakage currents. In all of the logic families except for the push-pull types such as CMOS, the static power tends to dominate. That is the reason why CMOS is the most suitable circuit style for very large scale integration (VLSI).
CMOS is the logic family preferred in many designs due to following reasons:-

(a) Impeccable noise margins.

(b) Perfect logic levels.

(c) Negligible static power dissipation.

(d) Gives good performance in most cases.

(e) Easy to get a functional circuits.

(f) Lot of tools available to automate the design process.

The power consumed when the CMOS circuit is in use can be decomposed into two basic classes: static and dynamic.

4.4.2.1 STATIC POWER

The static or steady state power dissipation of a circuit is expressed by the following relation [1]

$$P_{stat} = I_{stat} V_{DD} \tag{4.6}$$

Where, I_{stat} is the current that flows through the circuit when there is no switching activity. Ideally, CMOS circuits dissipate no static (DC) power since in the steady state there is no direct path from V_{DD} to ground as PMOS and NMOS transistors are never on simultaneously. Of course, this scenario can never be realized in practice since in reality the MOS transistor is not a perfect switch. Thus, there

will always be leakage currents and substrate injection currents, which will add to a static component of CMOS power dissipation. For a sub-micron NMOS device W/L = 10/ 0.5, the substrate injection current is of the order of 1- 100 μA for a V_{DD} of 5 V [2]. Another form of static power dissipation occurs for the so-called Ratioed logic. Pseudo-NMOS is an example of a Ratioed CMOS logic family. In this, the PMOS pull-up is always on and acts as a load device for the NMOS pull-down network. Therefore, when the gate output is in low-state, there is a direct path from V_{DD} to ground and the static currents flow. In this state, the exact value of the output voltage depends on the ratio of the strength of PMOS and NMOS networks – hence the name. The static power consumed by these logic families can be considerable. For this reason, logic families such as this, which experience static power consumption, should be avoided for low-power design. With that in mind, the static component of power consumption in low-power CMOS circuits should be negligible and the focus shifts primarily to dynamic power consumption.

4.4.2.2 DYNAMIC POWER

The dynamic component of power dissipation arises from the transient switching behavior of the CMOS device. At some point during the switching transient, both the NMOS and PMOS devices will be turned on. This occurs for gate voltages between V_{tn} and V_{DD} - $V_{tp.}$ During this time, a short-circuit exists between V_{DD} and ground and the currents are allowed to flow. A detailed analysis of this phenomenon by Veendrick reveals that with careful design of the transition edges, this component can be kept below 10-15% of the total power [2]; this can be achieved by keeping the rise and fall times of all the signals throughout the design within a fixed range(preferably equal). Thus, although short circuit dissipation cannot always be completely ignored, it is certainly not the dominant component of power dissipation in well-designed CMOS circuits. Instead, dynamic dissipation due to capacitance charging consumes most of the power. This component of dynamic power dissipation is the result of charging and discharging of the parasitic capacitances in the circuit. The situation is modeled in Figure 4.13, where the parasitic capacitances are lumped at the output in the

capacitor *C*. Consider the behavior of the circuit over one full cycle of operation with the input voltage going from V_{DD} to ground and back to V_{DD} again. As the input switches from high to low, the NMOS pull-down network is cut-off and PMOS pull-up network is activated charging load capacitance *C* up to V_{DD}.

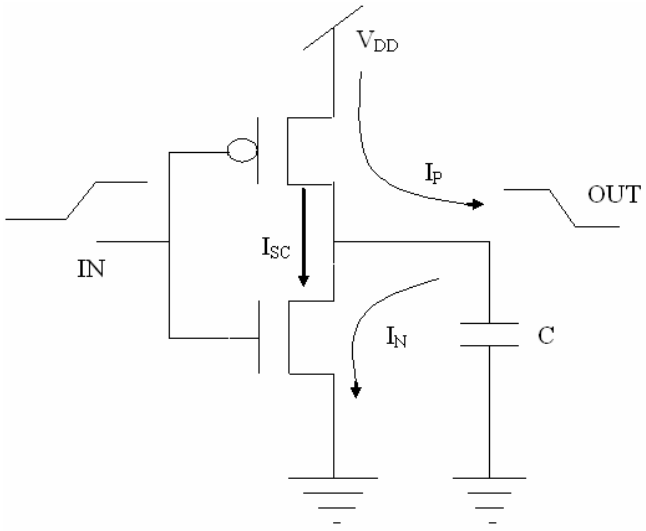


Figure 4.13: CMOS Inverter for Power Analysis

This charging process draws energy equal to CV_{DD}^{2} from the power supply. Half of this is dissipated immediately in the PMOS transistors, while the other half is stored on the load capacitance. Then, when the input returns to V_{DD}, the process is reversed and the capacitance is discharged, its energy being in the NMOS network. In summary, every time a capacitive node switches from ground to V_{DD} (and back to ground), energy of CV_{DD}^{2} is consumed.

This leads to the conclusion that CMOS power consumption depends on the *switching activity* of the signals involved. We can define *activity*, α as the expected number of zero to one transition per data cycle. If this is coupled with the average data rate, *f*, which may be the clock frequency in a synchronous system, then the effective frequency of nodal charging is given the product of the activity and the data rate: αf. This leads to the following formulation for the average CMOS power consumption:

$$P_{dyn} = \alpha CV_{DD}^{2} f \tag{4.7}$$

This classical result illustrates that the dynamic power is proportional to the switching activity, capacitive loading and the square of the supply voltage. In CMOS circuits, this component of power dissipation is by far the most important accounting for at least 90% of the total power dissipation [2].

So, to reduce the power dissipation, the circuit designer can minimize the switching event, decrease the node capacitance, reduce the voltage swing or apply a combination of these methods. Yet, in all these cases, the energy drawn from the power supply is used only once.

4.4.3 HOW TO MEASURE POWER DISSIPIATION IN A DIGITAL CIRCUIT

As explained above power dissipation is an important component of digital design. So it becomes necessary to make accurate and thorough measurements for power. As per equation (4.3) total power is integral of the total current drawn by all the PMOS in the circuit multiplied by Vdd and frequency. So, to measure power we have to plot power curves, following steps are followed to plot a power curve for digital circuit.

- Place a 1Ω resistance between the supply and circuit to measure the total current drawn by the circuit.
- Multiply the current with power supply i.e. 1.8 V
- Integrate the value obtained in the above step.
- Finally we divide the integral value by the time period to get power- time curve for a particular frequency.
- Measure the power at that particular frequency, and repeat the above steps for different frequencies.

Figure 4.14 shows each of the steps in pictorial view.

