COMPARISON OF 8-BIT MULTIPLIERS DESIGNED IN DIFFERENT NON-CLOCKED LOGIC STYLES ON THE BASIS OF POWER

Thesis Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology (VLSI Design & CAD)

Submitted by
Vaibhav Dua
Regn. No. 600861017

Under the supervision of
Ms. Alpana Agarwal
Assistant Professor, ECED
Thapar University, Patiala

Department of Electronics & Communication Engineering

THAPAR UNIVERSITY
PATIALA – 147004, INDIA
JULY- 2010

CERTIFICATE

I hereby declare that the work which is being presented in the thesis entitled **"COMPARISON OF 8-BIT MULTIPLIERS DESIGNED IN DIFFERENT NON-CLOCKED LOGIC STYLES ON THE BASIS OF POWER"** in the partial fulfillment for the award of Master of Technology in VLSI Design & CAD from Thapar University is an authentic record of my own work carried under the supervision and guidance of Ms. Alpana Agarwal, Assistant Professor, Department of Electronics & Communication Engineering, Thapar University, Patiala.

Date: 07/07/2010

(Vaibhav Dua)

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

Ms. Alpana Agarwal
Assistant Professor
Thesis Supervisor, ECED
Thapar University, Patiala

Counter Signed by:

13.7.10.

Dr. A. K. CHATTERJEE
Professor & Head (ECED)
Thapar University
Patiala-147004

14.7.10

Dr. R. K. SHARMA
Dean of academic affairs
Thapar University
Patiala-147004

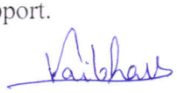
ACKNOWLEDGEMENTS

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

I am also thankful to our **Head of the Department, Dr. A. K. Chatterjee,** entire faculty and staff of Electronics and Communication Engineering Department and the friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Next, I would like to thank Mr. B. K. Hemant for all the good times in the lab. I specially thank to my classmates Amit Jindal, Anand Kamra, Pankaj Kumar and Deependra Sharma for their help, criticisms, suggestions, and friendship which makes everyday a pleasant one. Thank you so much to all of you for the fun and great memories here at Thapar University.

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 and without your constant support, encouragement and sacrifices I would never have made it to this stage in life. I love you so much. I would also like to express my heartily gratitude to my sister Mrs. Nitika Saigal for moral encouragement and support.



Vaibhav Dua

ABSTRACT

This thesis 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 proposed.

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 different non-clocked logic styles have been compared on the basis of power. The advantage of non-clocked logic styles is that there is less switching power in these styles because there no clock is used. Among DCVS, DSL and CNTL logic styles, CNTL consumes less power because there is less dynamic power dissipation in CNTL.

Multiplier is the most commonly used circuit 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. 8-bit CNTL multiplier consumes least power among the three non-clocked logic styles which has been discussed in the thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

CMOS	Complementary Metal Oxide Semiconductor
DCVS	Differential Cascode Voltage Switch Logic
DSL	Differential Split-Level Logic
CNTL	Cascode Non-Threshold Logic
DRC	Design Rule Check
PMOS	p-type Metal Oxide Semiconductor
NMOS	n-type Metal Oxide Semiconductor
IC	Integrated Circuit
VLSI	Very Large Scale Integration
WT	Wallace Tree
CSA	Partial Product Addition
DSP	Digital Signal Processing
DRC	Design Rule Check

CHAPTER 1

INTRODUCTION

1.1 Motivation

With much of the research efforts directed towards increasing the speed of digital systems, present day technologies possess computing capabilities that make possible powerful personal workstations, sophisticated computer graphics, and multimedia capabilities such as real-time speech recognition and real-time video. High-speed computation has thus become the expected norm for the average user, instead of being the province of the few with access to a powerful mainframe. Likewise, another significant change in the attitude of users is the desire to have access to this computation at any location, without the need to be physically tethered to a wired network. The requirement of portability thus places a severe restriction on size, weight and power. Power is particularly important since conventional nickel-cadmium battery technology only provides small amount of energy for each pound of weight. Improvements in battery technology are being made, but it is unlikely that a dramatic solution to the power dissipation is forth coming.

Although the traditional mainstay of portable digital applications has been in low power, low throughput uses such as wristwatches and pocket calculators, there are an ever increasing number of portable applications requiring low power and high throughput. For example, notebook and laptop computers, representing the fastest growing segment of the computer industry, are demanding the same computations capabilities as found in desktop machines. Equally demanding are developments in personal communications services (PCS), such as the current generation of digital cellular telephony networks which employ complex speech compression algorithms and sophisticated radio modems in a pocket sized device. Even more dramatic are the proposed future PCS applications, with universal portable multimedia access

supporting full motion digital video and control via speech recognition. In these applications, not only will voice be transmitted via wireless links, but data as well. This will facilitate new services such as multimedia access supporting database access and supercomputing for simulation and design, through an intelligent network which allows communication with these services or other people at any other place and time. Power for video compression and decompression and speech recognition must be added to the portable unit to support these services. Indeed, it is apparent that portability can no longer be associated with low throughput; instead, vastly increasing capabilities, accuracy in excess of that demanded of fixed workstations, must be placed in low power environment.

Even when low power is available in non portable applications, the issue of low power design is becoming critical. Until now, this power consumption has not been of great concern, since large packages, cooling fins, and fans have been capable of dissipating the generated heat. However, as the density and size of the chips and systems continue to increase, the difficulty in providing adequate cooling might either add significant cost to the system or provide a limit on the amount of functionality that can be provided.

Thus, it is evident that methodologies for the design of high throughput, low-power digital systems are needed. Fortunately, there are clear technological trends that give us a new degree of freedom, so that it may be possible to satisfy these seemingly contradictory requirements. Scaling of device feature sizes, along with the development high density, low-parasitic packaging, such as multichip modules, will alleviate the overriding concern with number of transistors being used. The important consideration, particularly in portable applications, is that many computations tasks are likely to be real-time; the radio modem, speech and video compression, and speech recognition require computation that is always at near-peak rates. Conventional schemes for conserving power in laptops, which are generally based on power-down schemes, are not appropriate for these continually active computations.

1.2 Organization of Thesis Work

This thesis is organized as follows:

CHAPTER 1: INTRODUCTION. This chapter gives an idea of need for less power in digital circuits.

CHAPTER 2: LITERATURE SURVEY. The topic covers the comparative study of the three non-clocked logic styles i.e. DCVS, DSL and CNTL. It has been analyzed in this chapter that CNTL consumes less dynamic power

CHAPTER 3: MULTIPLICATION BASICS. This chapter explains what multiplication is? How multiplication is performed in digital domain? It also talks about various multiplier architectures in detail.

CHAPTER 4: DESIGN OF MULTIPLIER. 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 there validation by comparing with truth tables.

CHAPTER 5: PHSICAL LAYOUT DESIGN. This chapter discusses the designs of layouts for the various blocks such as AND gates, OR gates, Adders and CNTL multiplier, which are designed in Cadence Virtuoso UMC 0.18 micron Technology and the Layout vs. Schematic (LVS) program was executed to perform a comparison of the schematic to the physical layout.

CHAPTER 6: SIMULATIONS AND RESULTS. This chapter discusses the schematic based and post layout results of the full adders and multipliers.

CHAPTER 7: CONCLUSION AND FUTURE SCOPE. This chapter compares multiplier design of three logic styles and future scope.

CHAPTER 2

LITERATURE SURVEY

2.1 Sources of Power Dissipation

There are three major sources of power dissipation in digital CMOS circuits, which are summarized in following equation:

$$P_{total} = a.C_l.V_{dd}^{\ 2}.f_{clk} + I_{sc}.V_{dd} + I_{leakage}.V_{dd}$$

The first term represents the switching component of power, where • is the switching factor, C_l is the loading capacitance, f_{clk} is the clock frequency. In most cases, the voltage swing is the same as the supply voltage V_{dd} ; however in some logic circuits, such as in single-gate pass transistor implementations, the voltage swing on some internal nodes may be slightly less. The second term is due to direct-path short circuit current I_{sc}, which arises when both NMOS and PMOS transistors are simultaneously active, conducting current directly from supply to ground. Finally, leakage current $I_{leakage}$, which can arise from substrate injection and sub-threshold effects, is primarily determined by fabrication technology considerations. The dominant term in a "well-designed" circuit is the switching component, and low power design thus becomes the task of minimizing C_l, V_{dd}, and f_{clk}, while retaining the required functionality [1].

The power-delay product can be interpreted as the amount of energy expended in each switching event (or transition) and is thus particularly useful in comparing the power dissipation of various circuit styles.

If it is assumed that only the switching component of the power dissipation is important, then it is given by:

$$\text{Energy per transition} = P_{total} / f_{clk} = C_l . V_{dd}^{\ 2}$$

where C_l is the load capacitance being switched to perform the computation.

Therefore for low power what we have to look out for is the switching power dissipation and techniques to reduce switching power dissipation. Some general logic style requirements for low-power circuit implementation are as follows:

1) Switched capacitance reduction:

Capacitive load, originating from transistor capacitances (gate and diffusion) and interconnect wiring, is to be minimized. This is achieved by having as few transistors and circuit nodes as possible, and by reducing transistor sizes to a minimum. In particular, the number of (high capacitive) inter-cell connections and their length (influenced by the circuit size) should be kept minimal. Transistor downsizing is an effective way to reduce switched capacitance of logic gates on noncritical signal paths. For that purpose, a logic style should be robust against transistor downsizing, i.e., correct functioning of logic gates with minimal or near-minimal transistor sizes must be guaranteed [1].

2) Supply voltage reduction:

The supply voltage and the choice of logic style are indirectly related through delay-driven voltage scaling. That is, a logic style providing fast logic gates to speed up critical signal paths allows a reduction of the supply voltage in order to achieve a given throughput. For that purpose, a logic style must be robust against supply voltage reduction, i.e., performance and correct functioning of gates must be guaranteed at low voltages as well. This becomes a severe problem at very low voltages of around 1 V and lower, where noise margins become critical [1].

3) Switching activity reduction:

Switching activity of a circuit is predominantly controlled at the architectural and registers transfer level (RTL). At the circuit level, large differences are primarily observed between static and dynamic logic styles. On the other hand, only minor

transition activity variations are observed among different static logic styles and among logic gates of different complexity, also if glitching is concerned [1].

4) Short-circuit current reduction:

Short-circuit currents (also called dynamic leakage currents or overlap currents) may vary by a considerable amount between different logic styles. They also strongly depend on input signal slopes (i.e., steep and balanced signal slopes are better) and thus on transistor sizing. Their contribution to the overall power consumption is rather limited but still not negligible (10–30%), except for very low voltages $V_{dd} <= V_{tn} + |V_{tp|}$, where the short-circuit currents disappear. A low-power logic style should have minimal short-circuit currents and, of course, no static currents besides the inherent CMOS leakage currents [1].

So if we are looking out for low power design, it is preferable to use Non-Clocked logic styles because they have less switching power dissipation. Now we will discuss about various Non-Clocked logic styles and there advantages and disadvantages over one another.

2.2 Non-Clocked Logic Styles for Low Power Design

Non-clocked logic is ubiquitous in electronic design, due to a number of considerations including:

- Low power consumption
- Straightforward delay rule timing
- Inherent reliability and noise immunity
- Process variation and defect tolerance
- Deterministic diagnostic capability
- Migration into successive technology generations [2].

Some of the non-clocked logic styles are discussed ahead:

2.2.1 Differential Cascode Voltage Switch Logic (DCVS):

Logic design is achieved in DCVS by cascoding differential pairs of MOS devices into powerful combinational logic tree networks capable of processing complex Boolean logic functions within a single circuit delay. Logic trees of N-high cascoding of differential pairs of NMOS devices are capable of processing Boolean functions with up to (2^N-1) input variables. DCVS has been found to offer a performance advantage of up to 4X compared to the CMOS/NMOS primitive NAND/NOR logic families, while maintaining the expected low power characteristics of CMOS circuitry. Potentially DCVS is twice as dense as primitive NAND/NOR logic, and is compatible with the existing design automation tools. Combinational logic trees can be designed in cascoded high-performance NMOS devices with unstacked PMOS devices used sparingly as pull-up devices in load and buffer circuitry. Optimization of the PMOS devices and the criticality of PMOS to NMOS spacing can therefore be relaxed, relieving the device/process complexity burden for DCVS designs [3].

The DCVS circuit family shares the sensitivity of static circuitry to device transconductance. Specifically, since the evaluate path commonly comprises NMOS devices, changes in NMOS device channel length, threshold or gate oxide will be immediately reflected in circuit performance.

The performance of the stacked devices found in the static DCVS evaluate trees suffer from body effect. The logic evaluation trees must provide low resistance paths to ground in order to switch the load circuit; positive NMOS source-substrate bias voltage increases threshold voltage and introduces additional delay.

The DCVS circuit concept, in its in differential form is illustrated in figure 2.1. Depending on the differential inputs, either node N1 or node N2 is pulled down by the NMOS combinational logic tree network. Regenerative action sets the PMOS latch to static outputs Q, QBAR of full differential V_H and ground logic levels. The logic trees are free of direct current after the latch sets. Since the inputs drive only the NMOS tree devices, input gate capacitance loading is typically a factor of 3X smaller than CMOS circuits that require the

complementary n-channel and p-channel devices to be driven. Performance of the circuit in figure 2.1 is limited by the set time of PMOS latches [3].

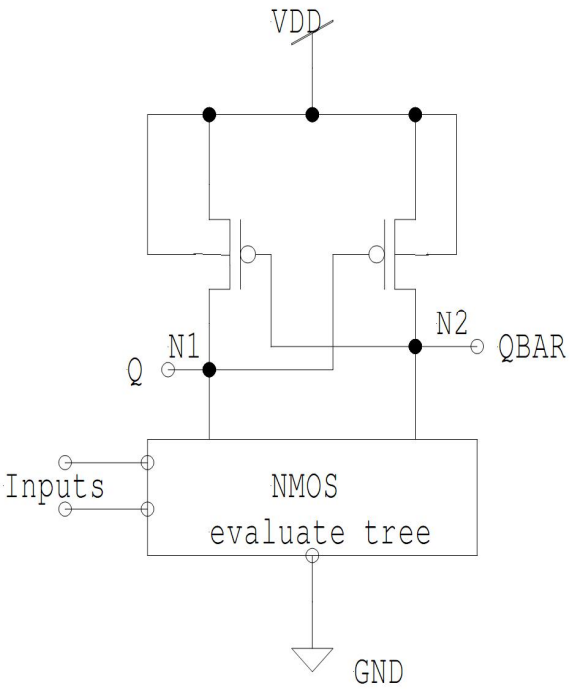


Figure 2.1: Basic Circuit of DCVS [3]

2.2.1.1 Function:

True and complement versions of logical inputs A and B are supplied to the DCVS circuit in figure 2.2. If A and B both transition high, then NMOS devices 3 and 4 are turned on, and NMOS devices 5 and 6 are in their off state. Now the output Q goes low as it gets path to ground via NMOS 3 and 4 but QBAR remains high as the NMOS 5 and 6 are off and it does not get path to ground. As the two pull-down networks implement complementary functions (Q and QBAR), and hence, the logic style has larger area or switched capacitance. However, it should be noted that it is sometimes possible to share some logic in the two pull-down networks to reduce the area. The complementary outputs can also help eliminate inverters, which might otherwise be required [2].