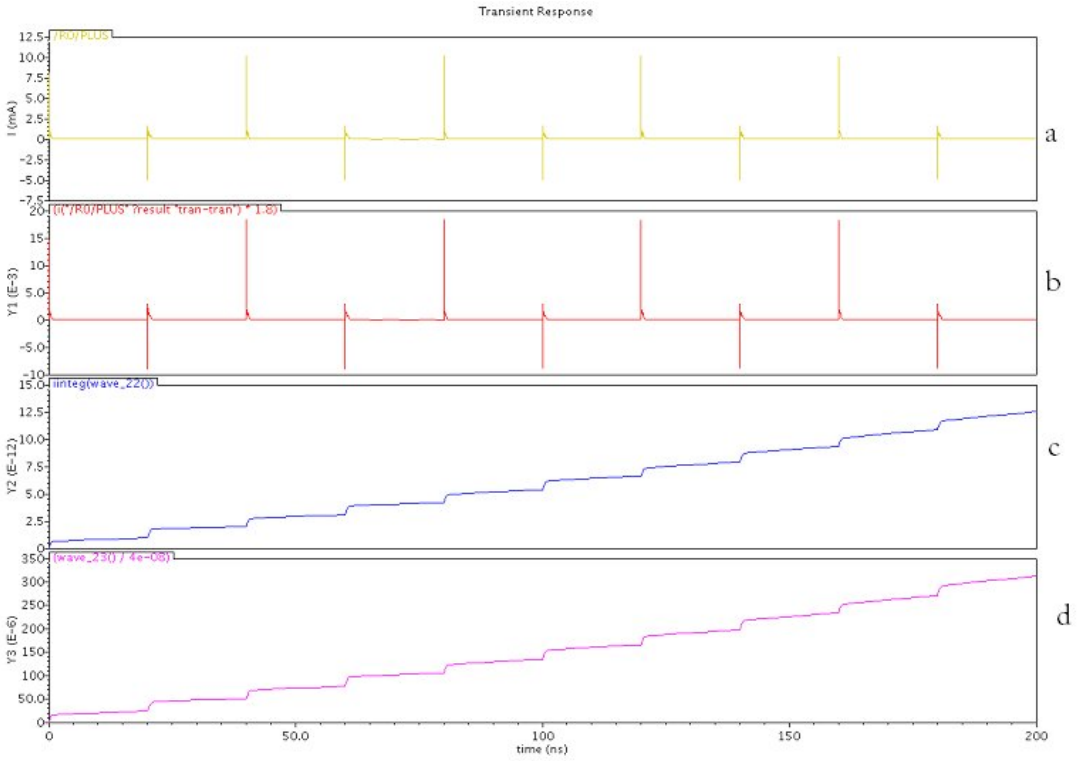


Figure 4.14: (a) Current drawn by the circuit (b) current multiplied by vdd (c) Integeral curve (d) Power curve at specific frequency

4.4.3.1 POWER ANALYSIS FOR FULL ADDER

Moving on the lines explained above measure the power consumption of full adder designed using different logic styles i.e. CMOS, GDI, DCVS at different frequencies. Table 4.3 shows a comparison of power among three adders and Figure 4.15 shows the frequency power curve for them.

Table 4.3: Power Dissipation by Full Adder 1-bit

Frequency (MHz)	Gate Diffusion Input Adder (W)	CMOS Adder (W)	DCVS Adder (W)
200	7.251×10^{-6}	8.35×10^{-6}	13.251×10^{-6}
100	3.87×10^{-6}	4.29×10^{-6}	9.215×10^{-6}
50	3.18×10^{-6}	3.65×10^{-6}	6.517×10^{-6}
33.33	2.84×10^{-6}	3.45×10^{-6}	5.256×10^{-6}
25	2.62×10^{-6}	3.19×10^{-6}	4.182×10^{-6}

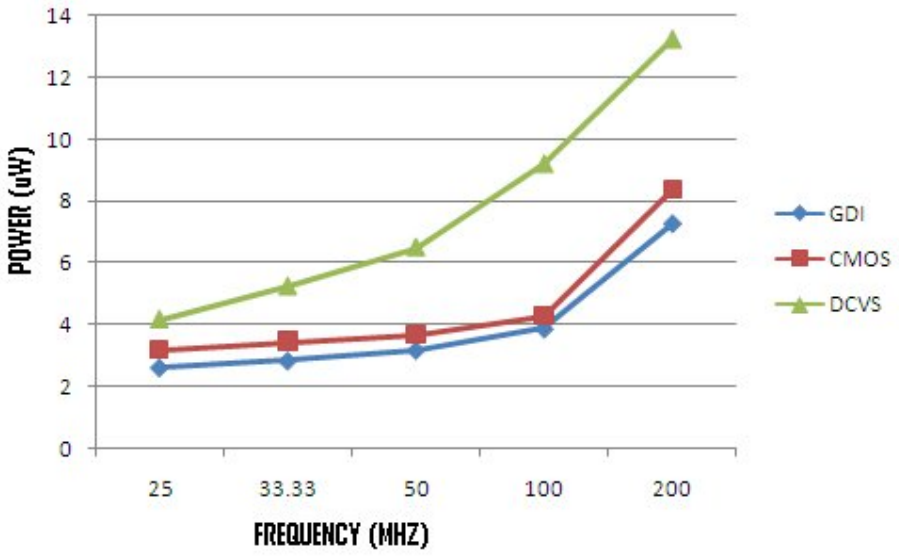


Figure 4.15: Power curves for adder

4.4.3.2 POWER ANALYSIS FOR MULTIPLIER 4-BIT

Moving on the lines explained above measure the power consumption of multiplier 4-bit, designed using different logic styles i.e. CMOS, GDI, DCVS at different frequencies. Table 4.4 shows a comparison of power and Figure 4.16 shows the frequency power curve for them.

Table 4.4: Power Dissipation of Multiplier 4-bit

Frequency (MHz)	Gate Diffusion Input Multiplier (W)	CMOS Multiplier (W)	DCVS Multiplier (W)
200	1.39×10^{-4}	1.45×10^{-4}	16.25×10^{-5}
100	7.14×10^{-5}	7.93×10^{-5}	11.23×10^{-5}
50	6.512×10^{-5}	6.875×10^{-5}	8.26×10^{-5}
40	4.316×10^{-5}	4.934×10^{-5}	6.48×10^{-5}
25	2.37×10^{-5}	2.879×10^{-5}	4.89×10^{-5}

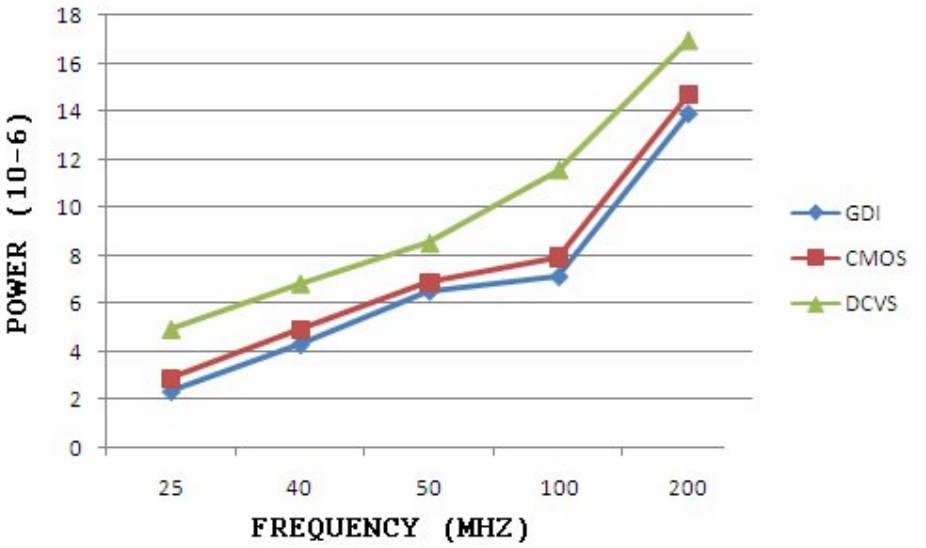


Figure 4.16: power curve for 4-bit multiplier

4.4.3.3 POWER ANALYSIS FOR MULTIPLIER 8-BIT

Moving on the lines explained above measure the power consumption of multiplier 8-bit designed using different logic styles i.e. CMOS, GDI, DCVS at different frequencies. Table 4.5 shows a comparison of power and Figure 4.17 shows the frequency power curve for them.