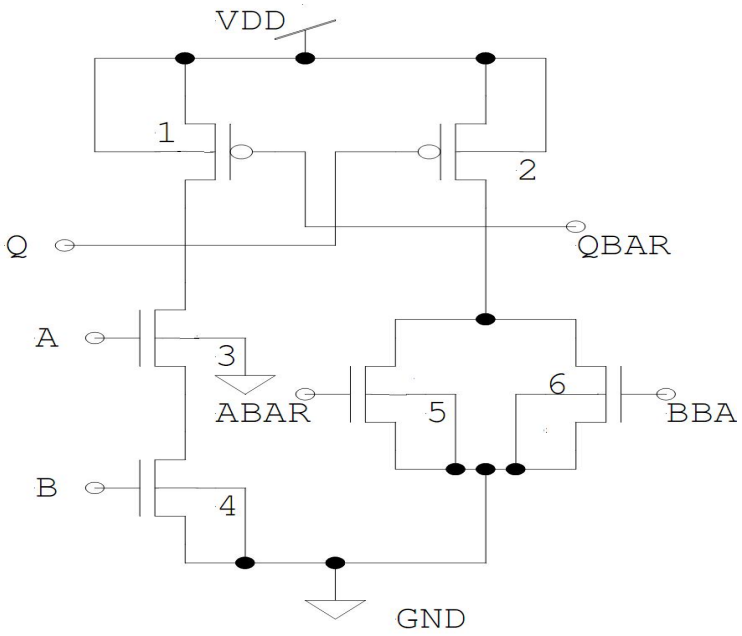


Figure 2.2: AND Logic Using DCVS [2]

2.2.1.2 Characteristics:

The strengths of DCVS are:

- Superior logic density
- Implicit invert available
- High reliability and noise immunity

The limitations of DCVS are:

- Dual rail wiring
- Struggle between PMOS output strength and latch hysteresis
- Higher device count in some applications

The advantage of Differential Cascode Voltage Switching is realized in the logic density achieved by evaluating complex trees of logic in one delay stage. Differential pair logic trees may easily be 4 device tall and process (2^4-1) inputs. Further efficiency is achieved in the elimination of large PMOS from each logic function executed in the tree. Boolean functions are implemented in NMOS only. The PMOS serve solely as pull up devices. Differential Cascode Voltage Switch logic offers implicit noise immunity at each stage, due to its cross-

coupled nature. Performance is somewhat compromised by the hysteresis associated with toggling the load devices. The PMOS load devices must be small enough that its on-current is easily overcome by the switching logical pull-downs, but large enough to drive high outputs with acceptable delays. Normal process tolerance can make it hard to remain within a window of acceptable device strengths. Until the output node flips, the evaluate tree sinks load current, highlighting the power trade-off that cross-coupled load devices require [2].

2.2.2 Differential Split-Level Logic (DSL):

The DSL principle is shown in figure 2.3. Two extra NMOS transistors N10 and N20 are placed between the PMOS part and the logic NMOS part. Their gates are controlled by the reference voltage, which must be equal to half V_{DD} plus the threshold voltage of the NMOS transistors to guarantee optimum circuit operation. The gates the PMOS transistors P1 and P2 are now connected to F and FN instead of outputs S and SN. Now we again switch input D low and DN high. Then node SN has a high level of V_{DD} and node S a low level. The reference voltage determines the high-level at node FN to be half V_{DD}. Node F has a low level of about 100mV, because PMOS P2 is weakly on. This causes static power dissipation. The NMOS transistor N1 is cut-off, which causes high impedance to V_{DD} for node FN while there is low impedance to V_{DD} for node SN. If we now switch the inputs D and DN then NMOS N1 turns on and N2 turns off [4].

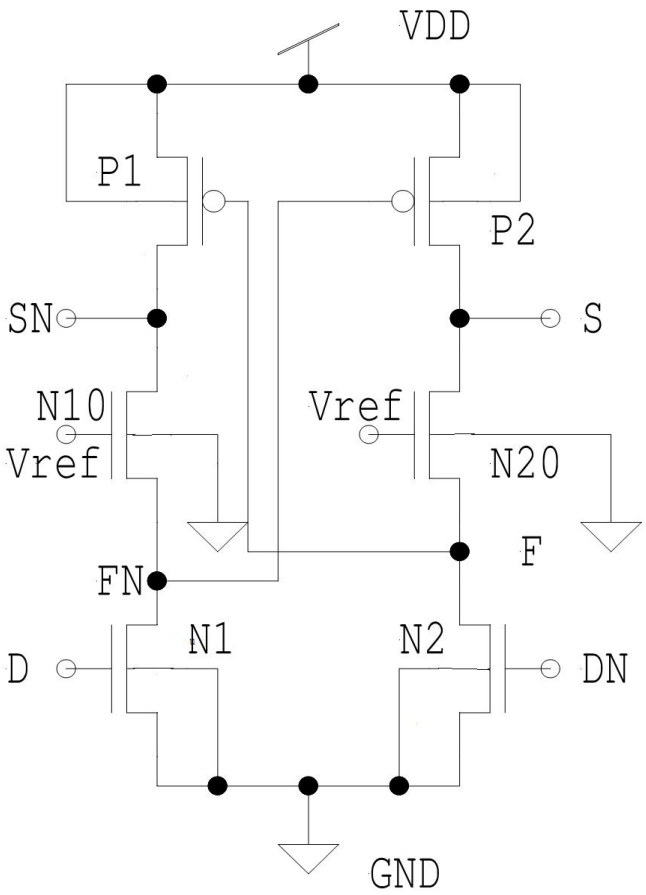


Figure 2.3: Basic Circuit of DSL [4]

The half V_{DD} level at node FN will immediately be discharged and PMOS P2 turns more on to its high drive state. At the same time node S and F start rising because PMOS P2 was already partly on, causing PMOS P1 to switch faster to its low drive state.

Differential split-level CMOS logic is a circuit technique which allows, in any given technology, shorter channel length owing to a reduced drain-source voltage.

2.2.2.1 Function:

With input B high and BBAR low, assume input A transitions from a low to a high (at voltage V_{DD}), and input ABAR transitions from high value to low value, as shown in figure 2.4. With bias V_{ref} set at (V_{DD}/2+ V_{TN}), PMOS load device 1, connected via node N2 as shown, receives a bias of V_{DD}/2; PMOS load device has its gate grounded through devices 5 and 6. Signal QBAR develops a low output (ground) through devices 3 and 6, and signal Q is developed as a high output (V_{DD}), as devices 7 and 8 are off [2].

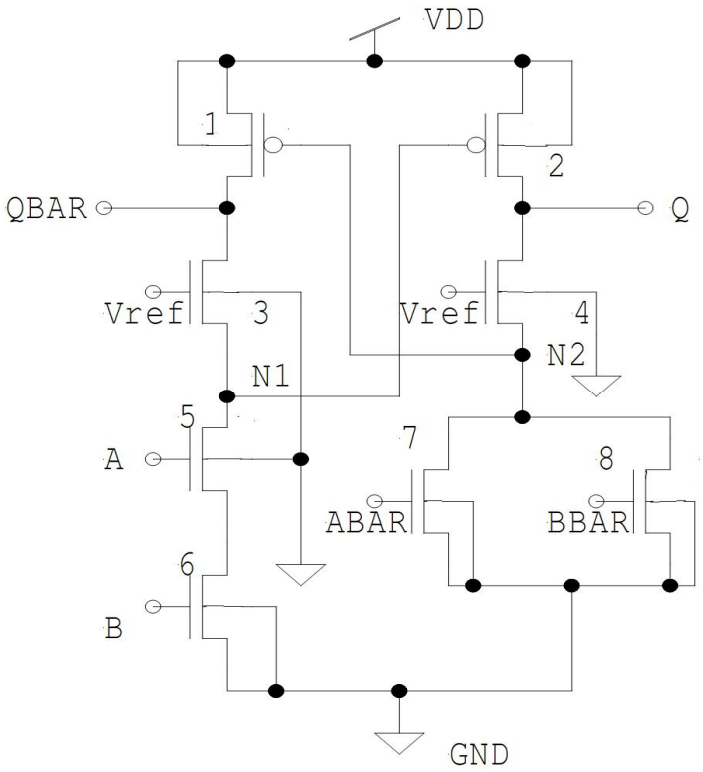


Figure 2.4: AND Logic Using DSL [2]

2.2.2.2 Characteristics:

The strengths of DSL are:

- Lower power
- Superior static performance
- High reliability

The limitations of DSL are:

- Reference voltage generation required
- High device count

DSL leverages the presence of a precision bias. Sampling output off immediately below the load PMOS device provides a strong high output at full V_{DD}; a low output, however, is held at ground through the resistance of the logic tree. The original literature recommends taking

output from these uppermost nodes. Transitions are hastened because the node N1 needs only to be discharged from $V_{DD}/2$ rather than V_{DD}.

The tree device gated by V_{REF} makes DSL substantially different from DCVS origin. Since the NMOS pull down device never sees drain voltage higher than $V_{DD}/2$, the channel hot electron-based performance degradation is minimized. It can argue that this enables usage of technology with channels shorter than what would be expected for the given V_{DD}, realizing higher performance. The added resistance of the added reference device necessarily reduces the total possible tree height.

The presence of that device may also never allows the PMOS load device on the "0" side of the output to be completely turned off, as its gate rise to $(V_{DD} - V_T)$. The presence of the resulting current causes the "low" output to hover 100-200 mV above ground, causing DC power consumption. This increase and the higher device count caused by the reduced logic content must be weighed against the AC power savings realized by the limited swing [2].

2.2.3 Cascode Non-Threshold Logic (CNTL):

CMOS circuits have played a dominant role in digital VLSI circuits, mainly due to low dc power consumption. However conventional static CMOS circuits suffer from speed and low packaging density. Several new static and dynamic CMOS circuits have been proposed to deal with these problems. Cascoding the differential NMOS logic trees while using only two cross-coupled PMOS devices as loads is the main approach to save chip area in those new static circuits. Of the various new static logic circuits, the differential split-level logic, DSL derived from cascoded logic, has been used in low power and high speed CMOS complex-gate applications. Two new CMOS logic circuits, one called the CMOS non-threshold logic (CMOS NTL) and the other the CMOS cascode non-threshold logic (CMOS CNTL), are there which have high speed and low power consumption than DSL. They are compatible in process with conventional CMOS circuits, and are expected to extend the application field of CMOS circuits to higher speed and low power regimes.

As in “Differential Cascode Voltage-Switched Logic” considerable power is consumed in cross-over current while switching the contents of the circuit. Additional power is consumed in achieving the full swing of the internal node. DSL reduced active power consumption by reducing voltage swing. Cascode Non-Threshold Logic (CNTL) further improves the concept.

Cascode Non-threshold Logic makes use of a concept first implemented in bipolar transistors; by resistively dividing the power supply voltage, the required output voltage swing can be reduced, improving performance. CNTL is developed from CMOS Non-threshold Logic (NTL), a single ended CVS structure not present due to its excessive DC power consumption. CNTL improves upon DSL by not requiring a precision voltage reference. The devices which add additional voltage drop have their gates tied to V_{DD} or to a feedback node, instead of a reference. The negative feedback substantially reduces the power consumption of the NTL origin, but at the expense of some performance. The performance penalty is addressed with the shunt capacitor. Like other CVS design styles, the circuit comprises a differential logic tree of NMOS, and a PMOS load circuit [5].

The CNTL principle is shown in figure 2.5. The circuit structure is similar to the differential-split level (DSL), but the electrical behavior is essentially different. Like all cascode logic circuits, the CNTL has a differential cascode voltage switch (DCVS) tree, constructed by NMOS devices. Similar to the DSL circuit, two cross-coupled PMOS devices are used as loads, and the two NMOS devices, M_{N1} and M_{N2}, are inserted between the PMOS loads and the NMOS logic tree to split the supply voltage to reduce the output voltage swing. Q and QBAR are the two true outputs nodes, and F and FBAR are two pseudo output nodes which have similar voltage swings and be used for wire routing among different gates as in the DSL circuits.

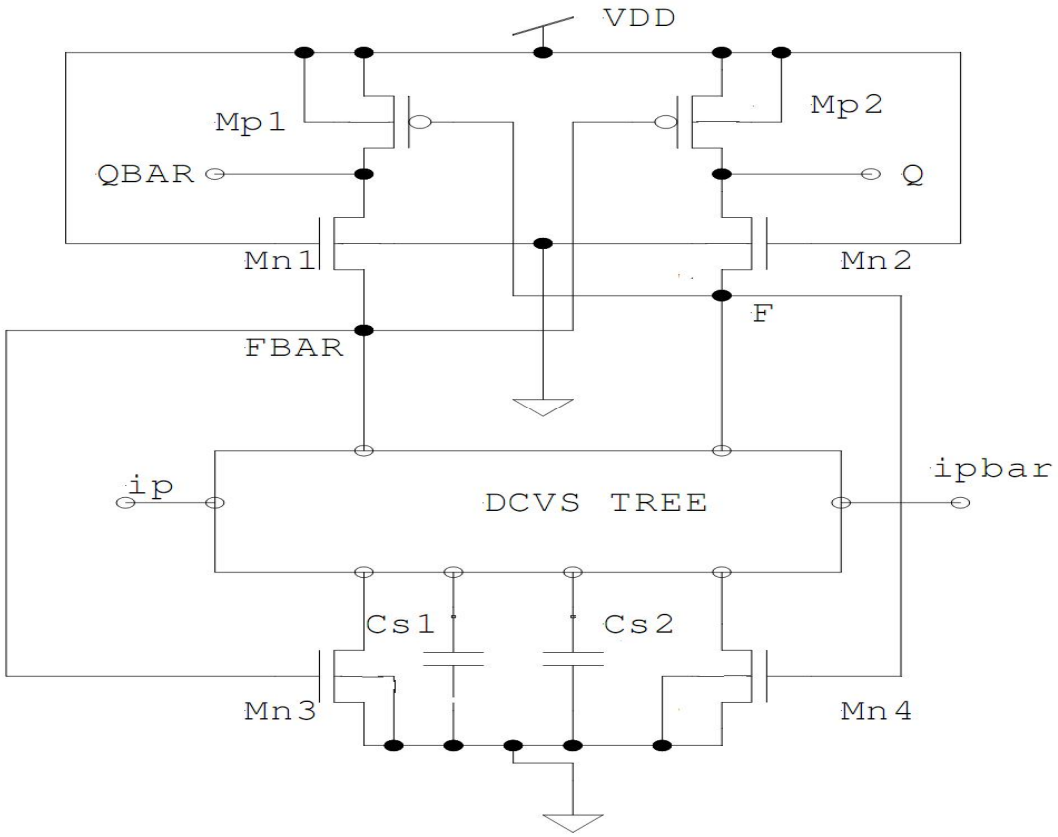


Figure 2.5: Basic Circuit of CNTL [5]

The main difference between the CNTL and DSL circuit is that the two NMOS devices, M_{N3} and M_{N4}, and the two shunting capacitors, C_{S1} and C_{S2}, are added below the DCVS tree to form non-threshold circuit. The shunting capacitors can be implemented using MOS transistors as in the NTL case. The gate terminals of M_{N3} and M_{N4} are connected to the nodes F and FBAR respectively, to make the layout more compact. It is found that this connection does not affect the negative feedback effects of M_{N3} and M_{N4}. As expected, the NMOS M_{N3} and M_{N4} can further reduce the output voltage swing, and their negative feedback effects can be reduced by C_{S1} and C_{S2}.

However, the gates of the two NMOS devices M_{N1} and M_{N2} are connected together to V_{DD}, not to reference voltage as in DSL circuits. So the voltage reference circuit and the routing area needed for reference voltage is eliminated [5].

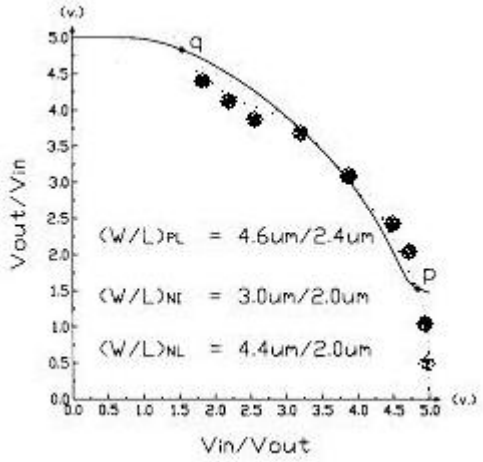


Figure 2.6(a) [5]

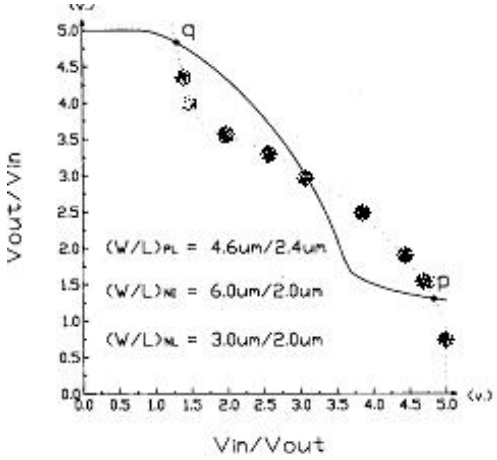


Figure 2.6(b) [5]

Typical dc voltage characteristics of CMOS CNTL inverters are shown in figure 2.6(a) and (b) for two different dimension designs are taken reference [5]. For the solid lines in these figures, the abscissa stands for the input voltage, and the ordinate for the output voltage, and vice versa for the dotted lines. As may be seen from these figures, the transfer curves are quite smooth. This implies no clear cut at logic threshold. However, the two cross point p and q guarantee a correct logic operation. This is why the name non-threshold is adopted

2.2.3.1 Function:

True and complement logic inputs are provided to the DCVS tree of NMOS devices in figure 2.7. The load circuit comprised of devices 1, 2, 3 and 4 latches in the direction determined by the logic tree. NMOS devices 2 and 4 limit the upper end of the logic voltage swing. NMOS devices 9 and 10 generate negative feedback to their respective trees by increasing there on resistance as the tree discharges, limiting the lower end of the logic voltage swing. Capacitors C1 and C2 are present to shunt the effect of the devices 9 and 10 and moderate the negative feedback. The transfer function associated with this style has no distinct “break” at a logic threshold, but produces logically valid outputs levels [2].

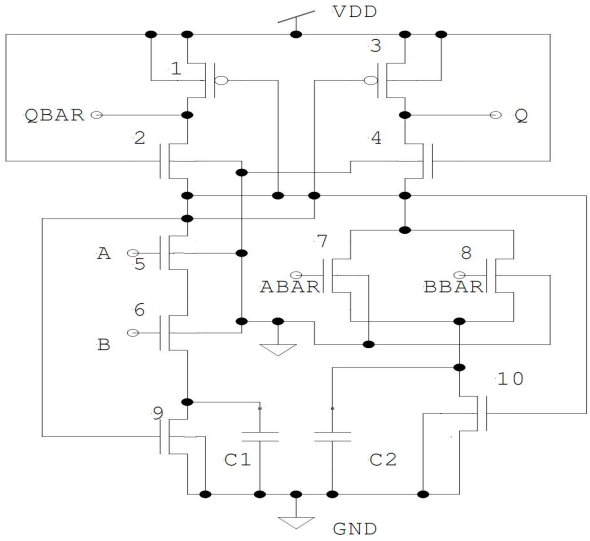


Figure 2.7: AND Logic Using CNTL [2]

2.2.3.2 Characteristics:

The strengths of CNTL are:

- Low power
- Heavy load drive capability
- Differential noise immunity

The limitations of CNTL are:

- Separate rail-level interface required
- Compromised performance
- Substantial area increase [2].

CHAPTER 3

Multiplication Basics

3.1 Introduction

Digital multipliers are major source of power dissipation in Digital Signal Processors. High power dissipation in these structures is mainly due to the switching of a large number of gates during multiplication. In addition, much power is also dissipated due to a large number of spurious transitions on internal nodes. Timing analysis of a full adder, which is a basic building block in multipliers, has resulted in a different array connection pattern that reduces power dissipation due to the spurious transition activity. Furthermore, this connection pattern also improves the multiplier throughput.

A variety of measures can be used to evaluate the efficiency of the processors. So both the area occupied by the circuit and the time required for the performance of computation must be taken into consideration. The speed of the multiplier is determined by both architecture and circuit. The speed can be expressed by the number of the cell delays along the critical path on the architecture level of the multiplier. The cell delay, which is normally the delay of the adder, is determined by the design of the circuit of the cell. Therefore depending on the speed and area requirements, the digital multipliers used can be either of bit-serial or a bit-parallel based architecture. The bit-serial approach processes the data serially where at every clock cycle a single data bit is fed to the processor to be processed. In contrast, the parallel approach processes the data bits in a parallel fashion in just one clock cycle.

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 [6].

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 3.1 below represents the multiplication flow in detail.

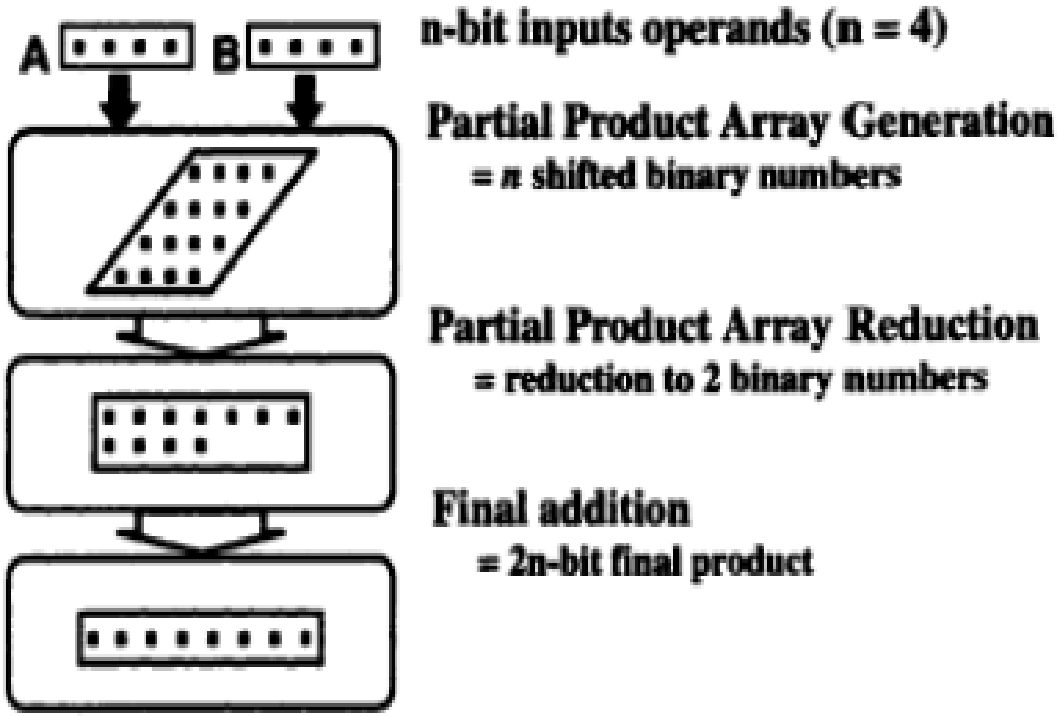


Figure 3.1: Multiplication Flow [6]

3.2 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 ($0 \bullet i \bullet n \bullet 1$) 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

($x_i y_j$) ($0 \bullet i \bullet n \bullet 1$ & $0 \bullet y \bullet n \bullet 1$). The resulting values are called partial products. Fig.3.2 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 [6].

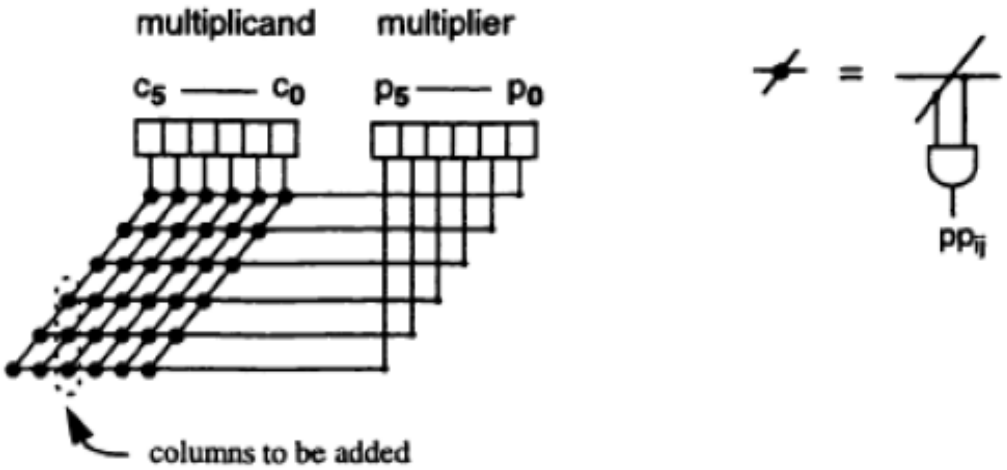


Figure 3.2: Partial Product Generation [6]

3.3 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.

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 influences that addition in separate columns can be performed independently. 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 [6].

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

3.3.1 Array Multiplier

Array multiplier 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.

There are several possible array topologies including simple, double and higher order arrays.

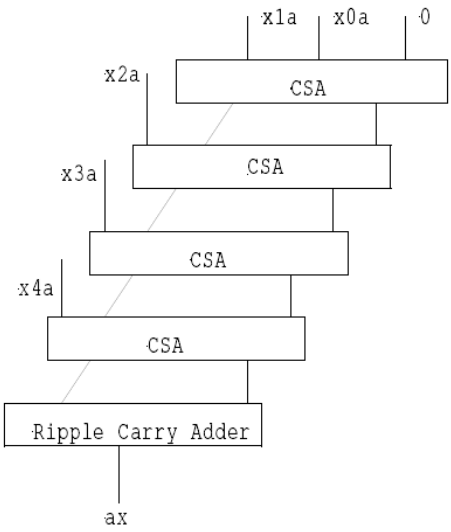


Figure 3.3: Array Multiplier Mechanism [7]

3.3.1.1 Simple Array Multiplier

The array multiplier originates from the multiplication parallelogram. As shown in Figure 3.4, 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.

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 [7].

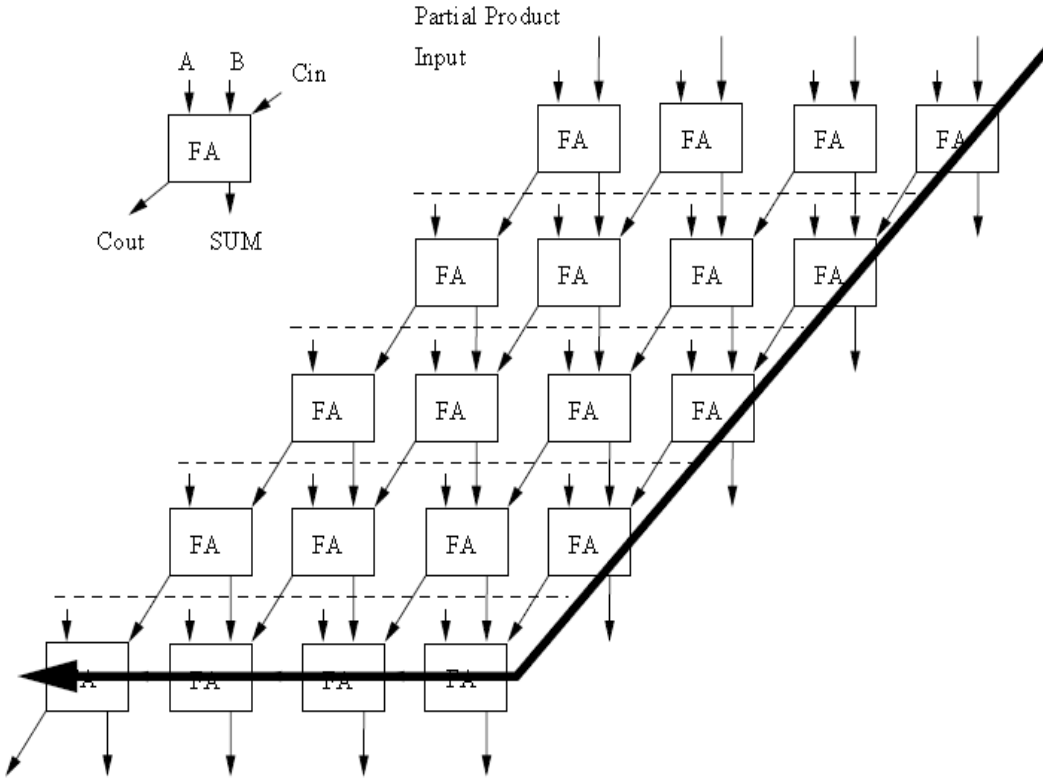


Figure3.4: 4x4 Array Multiplier [7]

3.3.1.2 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 of the 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 [7].

3.3.1.3 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 [7].

3.3.2 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.

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 3.1 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 [8].

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 3.1: 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 3.5 shows a 4-bit by 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.

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 [8].

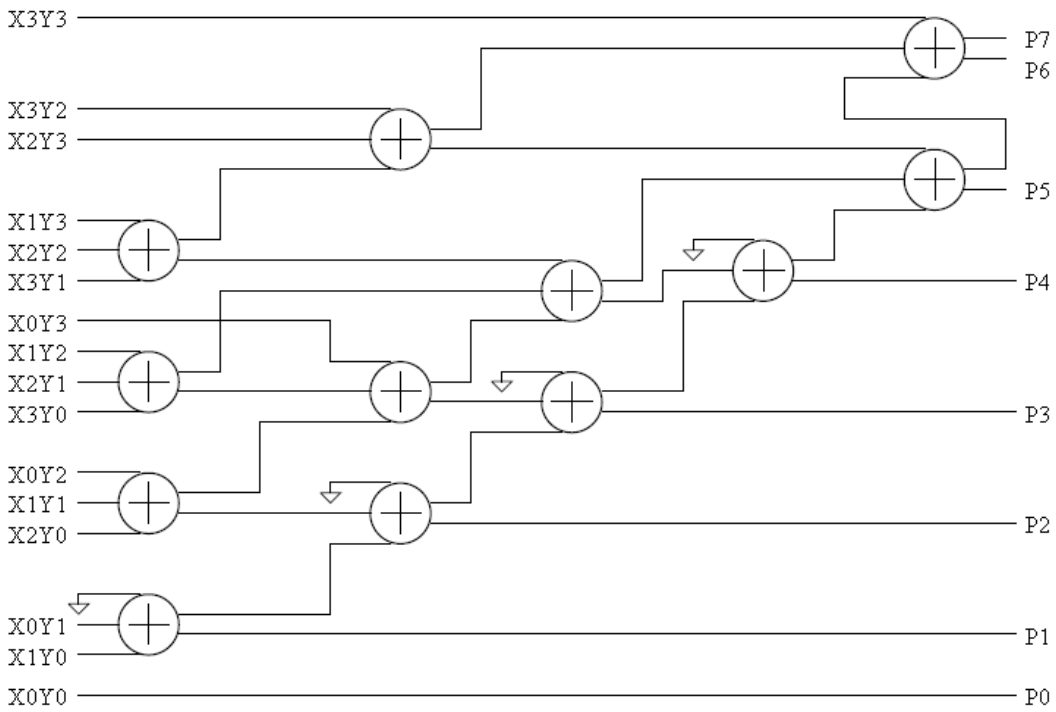


Figure 3.5: 4x4 Tree Multiplier [8]

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 [3:2] as well as [4:2] compressors.

3.3.2.1 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 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 [8].

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.

- 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. Wallace tree flow can be seen in Figure 3.6.

In WT 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 a function of (log N) [8].

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.