Table 4.5: Power Dissipation of Multiplier 8-bit

Frequency (MHz)	Gate Diffusion Input Multiplier (W)	CMOS Multiplier (W)	DCVS Multiplier (W)
200	5.82×10^{-4}	8.153×10^{-4}	9.68×10^{-4}
100	2.919×10^{-4}	4.32×10^{-4}	7.54×10^{-4}
50	1.42×10^{-4}	2.12×10^{-4}	6.34×10^{-4}
40	1.436×10^{-4}	1.93×10^{-4}	5.12×10^{-4}
25	1.292×10^{-4}	1.69×10^{-4}	4.04×10^{-4}

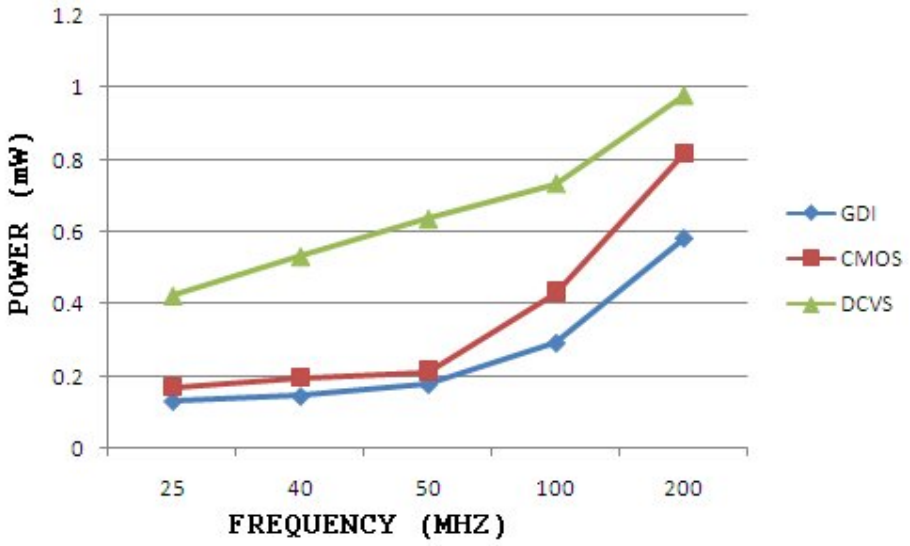


Figure 4.17: Power Curve of 8-bit multiplier

Similarly we can compare the number of transistors required in three logic styles. Figure 4.18 shows bar chart comparison.

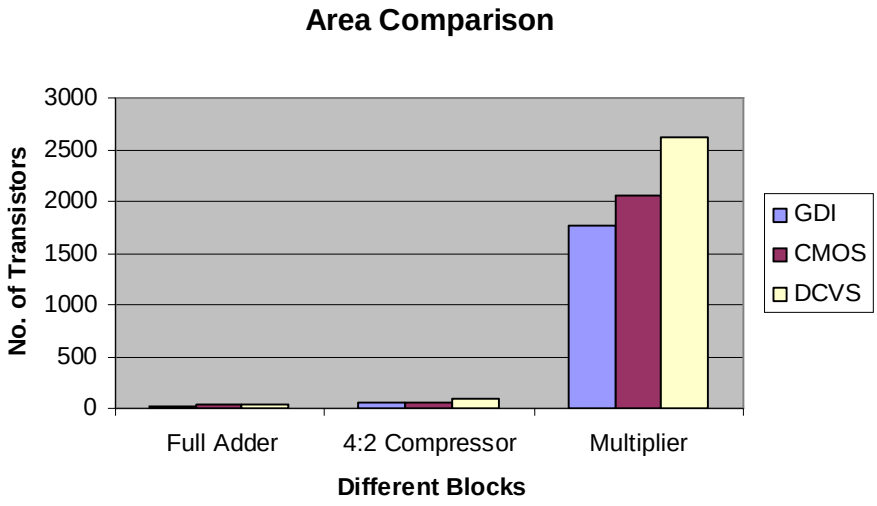


Figure 4.18: Area Comparison of Logic Styles

CHAPTER 5

PHYSICAL LAYOUT DESIGN AND POST LAYOUT SIMULATIONS

5.1 WHAT IS LAYOUT?

Integrated Circuit (IC) Layout or mask design is the representation of an integrated circuit in terms of planar geometric shapes which correspond to the patterns of metal, oxide, or semiconductor layers that make up the components of the integrated circuit. In other words, Layout is the process by which a circuit specification is converted to a physical implementation with enough information to deduce all the relevant physical parameters of the circuit. A layout engineer's job is to place and connect all the components that make up a chip so that they meet all criteria. Typical goals are performance, size, and manufacturability path on the architecture level.

5.2 THE ROLE OF LAYOUT IN THE DESIGN PROCESS

From a computer scientist's point of view, the layout process seems familiar enough. We are given a piece of *source code*, this time usually in terms of a circuit diagram, and we want to compile it to an *object code*, the physical layout of the circuit. The entire process consists of three steps, partial product generation, partial product reduction and final addition. The layout step is the last major step in the design process before testing and fabrication; it is the step which

reveals to the designer all the subtle electrical characteristics of the clean and logical digital systems.

5.3 TOLERANCES AND DESIGN RULES

The layout must pass a series of checks in a process known as Verification. The two most common checks in the verification process are Design Rule Checking (DRC), and Layout Versus Schematic (LVS). When all verification is complete, the data is translated into an industry standard format, typically GDSII, and sent to a semiconductor foundry. The process of sending this data to the foundry is called tapeout, due to the fact the data used to be shipped out on a magnetic tape. The foundry converts the data into another format and uses it to generate the photo masks used in a photolithographic process of semiconductor device fabrication.

5.4 DESIGN RULE CHECK

Design Rule Checking of Check(s) (DRC) is the area of Electronic Design Automation that determines whether a particular chip layout satisfies a series of recommended parameters called Design Rules. Design Rule Checking is a major step during Physical Verification of the design, which also involves LVS (Layout Versus Schematic) Check, XOR Checks, ERC (Electrical Rule Check) and Antenna Checks. Design rules are a set of parameters provided by the semiconductor manufacturer that enable the designer to verify the correctness of the mask set. Design rules are specific to a particular semiconductor manufacturing process. A design rule set specifies a minimum size or spacing requirements between the layers of the same type or of different types. This provides a safety margin for various process variations, to ensure that the design will still have reasonable performance after the circuit is fabricated. There is a limit to how small features the photolithographic process can generate. Generally, this *feature size* is the width of a single minimum-width polysilicon wire used as a transistor gate (since this is the most important physical dimension in determining the speed of circuit)

5.5 DESIGN RULE CHECKING (DRC) SOFTWARE

The main objective of design rule checking (DRC) is to achieve a high overall yield and reliability for the design. If the design rules are violated the design may not be functional. While design rule checks do not validate that the design will operate correctly, they are constructed to verify that the structure meets the process constraints for a given design type and process technology. DRC software usually takes as input a layout in the GDSII standard format and a list of rules specific to the semiconductor process chosen for fabrication. From these it produces a report of design rule violations that the designer may or may not choose to correct.

DRC products define rules in a language to describe the operations needed to e performed in DRC. For example, Mentor Graphics uses Standard Verification Rule Format (SVRF) language in their DRC rules files.

Some examples of DRC's in IC design includes:

- Active to active spacing
- Well to well spacing
- Minimum channel length of the transistor
- Minimum metal width
- Metal to metal spacing
- ESD and I/O rules.

5.6 SCMOS DESIGN RULES

Basic Design Rules are
(1) Size Rules.
(2) Separation Rules.
(3) Overlap Rules.

A listing of the design rules is available in the following file:
~cad / FDK / UMC_180_CMOS/RuleDecks/ Assura DRC / umc018.calibre.rules

The most important design rules are summarized below (all distances are *minimum*):

Poly silicon Region Width	0.18µm	Poly – Poly Spacing	0.18 µm
Poly silicon Gate Extension	0.22 µm	P/N Select Extension	0.23 µm
Diffusion- Diffusion Spacing	0.9 µm	Contact Extension	0.1 µm
Metal1 Width	0.24 µm	Metal1- Metal1 Spacing	0.24 µm
Metal1 Width	0.28 µm	Metal2 _ Metal2 Spacing	0.28 µm
Via Size	0.36 µm		

5.7 LAYOUT VERSUS SCHEMATIC (LVS)

The Layout Versus Schematic (LVS) is the class of electronic design automation (EDA) verification software that determines whether a particular integrated circuit layout corresponds to the original schematic of circuit diagram of the design.