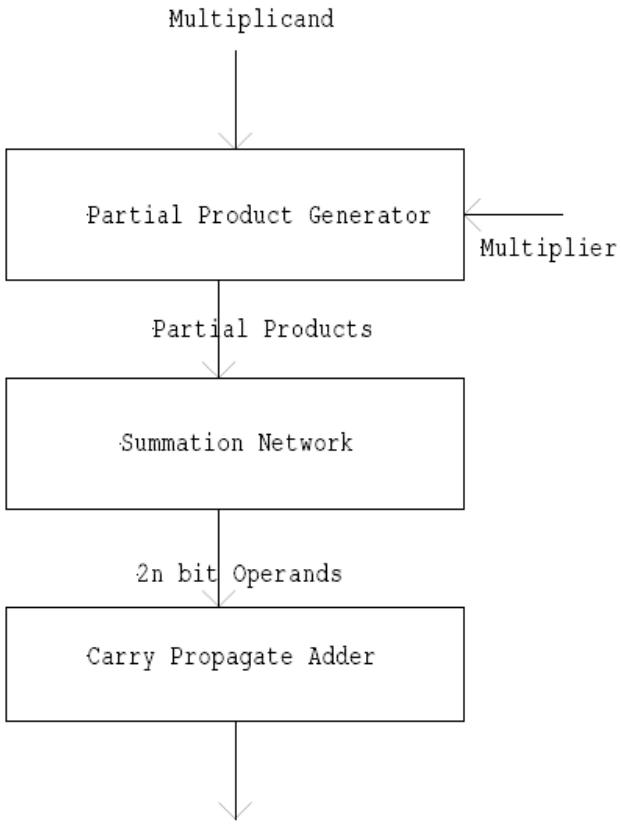


Figure 3.6: Flow of Wallace Tree Multiplier [8]

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 3.7 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 indicates [4:2] compressors.

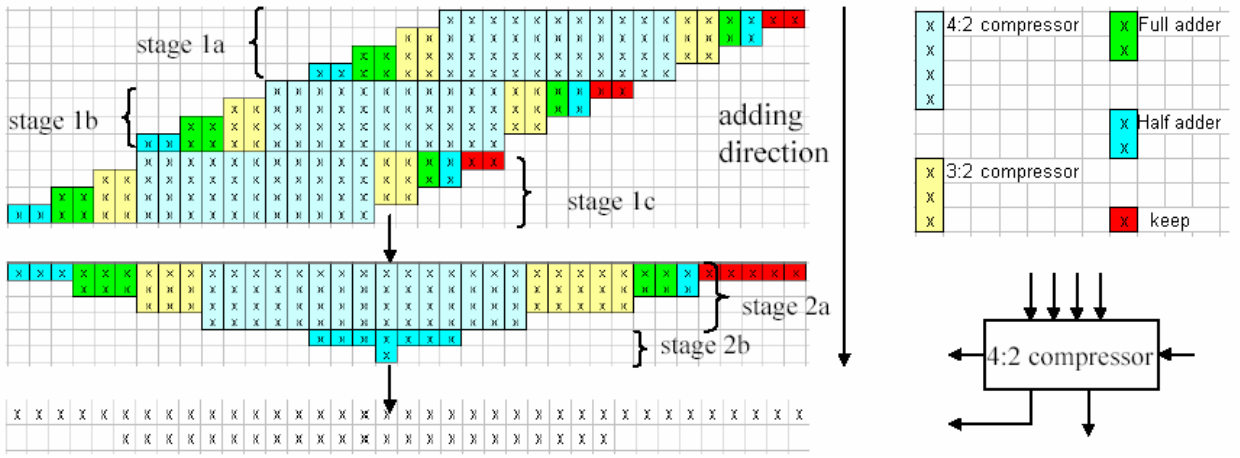


Figure 3.7: Wallace Tree Example [8]

3.4 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 3.8 shows a simple compressor.

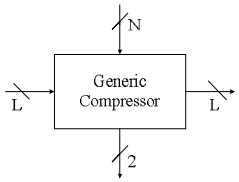


Figure 3.8: A Generic Compressor

3.4.1 [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 Figure3.9 [9].

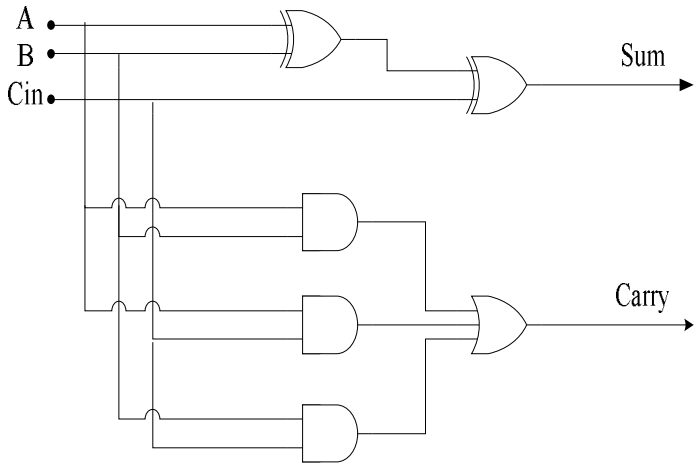


Figure 3.9: Gate level design of [3:2] compressor

3.4.2 [4:2] Compressor

4 to 2 compressor is shown in figure below. It takes four inputs of equal weight and produces two outputs. [4:2] compressors can be used in binary tree to produce a much more regular layout. It can be constructed from two [3:2] compressors and OR Gate. 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, first full adder takes in the three inputs to be added and the next full adder takes in the fourth bit to be added, sum of the first adder and third input is grounded. The carryout of first adder and the second adder is given as input to OR Gate. Output of the OR Gate is the carryout of the [4:2] compressor and the output of the second full adder is the sum of the [4:2] compressor. Block diagram of [4:2] compressor is shown in figure 3.10 [9].

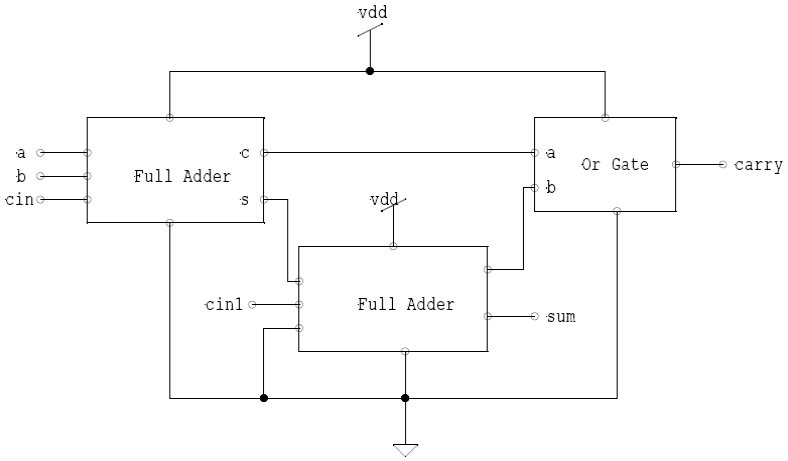


Figure 3.10: [4:2] compressor using [3:2] compressor [9]

Table 3.2: Truth Table of [4:2] Compressor

INPUTS				OUTPUTS	
a	**B**	**cin**	**cin1**	**sum**	**carry**
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	1	0	0	1	0
1	0	0	0	1	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	0	0	1
0	1	1	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	0	1

CHAPTER 4

Design of Multiplier

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.

4.2 Partial Product Generation

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 different non-clocked logic styles (DCVS, DSL, CNTL). For n-bit multiplication we require n^2 AND gates, hence for our design of 8-bit multiplier we require 64 AND gates. The schematic of AND gate using DCVS, DSL, CNTL is shown in figure 2.2, 2.4, 2.7 respectively. Truth table of AND gate is given in table 4.1 and its logical output waveform is given in figure 4.1.

Table 4.1: Truth Table of AND Gate

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

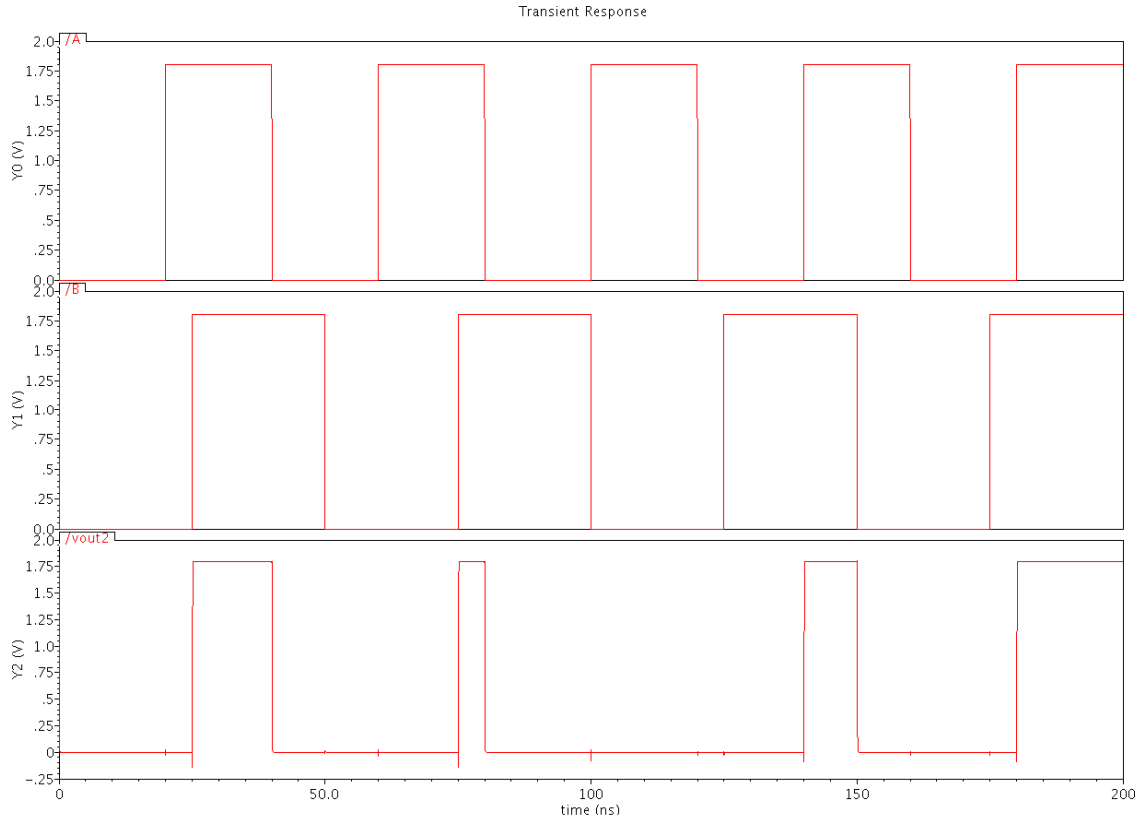


Figure 4.1: Logical Output Waveform of AND Gate

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; [3:2] Compressor and [4:2] Compressor.

4.3.1 [3:2] Compressor

[3:2] compressor is basically a 1-bit full adder. Before discussing full adder, we will first discuss half adder using DCVS, DSL, CNTL respectively which can be used as a component to make 1-bit full adder Considering that A and B are the input bits to be added, S is the sum output and C is the carry output, the truth table of the half adder cell is shown in Table 4.2.

Table 4.2: Truth Table of Half Adder

A	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

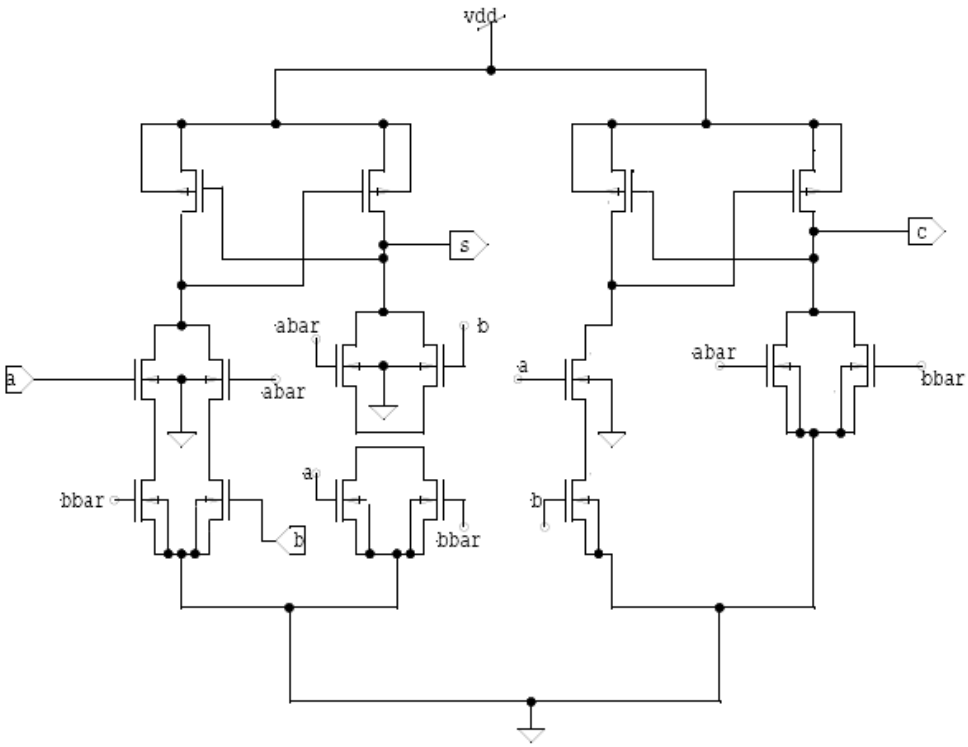


Figure 4.2: Schematic of DCVS Half Adder

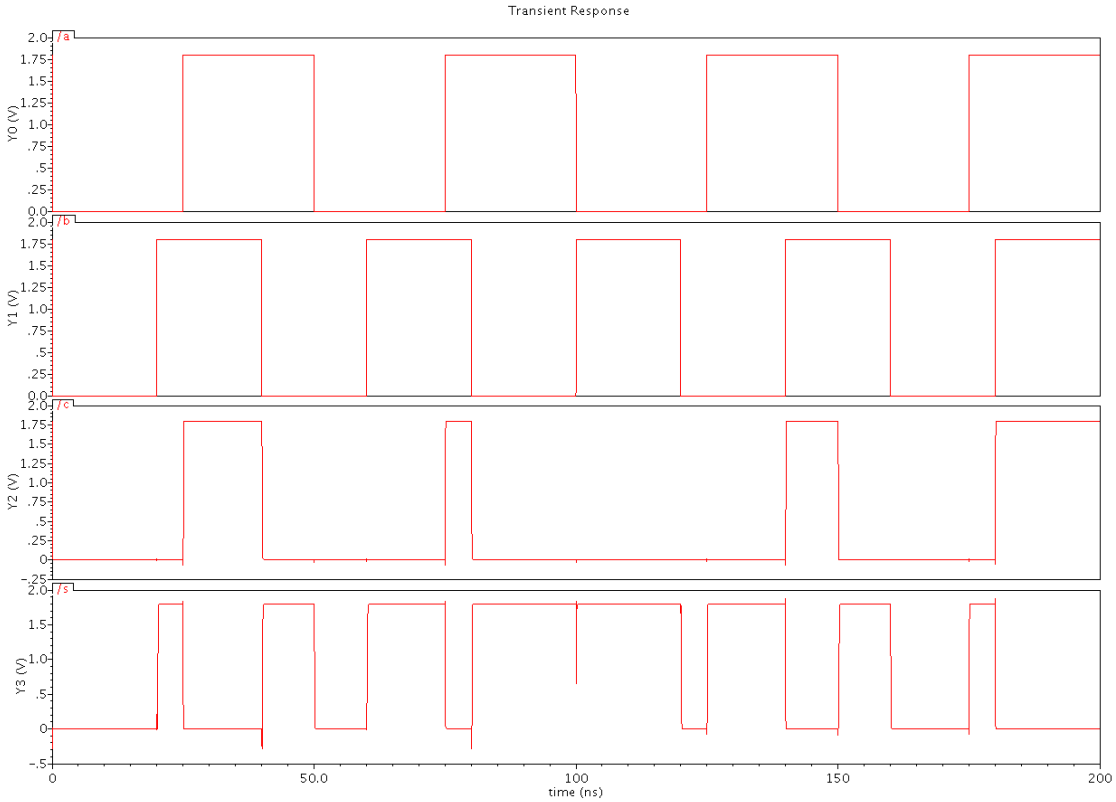


Figure 4.3: Logical Output Waveform of DCVS Half Adder

Now we will discuss about the DSL Half Adder. The schematic for DSL half adder is given in figure 4.4 below:

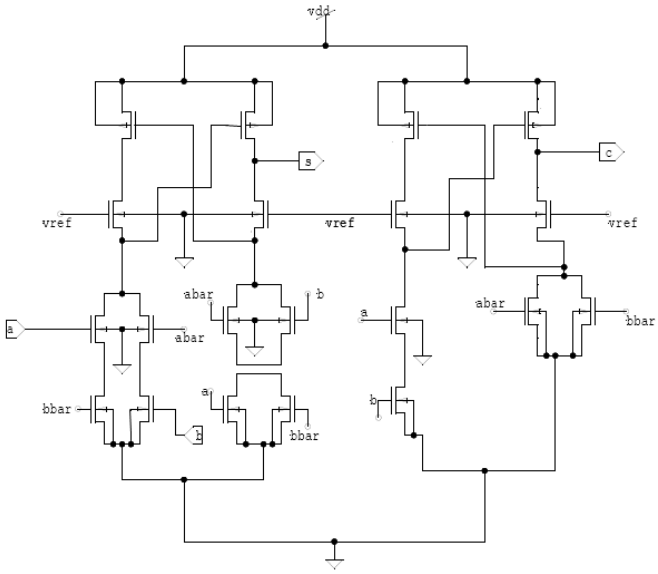


Figure 4.4: Schematic of DSL Half Adder

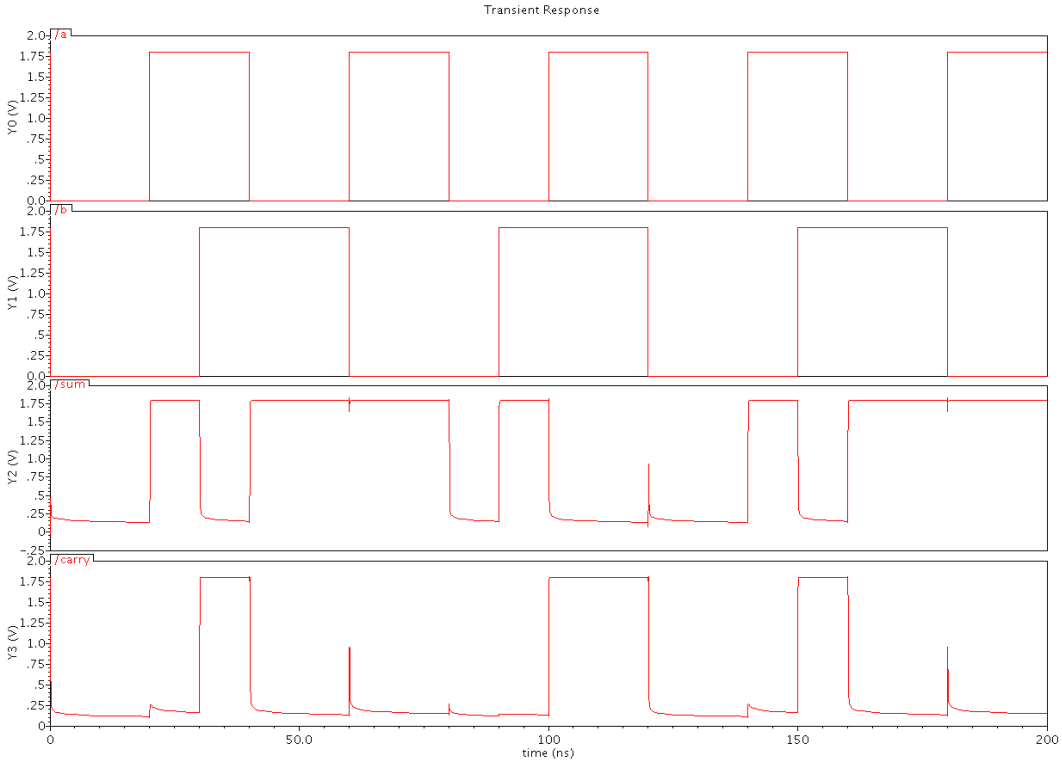


Figure 4.5: Logical Output Waveform of DSL Half Adder

We can see from the simulation result that we are not getting the full swing in DSL Half adder but the dynamic power consumed is less than the DCVS logic style.

Now the Half Adder in third logic style i.e. CNTL is discussed in figure 4.6 below:

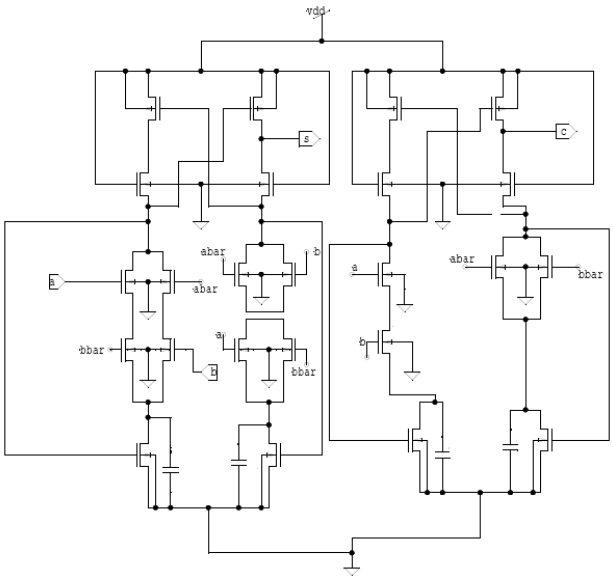


Figure 4.6: Schematic of CNTL Half Adder

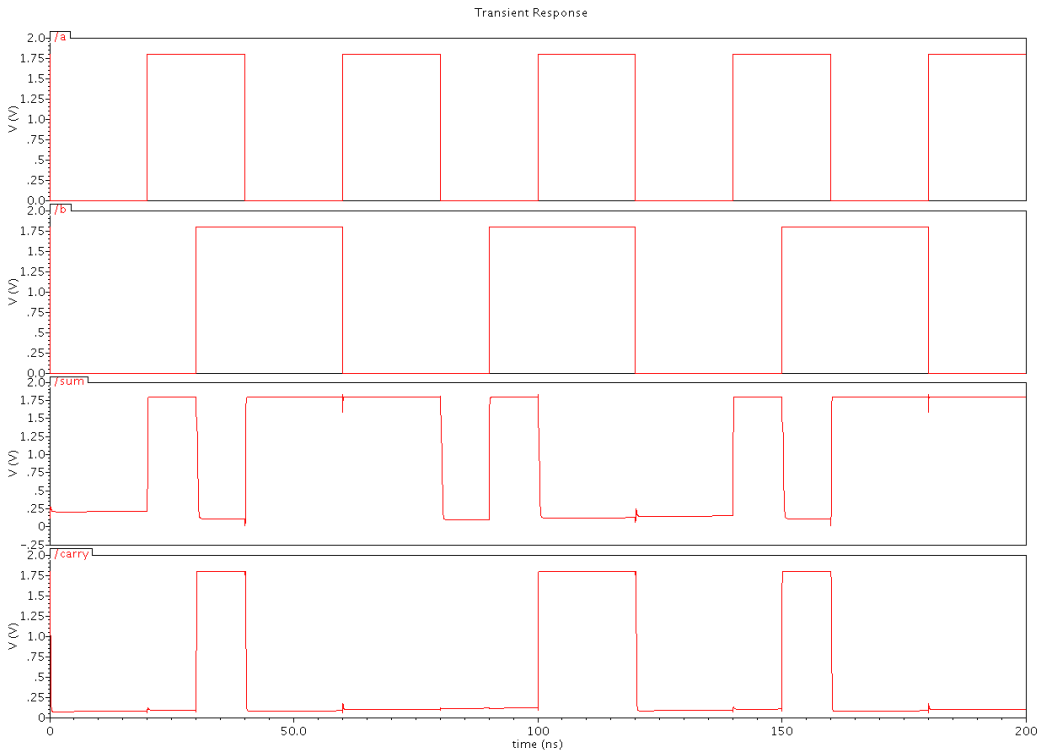


Figure 4.7: Logical Output Waveform of CNTL Half Adder

The output swing in CNTL is also less than the DCVS logic style but the dynamic power consumed is less than the DCVS logic style.

For 1-bit full adder we also need OR gate which will generate the carry output of the full adder. Truth table, schematic and simulation are given in table 4.3, figure 4.8 and figure 4.9 respectively for OR gate.

Table 4.3: Truth Table of OR Gate

A	B	Vout
0	0	0
0	1	1
1	0	1
1	1	1

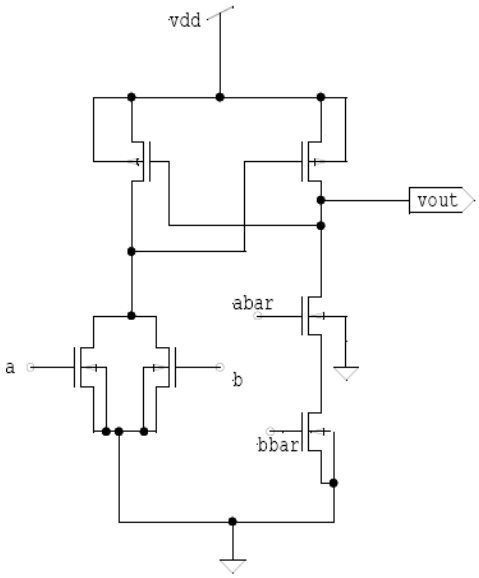


Figure 4.8: Schematic of OR Gate

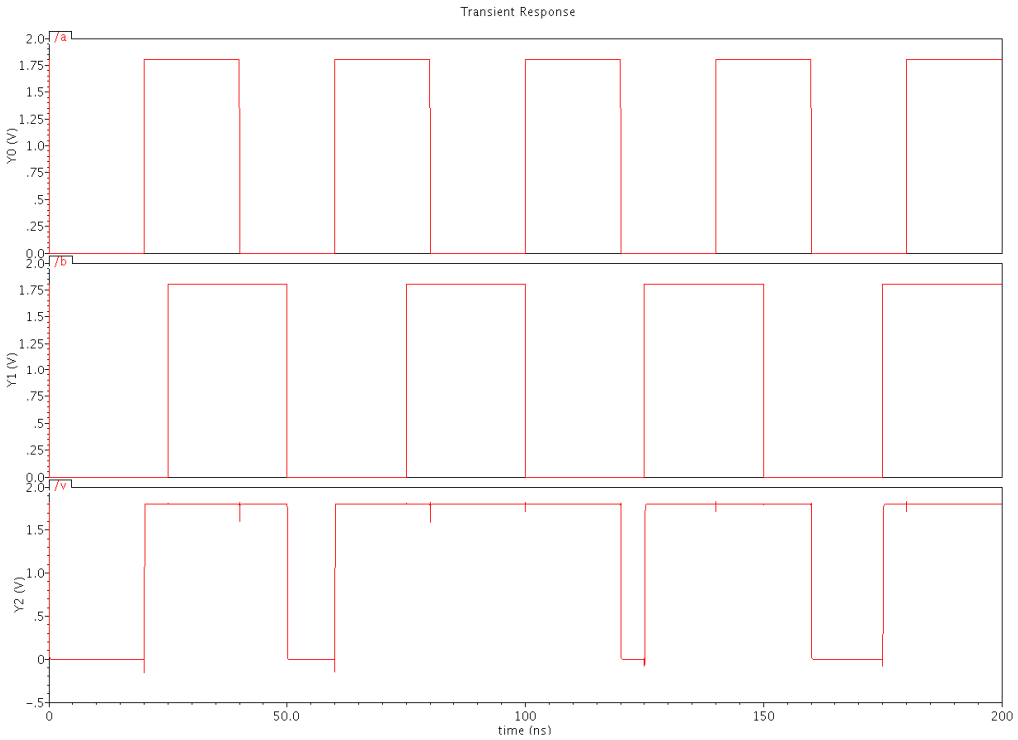


Figure 4.9: Logical Output Waveform of OR Gate

Block diagram of full adder is given in figure 4.10 below:

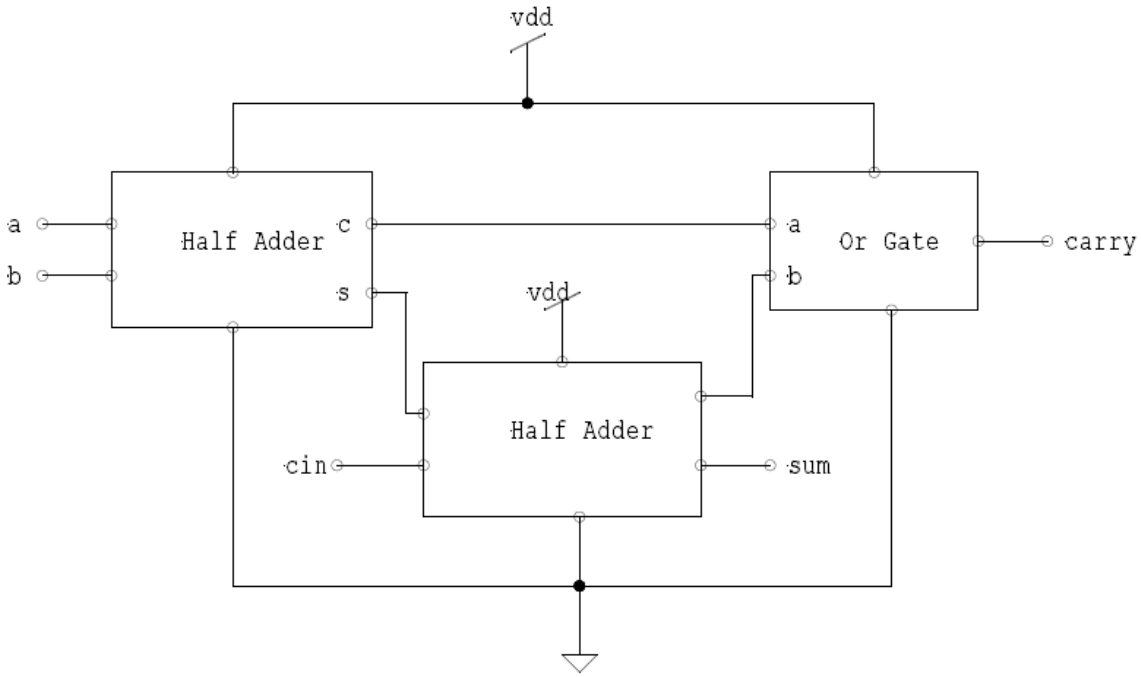


Figure 4.10: Block Diagram of Full Adder

Truth Table of full adder is given in table 4.4 below:

Table 4.4: Truth Table of Full Adder

a	B	cin	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

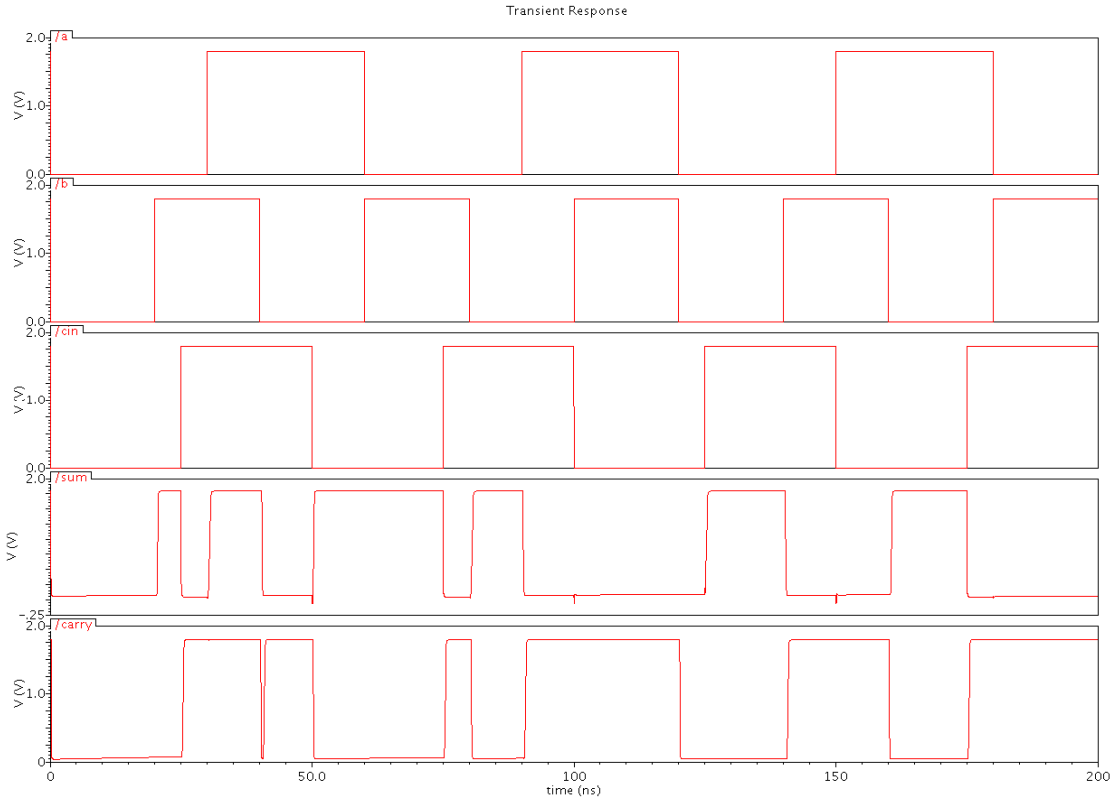


Figure 4.11: Logical Output Waveform of CNTL Full Adder

4.3.2 **[4:2] Compressor**

[4:2] compressor is shown in figure below. It takes four inputs of equal weight and produces two outputs. [4:2] compressors can be used in binary tree to produce a much more regular layout. It can be constructed from two [3:2] compressors and OR Gate. 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, first full adder takes in the three inputs to be added and the next full adder takes in the fourth bit to be added, sum of the first adder and third input is grounded. The carryout of first adder and the second adder is given as input to OR Gate. Output of the OR Gate is the carryout of the [4:2] compressor and the output of the second full adder is the sum of the [4:2] compressor. Block diagram of [4:2] compressor is shown in figure 4.12.

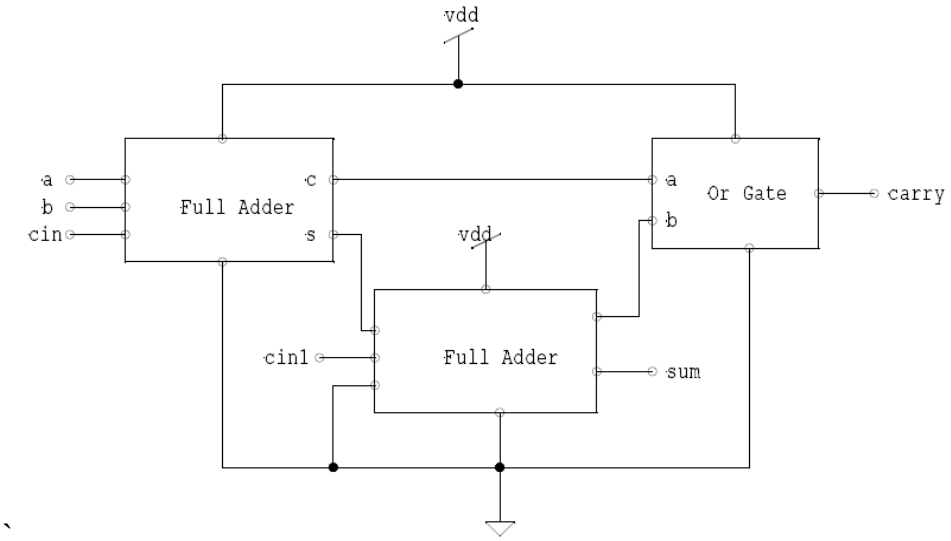


Figure 4.12: Block Diagram of [4:2] compressor

Figure 4.13 below shows the simulation results of [4:2] compressor.

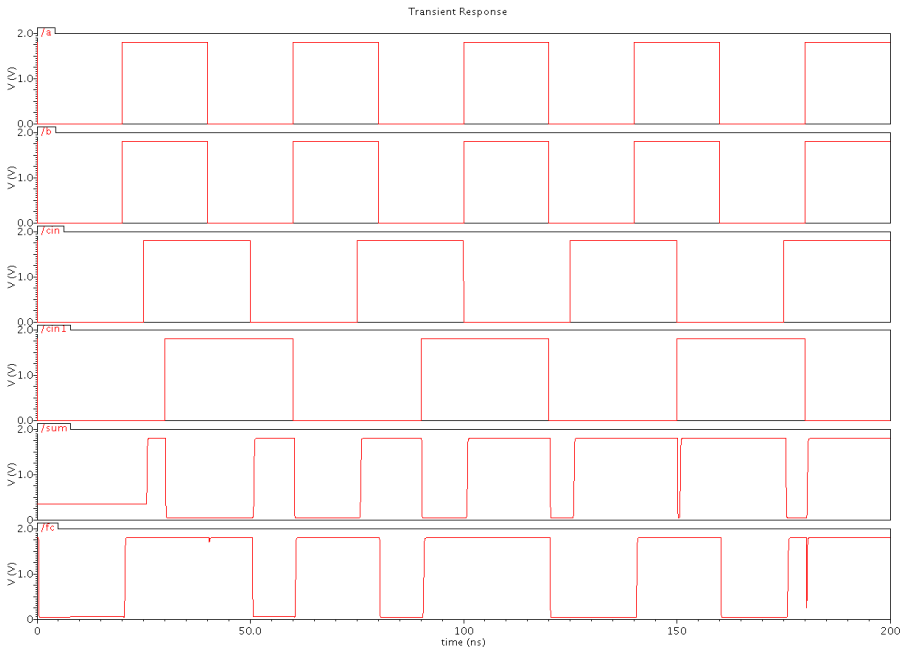


Figure 4.13: Logical Output Waveform of 4:2 compressors

After reduction of partial products as per Wallace Tree Multiplier algorithm, discussed in chapter 3, we have reduced 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 [9].

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 4.14 [9].

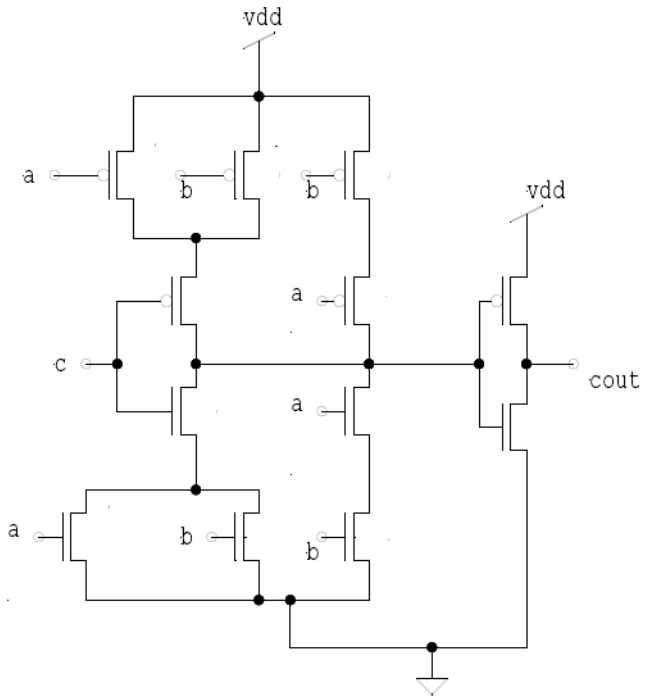


Figure 4.14: Schematic of carry propagate block [9]

So the design of 4-bit carry propagate adder is shown in figure 4.15. We can design any n-bit CPA adder by moving on the same lines. In my design i have designed a 12-bit carry propagate adder.

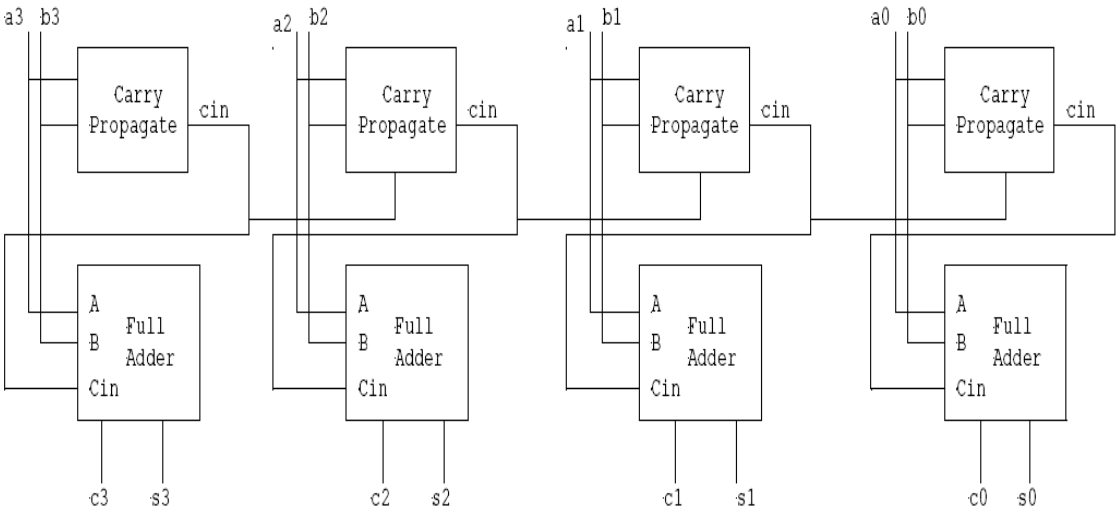


Figure 4.15: Schematic of carry propagate adder

Figure 4.16 shows the simulation waveforms for carry propagate adder.

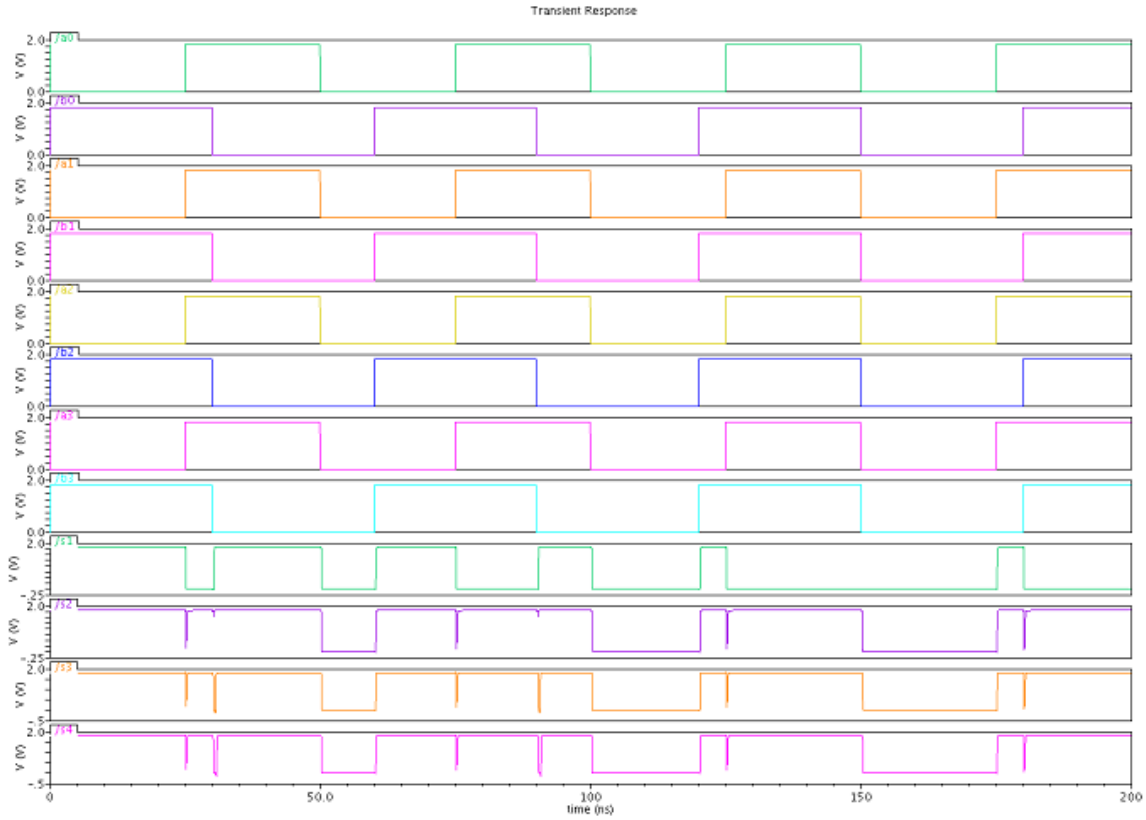


Figure 4.16: Logical Output Waveform of Carry Propagate Adder

So now after design each block we now can design a multiplier as per our requirement, we discuss the design of 8-bit multiplier.

As per the first step we would generate partial products, so we would 64 AND gates and place, as a part of second step we would place compressors and feed them with the required partial product input, this reduction would be completed in two steps . Finally we would use CPA to add the reduced partial products; we would require a 13-bit CPA in last stage. Figure 4.17 shows the block diagram of 8-bit multiplier.

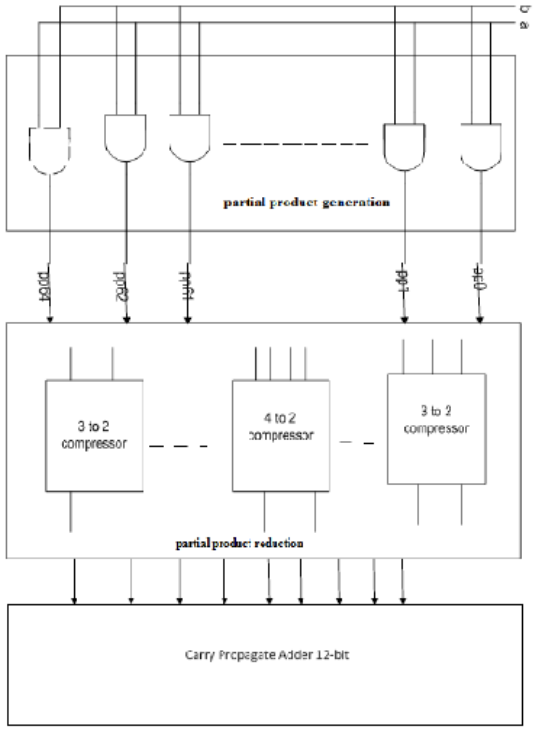


Figure 4.17: Block Diagram of 8-bit Multiplier

Figure 4.18 shows the simulated waveforms form 8-bit CNTL multiplier.