A successful Design rule check (DRC) ensures that the layout conforms to the rules designed / required for faultless fabrication. However, it does not guarantee if it really represents the circuit you desire to fabricate. This is where an LVS check is used. LVS checking software recognizes the drawn shapes of the layout that represent the electrical components of the circuit, as well as the connections between them. The software then compares them with the schematic or circuit diagram. In most cases the layout will not pass LVS the first time requiring the layout engineer to examine the LVS software's reports and make changes to the layout.

5.8 PHYSICAL LAYOUT DESIGN OF DIFFERENT BLOCKS OF MULTIPLIER

The physical layout design of different cells based on GDI CMOS logic is done in standard UMC 0.18um CMOS technology. For this project work, Cadence Corporation Ltd, Assura was used for the design of physical layout along with spectra simulator for validating the physical layout designs. This section discusses the various layout designs of different blocks of multiplier and the LVS program was made to run for the comparison of the schematic to the physical layout structures. It will use both the extracted view and the schematic view of the physical layout.

5.8.1 LAYOUT CELL DESIGN OF FULL ADDER

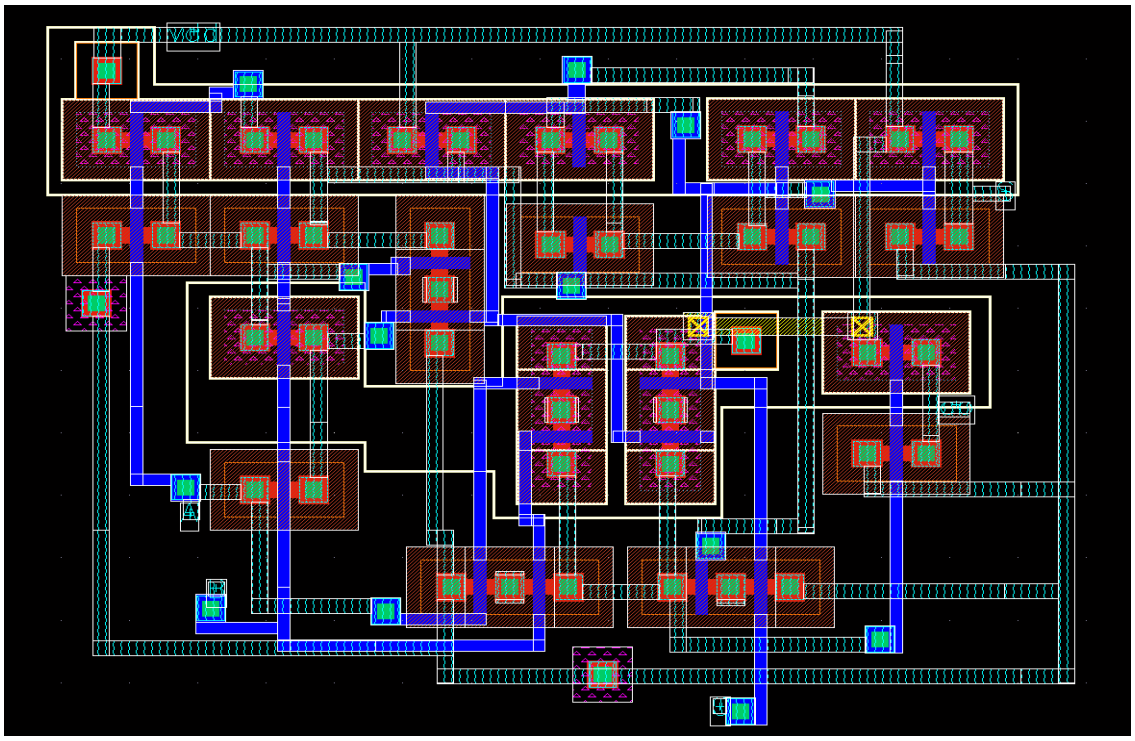


Figure 5.1: Layout of Adder 1-bit

5.8.2 LAYOUT CELL DESIGN OF FOUR TO TWO COMPRESSORS

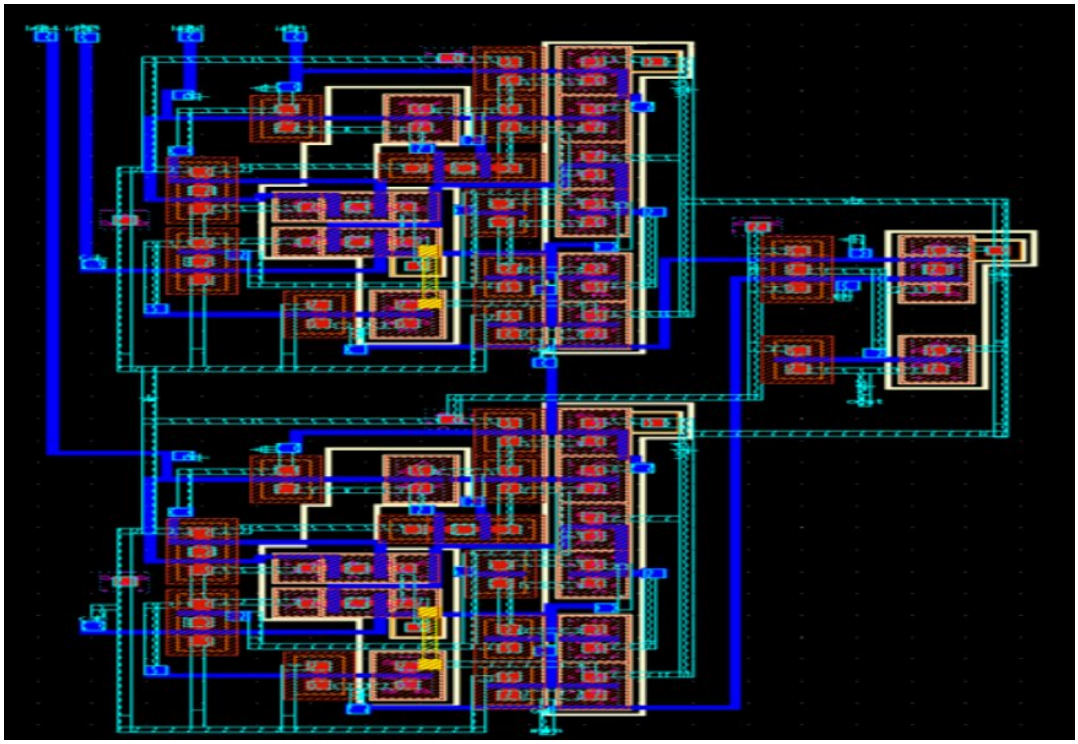


Figure 5.2: Layout of [4:2] Compressor

5.8.3 LAYOUT CELL DESIGN OF CARRY PROPAGATE ADDER

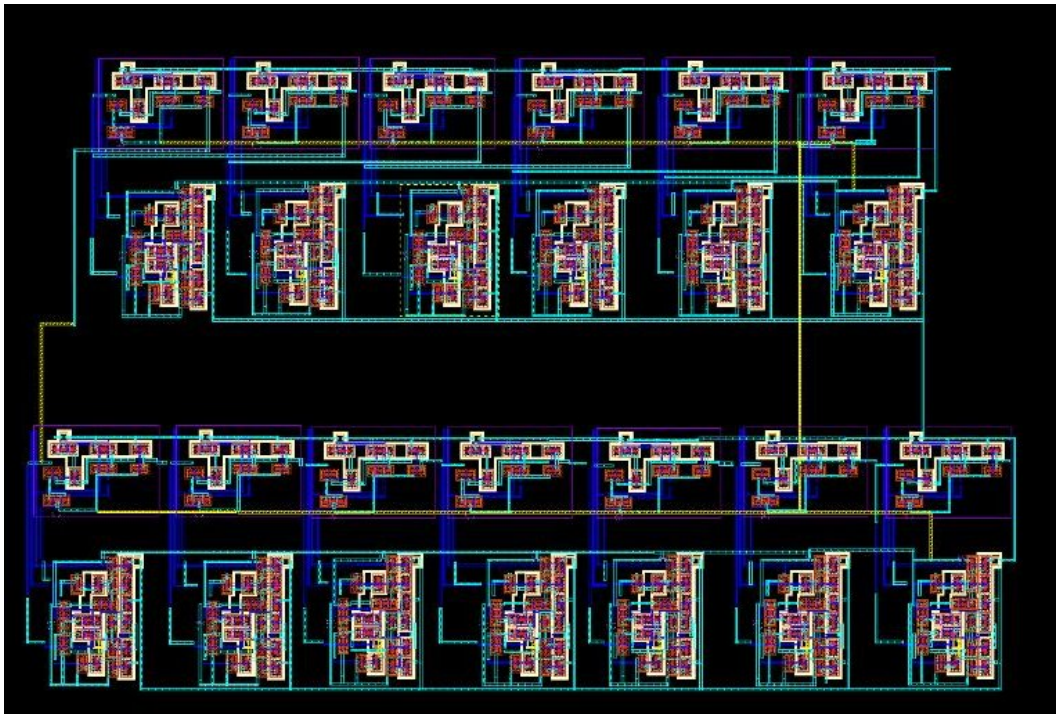


Figure 5.3: Layout of carry propagate adder

5.8.4 LAYOUT CELL DESIGN OF MULTIPLIER 4-BIT

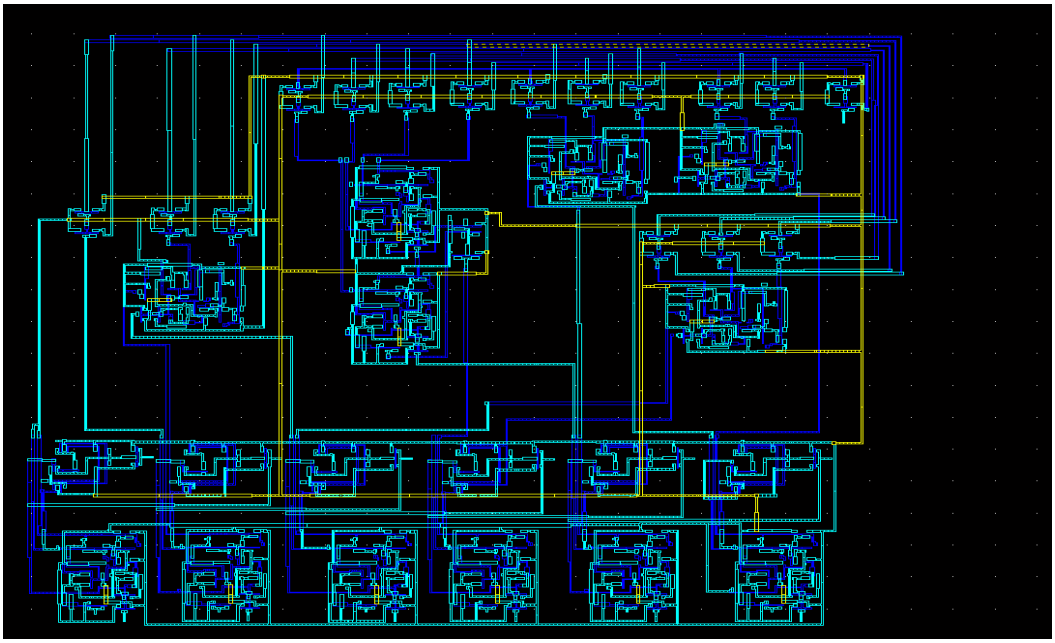


Figure 5.4: Layout of multiplier 4-bit

5.8.5 LAYOUT CELL DESIGN OF 8-BIT MULTIPLIER

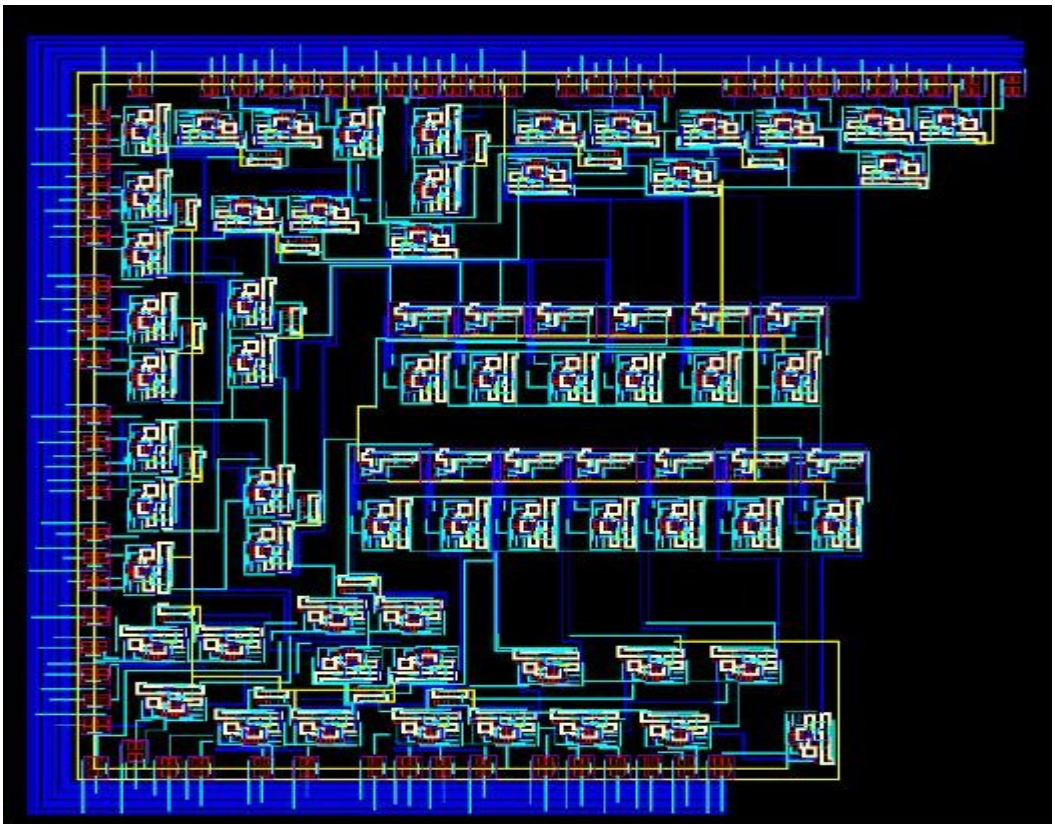


Figure 5.5: Layout of multiplier 8-bit

5.9 RC EXTRACTION OF LAYOUT DESIGNS

After the physical layout is matched with schematic, RC extraction of the design is carried out to find parasitic (resistance and capacitance) due to wiring and embedded into the design to carry out post layout simulations.