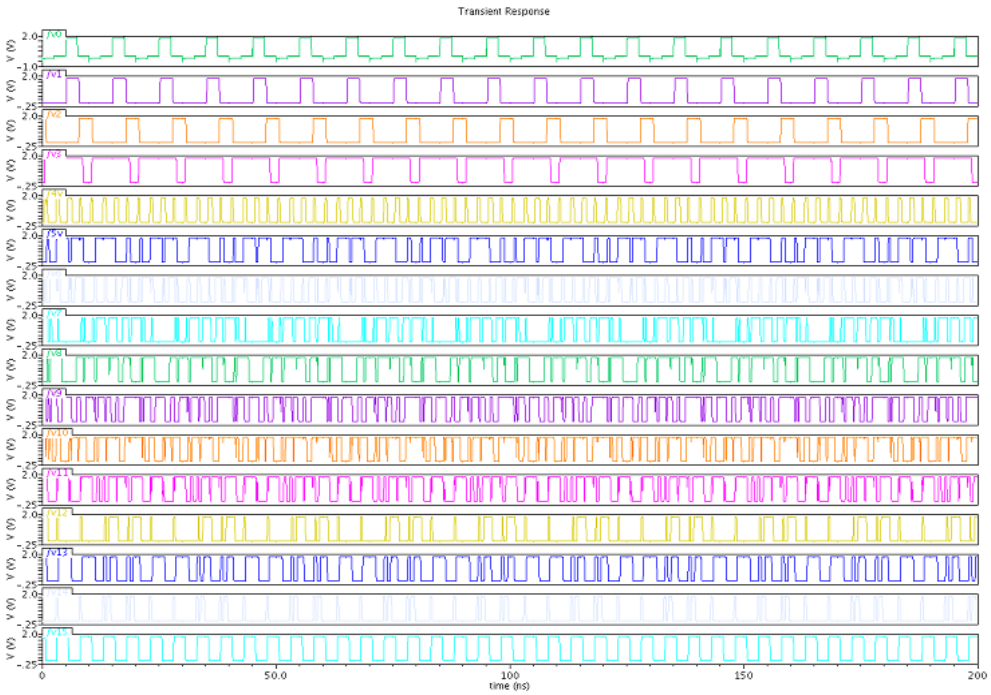


Figure 4.18: Logical Output Waveform of 8-bit CNTL Multiplier

CHAPTER 5 Physical Layout Design

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 of the multiplier [10].

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 i.e. the physical layout of the circuit.

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 [10].

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 tape out, 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 Checking

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. 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 circuit dimension in determining circuit speed) [10].

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.
Some example 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 Layout vs. Schematic (LVS)

The Layout vs. 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.

5.7 Physical Layout Design of Different Blocks of Multiplier

The physical layout design of different cells based on different non-clocked logic styles i.e. DCVS, DSL, CNTL is done in standard UMC 0.18 • m CMOS technology. For this project work, Cadence Corporation Ltd, Virtuoso was used for the design of physical layout along with spectre 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.

5.7.1 Layout Cell Design of Various Gates

The layout of various gates in different logic styles are shown in figure 5.1, 5.2, 5.3, 5.4

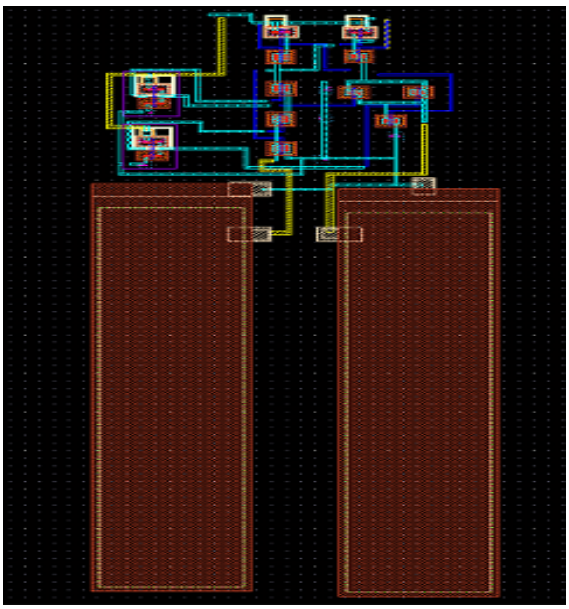


Figure 5.1: CNTL AND Gate layout

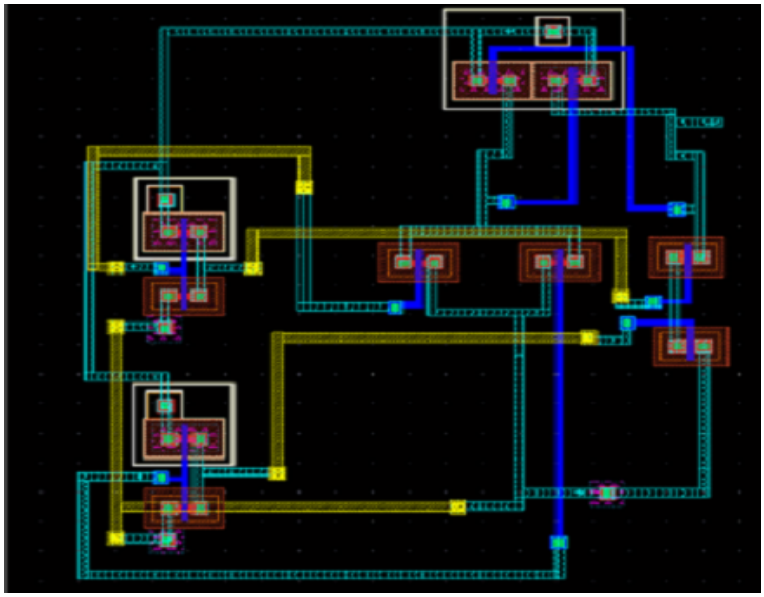


Figure 5.2: Layout of DCVS OR Gate

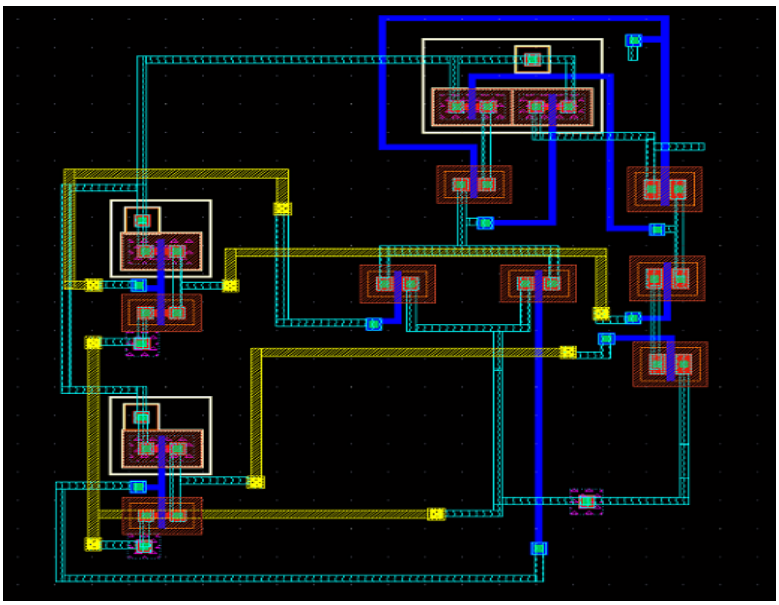


Figure 5.3: Layout of DSL OR Gate

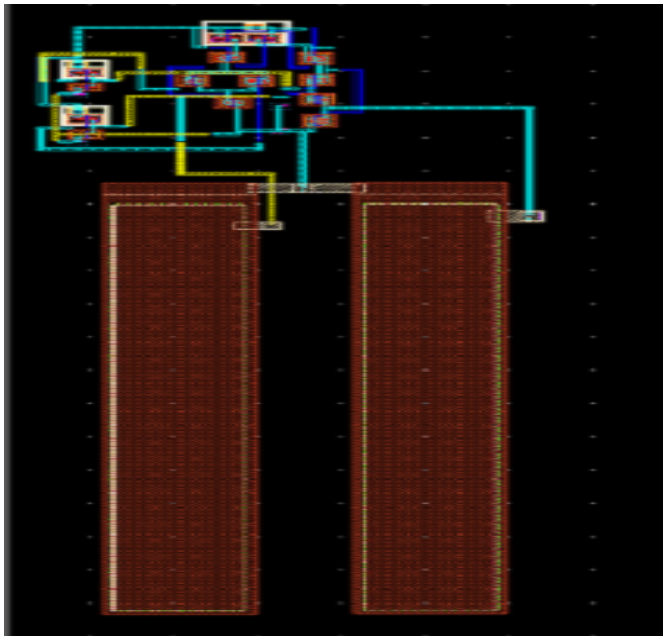


Figure 5.4: Layout of CNTL OR Gate

5.7.2 Layout Cell Design of Various Adders

The layout of half adders, full adders and [4:2] compressors in different logic styles is shown in figure 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11.