5.9.1 AV_EXTRACTED VIEW OF FULL ADDER

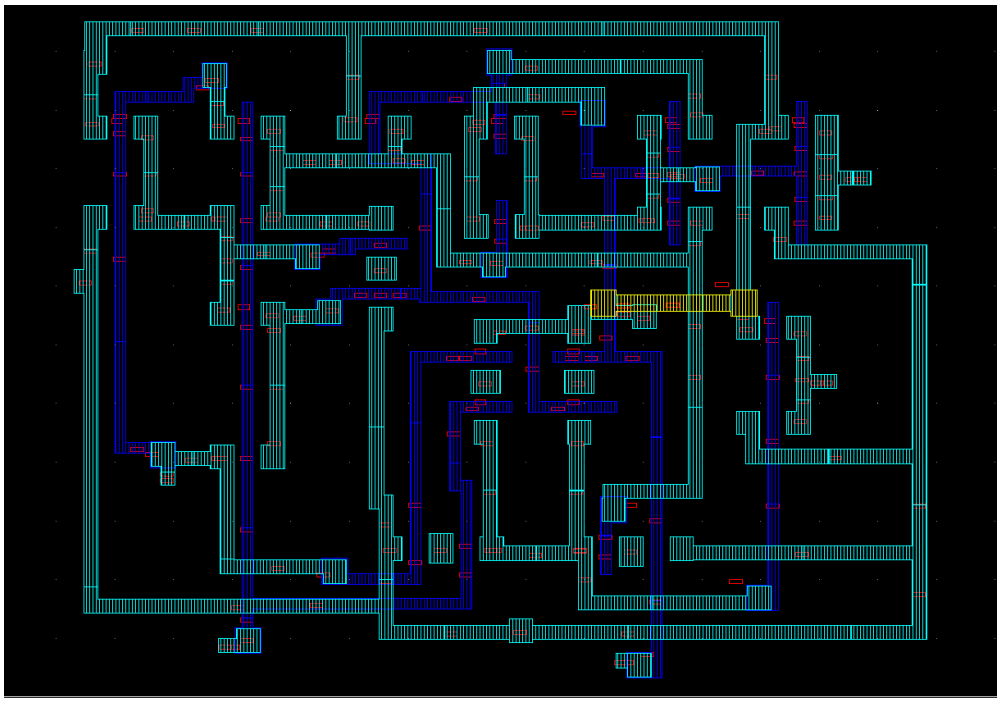


Figure 5.6: AV_Extracted of full adder

5.9.2 AV_EXTRACTED VIEW OF 4-BIT MULTIPLIER

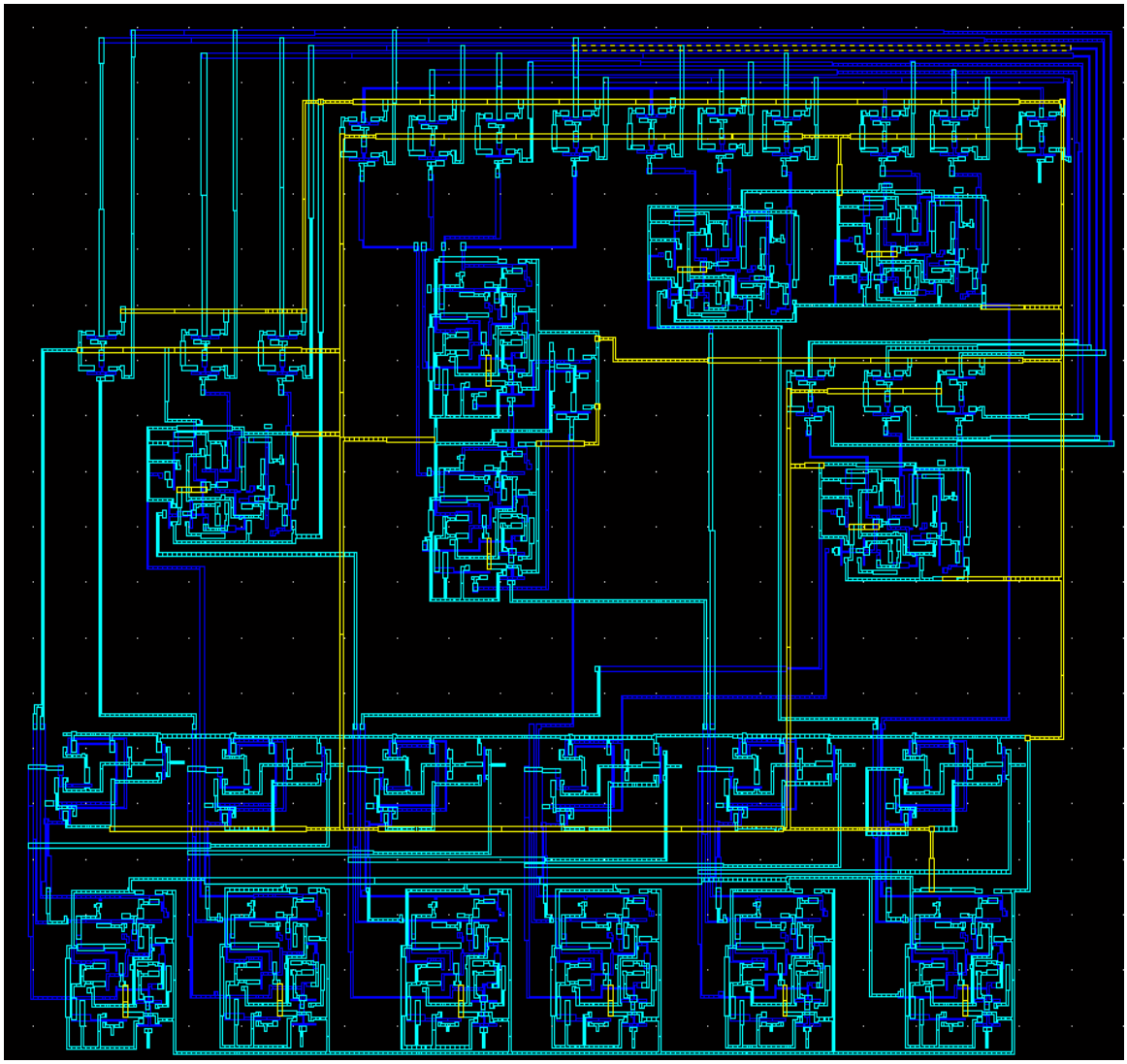


Figure 5.7: Av_Extracted of multiplier 4-bit

5.9.3 AV_EXTRACTED OF 8-BIT MULTIPLIER

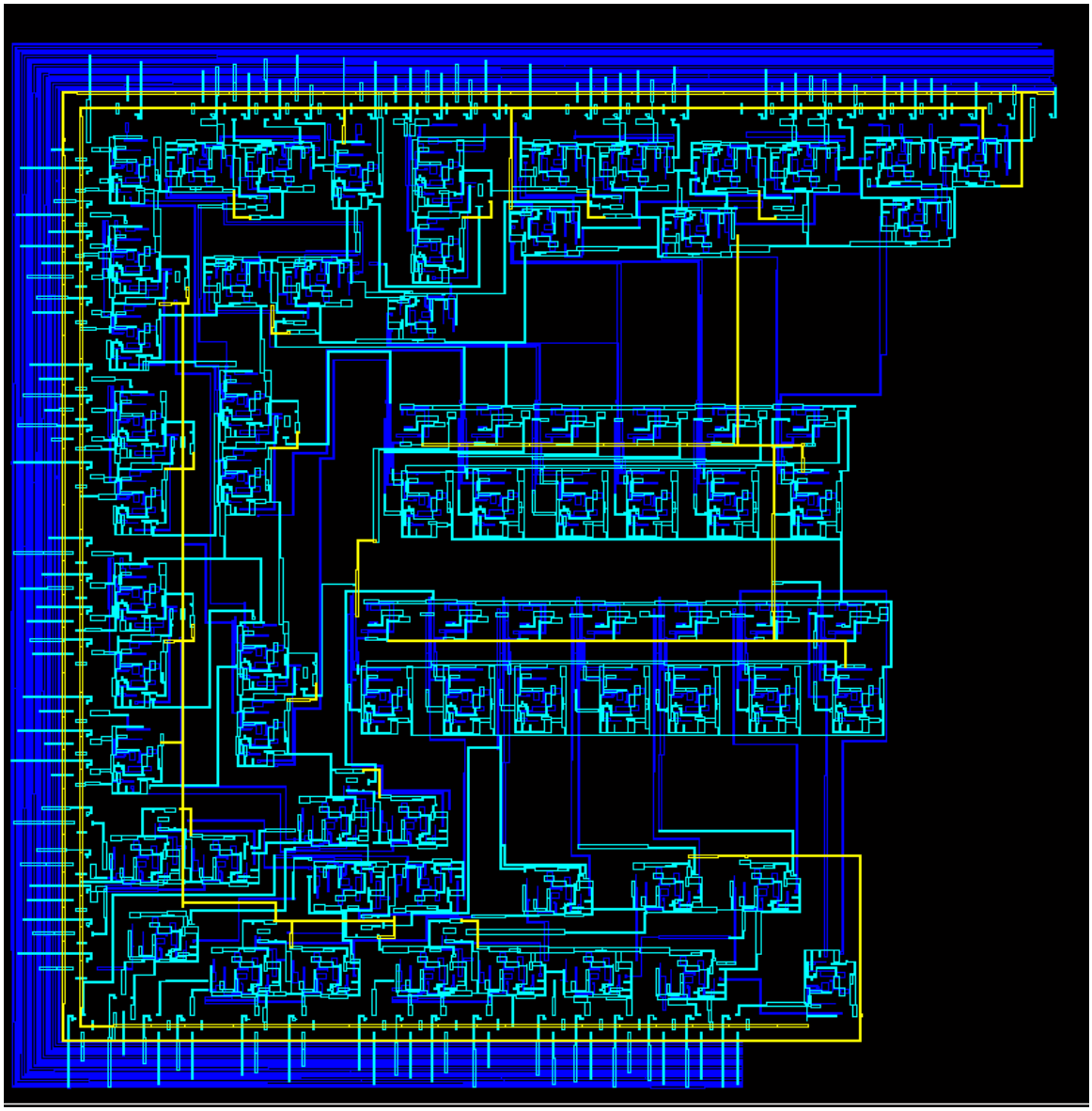


Figure 5.8: Av_Extracted of 8-bit multiplier

5.10 POST LAYOUT SIMULATIONS

After the physical layout is matched with schematic and RC extraction of the parasitic, Post Layout Simulations are carried out with parasitic extracted in RC extraction embedded into the schematic.

5.10.1 POST LAYOUT SIMULATION OF FULL ADDER

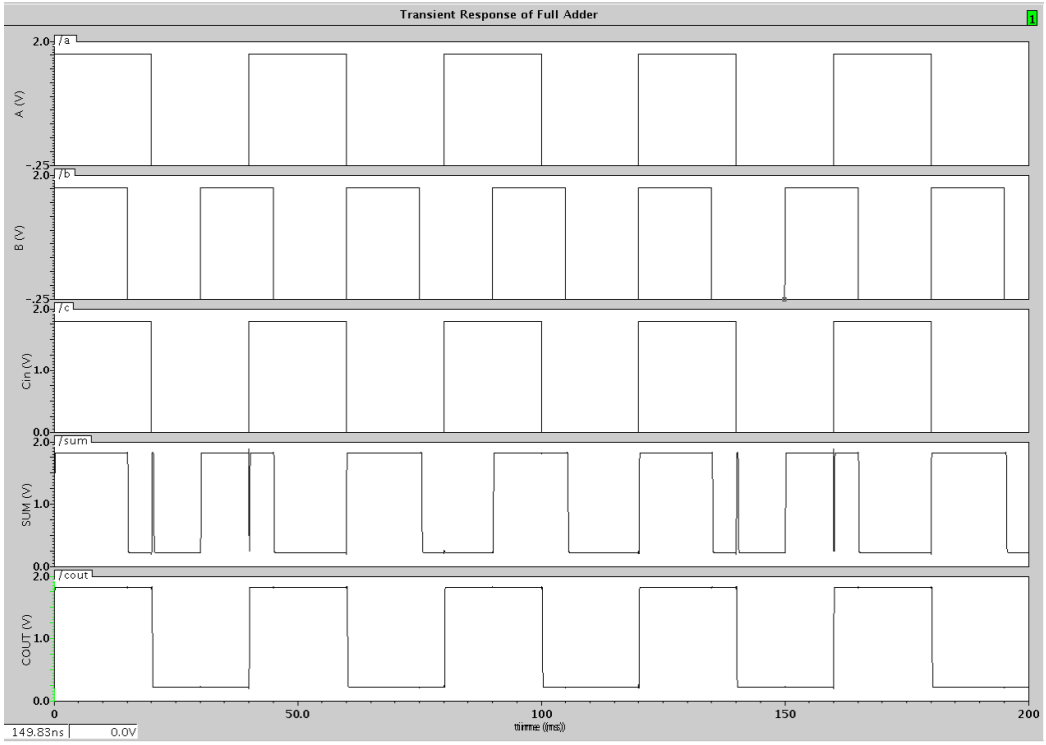


Figure 5.9: Post Layout Simulation of Full Adder

5.10.2 POST LAYOUT SIMULATION OF 4-BIT MULTIPLIER

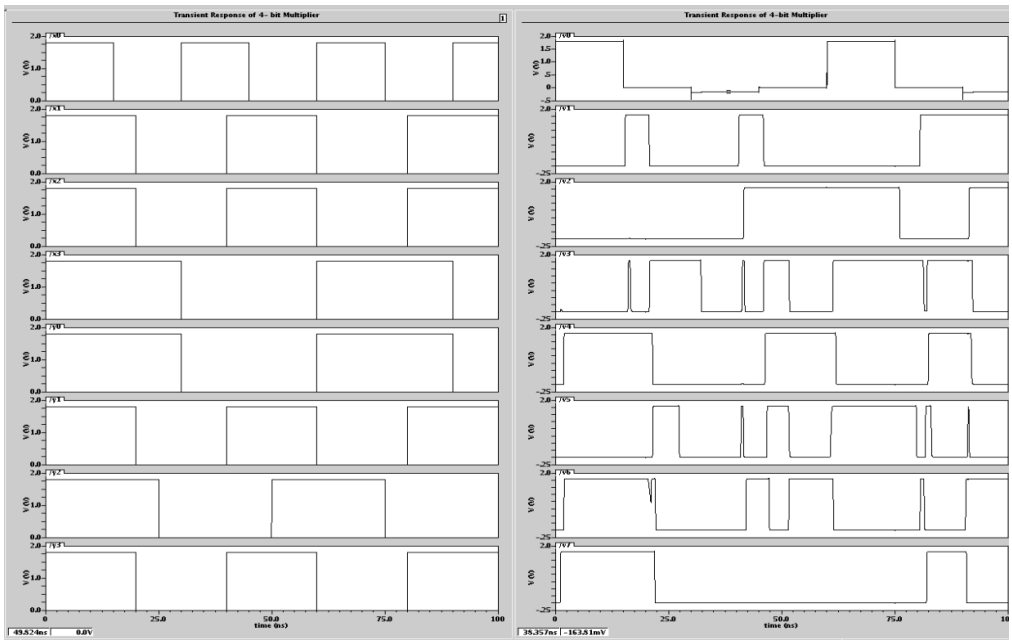


Figure 5.10: Post Layout Simulation of 4-bit Multiplier

5.10.2 POST LAYOUT SIMULATION OF 8-BIT MULTIPLIER

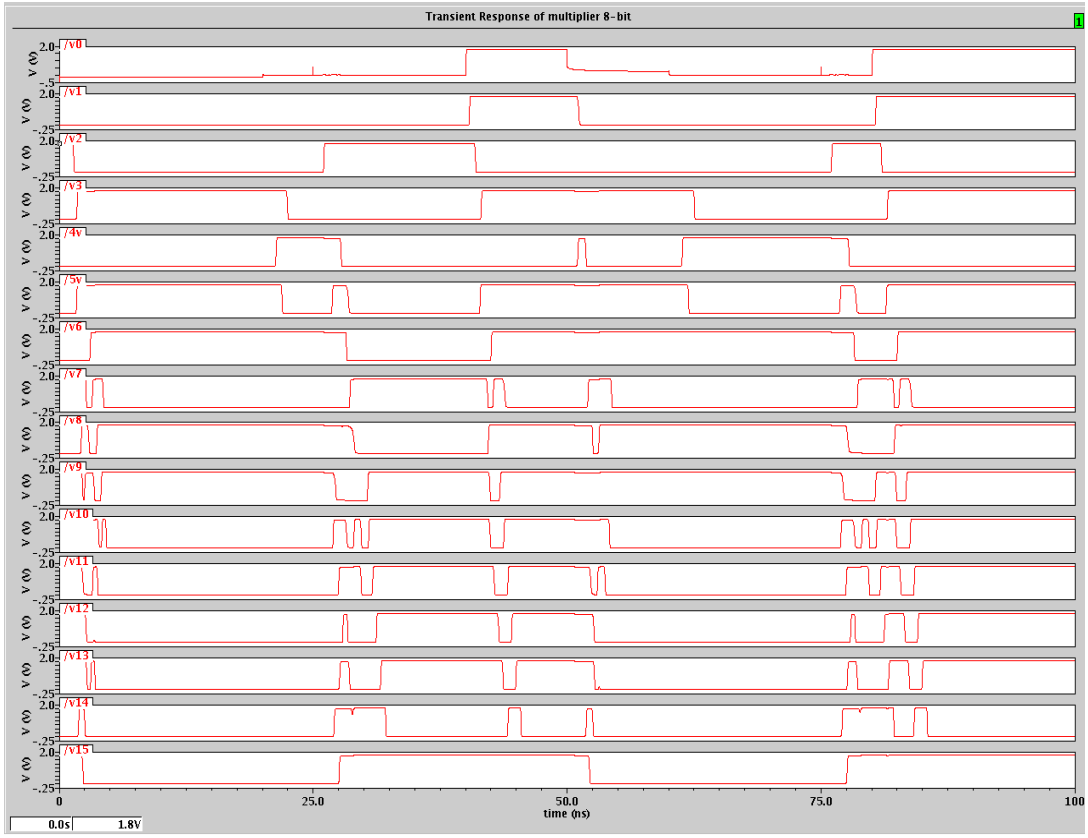


Figure 5.11: Post Layout Simulation of 8-bit Multiplier

Power consumption of a digital circuit is also increased of physical layout design due to parasitic of the connecting wires. Table 5.1 shows the comparison of various blocks of multiplier before and post layout

Table 5.1: Comparison of power pre and post layout

Block	Pre- Layout Power Consumption (W)	Post- Layout Power Consumption (W)
Full adder	3.87×10^{-6}	5.25×10^{-6}
Four to Two Compressor	7.25×10^{-6}	9.21×10^{-6}
4-bit Multiplier	7.14×10^{-5}	9.32×10^{-5}
8- bit Multiplier	2.919×10^{-4}	2.965×10^{-4}

CONCLUSION

The thesis primarily was focused on the design of low power and high performance multiplier. Gate Diffusion Input CMOS logic style used in this work provides us low power design as compared to CMOS and DCVS logic styles. It also presents an area efficient approach to low power, as GDI requires less number of transistors as compared to CMOS and DCVS for any design.

An Adder, 4-bit multiplier, 8-bit Multiplier was designed using GDI, CMOS and DCVS logic styles in Cadence IC5141 using UMC 0.18 µm, layout them in Assura by cadence and the analysis of average dynamic power dissipation with respect to frequency was done. Table below shows their power comparison. The results in Table 1 are at 50 MHz for UMC 0.18 µm and minimum size transistors are used.

Table 1: Power Comparison of Logic Designs

Component	**GDI Logic (W)**	**CMOS Logic(W)**	**DCVS Logic(W)**
Adder	3.18×10^{-6}	3.65×10^{-6}	6.517×10^{-6}
4-bit Multiplier	6.512×10^{-5}	6.875×10^{-5}	8.26×10^{-5}
8-bit Multiplier	1.52×10^{-4}	2.12×10^{-4}	6.34×10^{-4}

The three design logic styles also differ in number of transistors required for designing a particular component. Table 2 shows comparison of transistor count for the various blocks.

Table 2: Transistor Count Comparison of Logic Designs

Component	**GDI Logic**	**CMOS Logic**	**DCVS Logic**
Adder	24	28	32
4-bit Multiplier	382	422	464
8-bit Multiplier	1920	2152	2512

So we can see that percentage saving in terms of power for GDI is 21% as compared to CMOS and 77% as compared to DCVS. Similarly saving in terms of area is 14% as compared to CMOS and 24.12% as compared to DCVS.

As a future scope of the work, Power of the circuit can be further reduced by sizing the transistor keeping logical effort in mind. We can consider the GDI technique in sequential logic designs and mixed circuits. Also, we can compound this technique with other methods used in low-power circuits such as using sleep transistors, dual threshold CMOS, dynamic threshold CMOS. The proposed circuit is implemented in regular p-well CMOS processes, which casts a limitation on a GDI cell library. Still, even in limited-library-based GDI circuits, significant improvements of performance are observed. Implementations of GDI circuits in SOI or twin-well CMOS processes are expected to supply more power-delay efficient design, due to the use of a complete cell library with reduced transistor count. The wasted area problem of large Wallace tree multipliers can be solved using a new method of tree construction.

REFERENCES

[1] S. Shah and A. J. Al-Khalili, "Comparison of 32-bit multipliers for various performance measures", *12th International conference on Microelectronics*, Tehran, 2000.

[2] K. Roy and S. C. Prasad, "Low-Power CMOS VLSI Circuit Design", John Wiley &Sons, 1999

[3] C. S. Wallace. "A Suggestion for a Fast Multiplier", *IEEE Transactions on Electronic Computers*, EC-13:14–17, February 1964.

[4] P. C. H. Meier, "Analysis and Design of Low Power Digital Multipliers", Ph.D. Thesis, Carnegie Mellon University, Dept. of Electrical and Computer Engineering, Pittsburgh, Pennsylvania, 1999.

[5] W. Wolf, "Modern VLSI Design Systems on Silicon", Upper Saddle River: Prentice Hall, 1998.

[6] Biswas, Kelvin, "Multiplexer-Based Array Multipl*iers*" Research work 2005 at university of Windsor

[7] Santaro, "Design and clocking of VLSI Multipliers" *Technical report* October, 1989.

[8] K. Myeir, "Advanced Computer Arithmetic EE486" *IEEE Transactions on Electronic Computers*, EC-13:14–17, Feb, 2003.

[9] K. Roy and Yeo, Kiat-Seng, "Low voltage, Low-power VLSI Subsystems", McGraw-Hill pp.124-141.

[10] B. Parhami, "Computer Arithmetic Algorithms and Hardware Designs", Oxford University Press, 2000.

[11] J. Rabaey and A. Nikolic and A. Chandrakasan, "Digital Integrated Circuits: A Design Perspective" Prentice Hall 2003.

[12] N. Ohkubo, M. Suzuki, T. Shinbo, T. Yamanaka, A. Shimizu, K. Sasaki and Y. Nakagome, "A 0.4ns CMOS 54 x 54-bit Multiplier Using Pass Transistor Multiplexer", *IEEE Journal of Solid-State Circuits*, vol.30, no. 3, pp.252-257, March, 1995.

[13] N. H. E. Weste, "Principle Of Cmos Vlsi Design" Adison – Wesley 1998

[14] G. W. Bewick, "Fast Multiplication: Algorithms And Implementation" Thesis Work For Degree Of Masters In Stanford university, 2002.

[15] J. P. Uyemura, "Fundamentals of MOS Digital Integrated Circuits". Reading, Addison-Wesley, 1996 pp. 136-137.

[16] D. A. Pucknell and K. Eshraghin, "Basic VLSI Design", Prentice Hall, 1994.

[17] A. Saberkari, "A Novel Low-Power Low-Voltage CMOS 1-Bit Full Adder Cell with the GDI Technique," *JME - INTERTECH Conference*- 2006

[18] P. Balasubramanian and J. John, "Low Power Digital Design using Modified GDI" *IEEE Transactions on Electronic Computers*, August 2009.