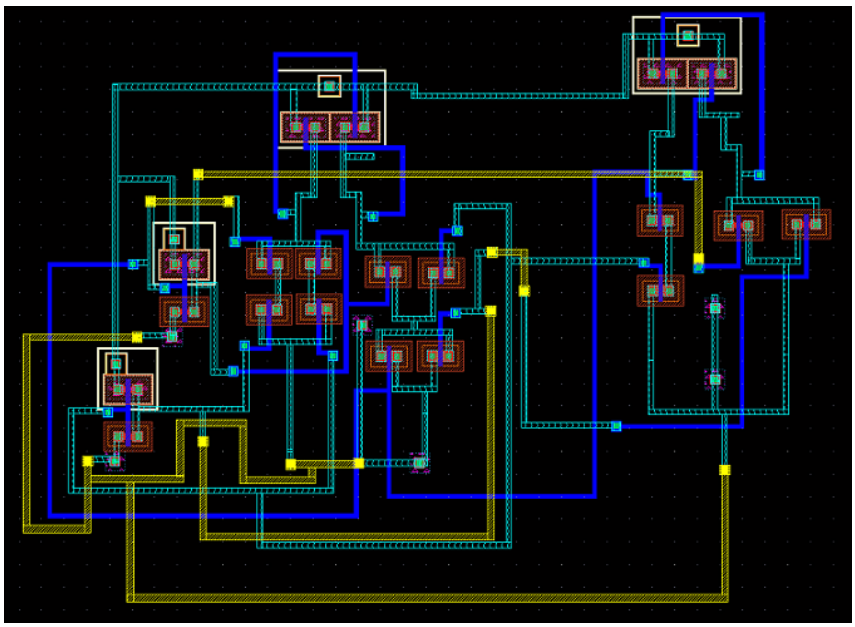


Figure 5.5: Layout of DCVS Half Adder

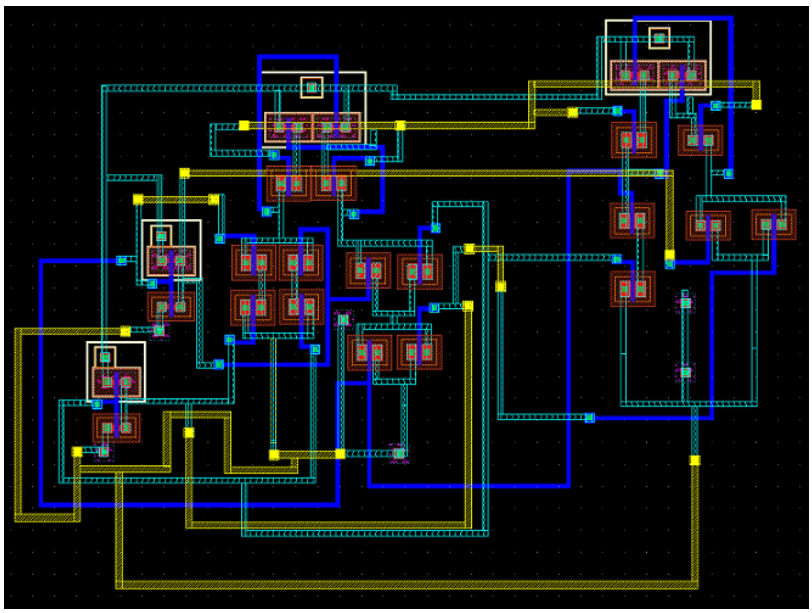


Figure 5.6: Layout of DSL Half Adder

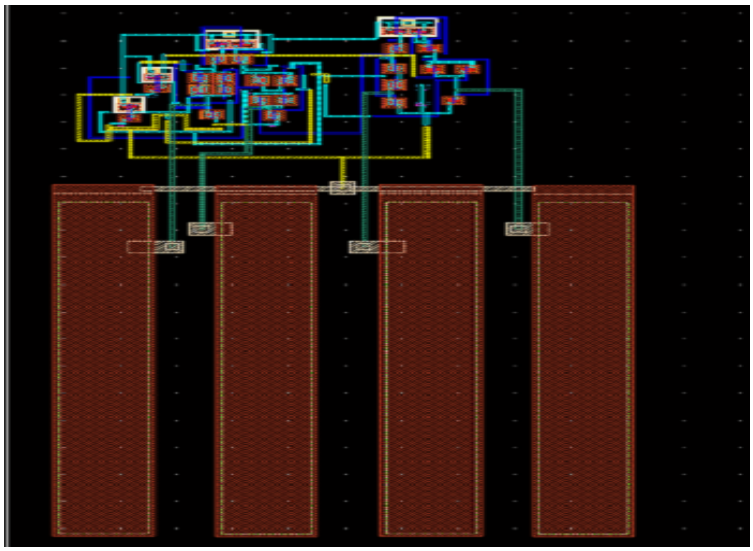


Figure 5.7: Layout of CNTL Half Adder

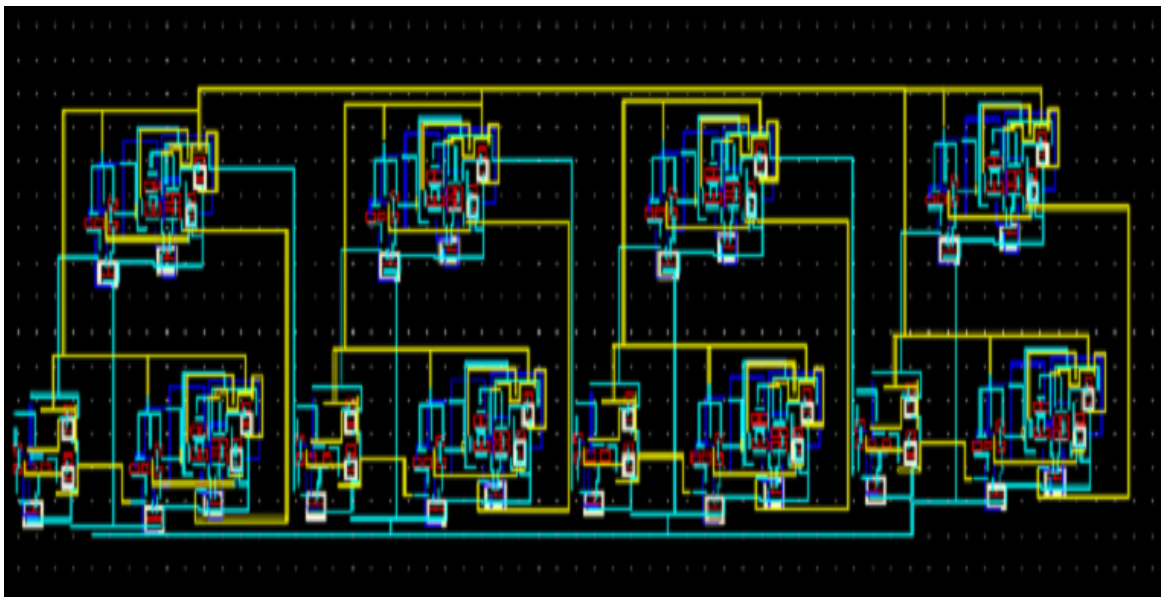


Figure 5.8: Layout of 4-bit DCVS Full Adder

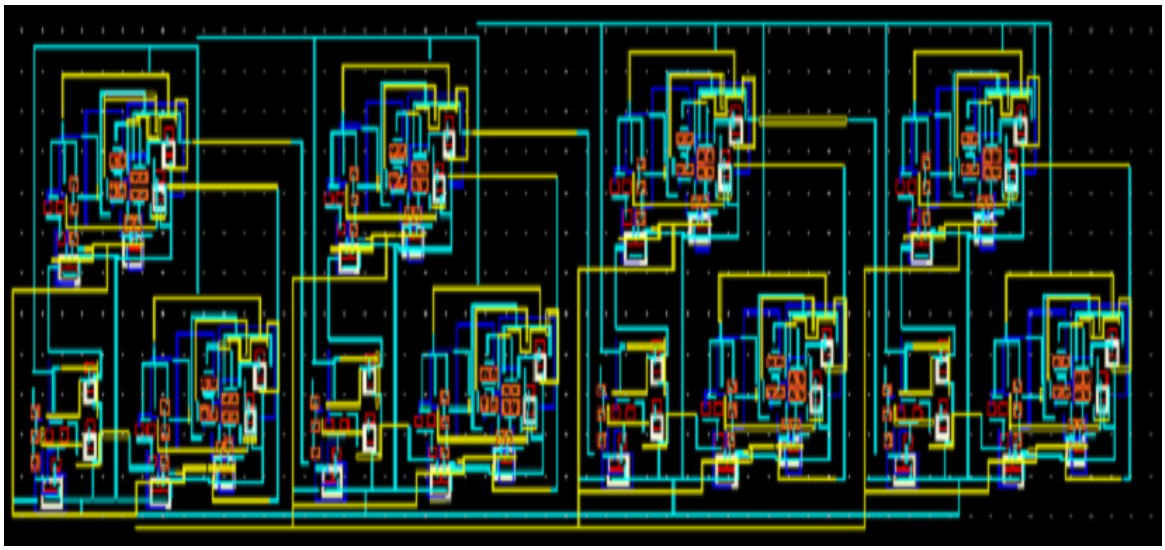


Figure 5.9: Layout of 4-bit DSL Full Adder

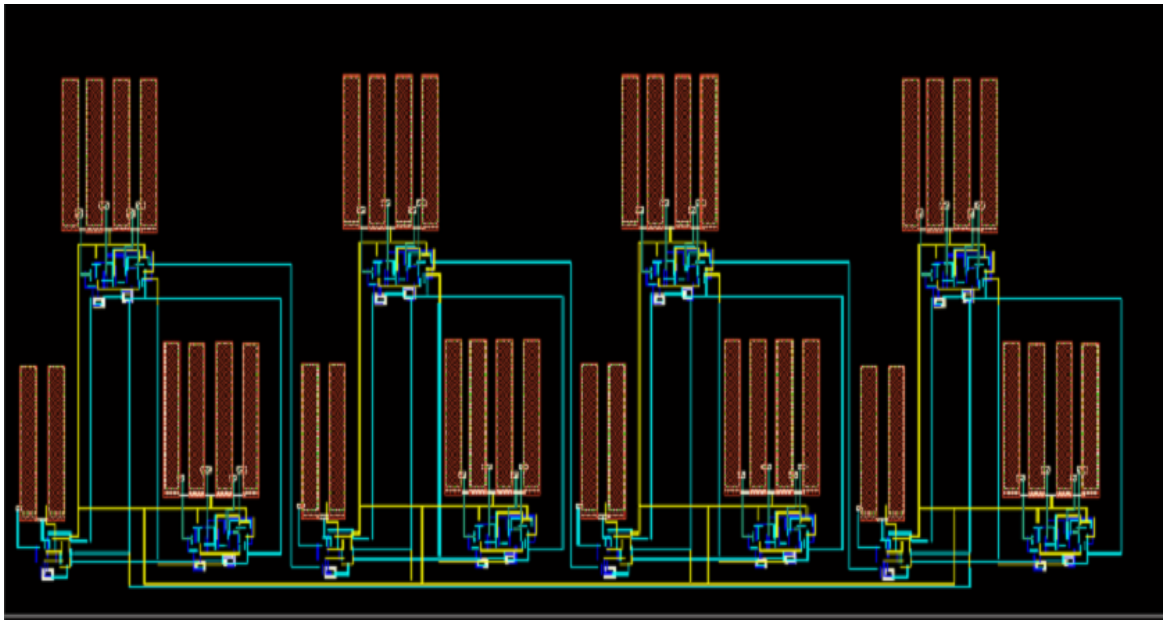


Figure 5.10: Layout of 4-bit CNTL Full Adder

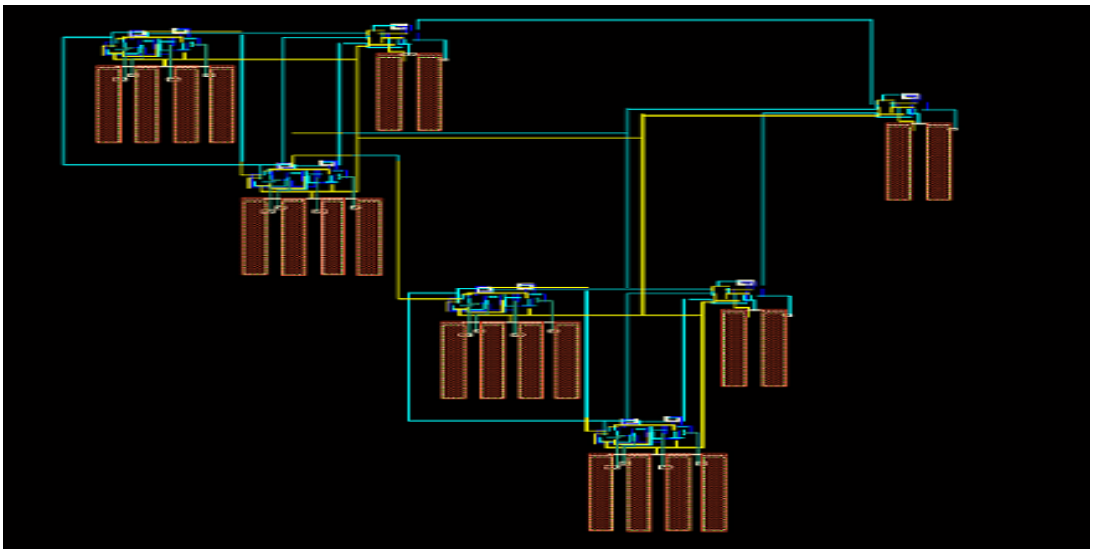


Figure 5.11: Layout of CNTL [4:2] Compressor

5.7.3 Layout Cell Design of Carry Propagate Block

The layout of carry propagate block and carry propagate adder is shown in figure 5.12, 5.13.

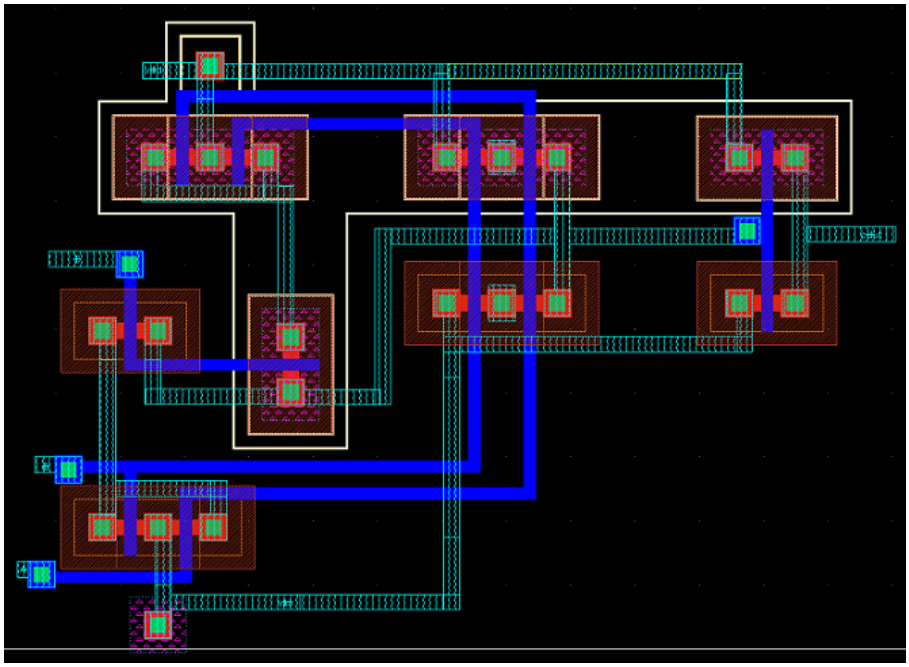


Figure 5.12: Layout of Carry Propagate Block

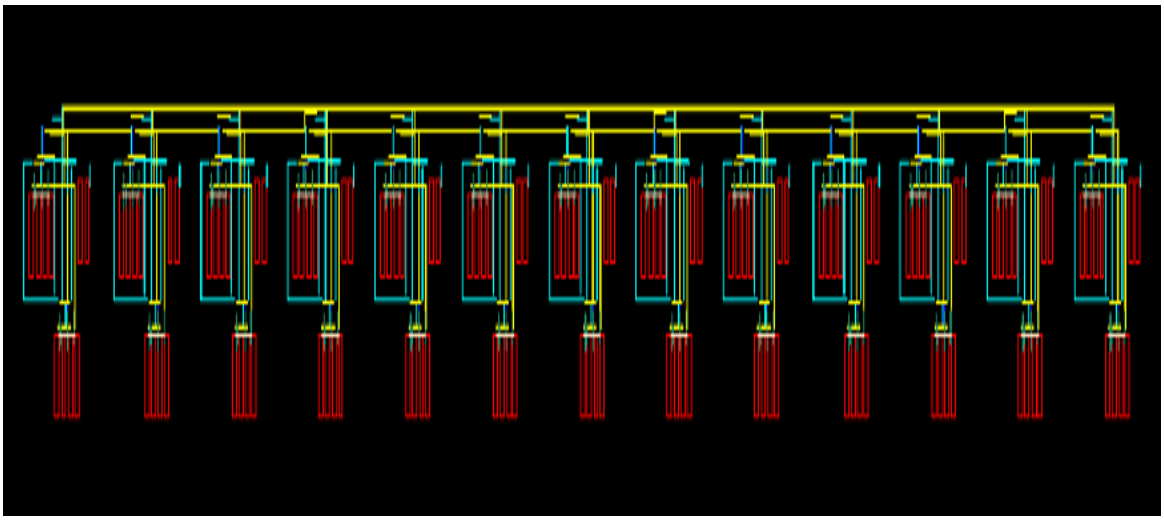


Figure 5.13: Layout of CNTL Carry Propagate Adder

5.7.4 Layout Cell Design of CNTL 8-bit multiplier

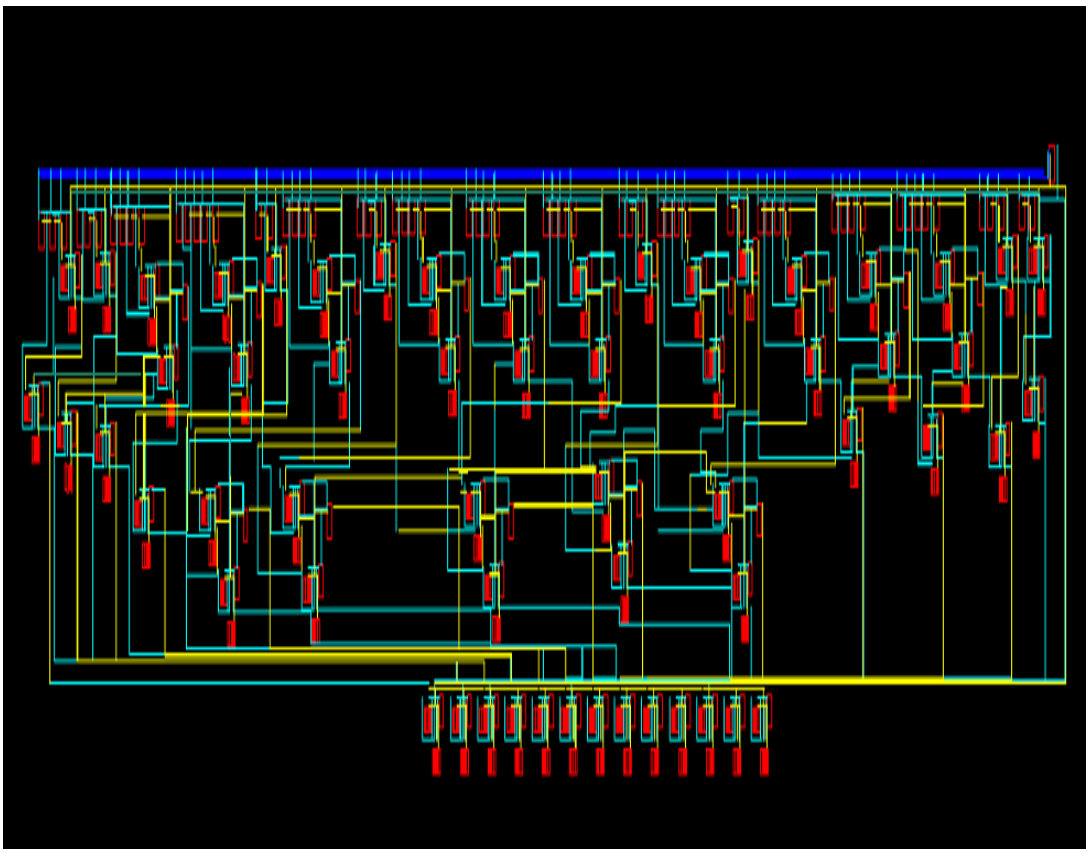


Figure 5.14: Layout of CNTL 8-bit multiplier

CHAPTER 6

SIMULATIONS AND RESULTS

6.1 Simulations Based on Schematic

The power curves for 1-bit full adder (DCVS, DSL and CNTL) are given below:

Curve (a) shows the total current I_D drawn from the circuit.

Curve (b) shows the total current drawn multiplied by V_{DD}.

Curve (c) shows the integration of $I_D.V_{DD.}$

Curve (d) shows the total power (Static and Dynamic) of the circuit.

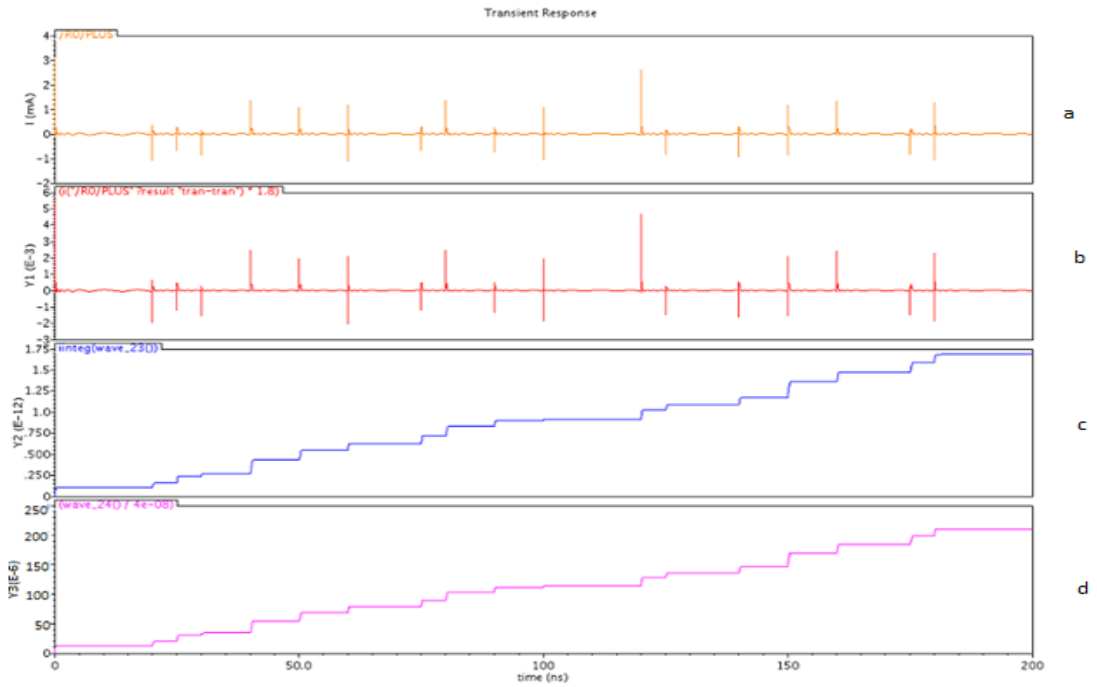


Figure 6.1: Power curve for 1-bit DCVS Full Adder

Figure 6.2 shows the power curve for 1-bit DSL Full Adder below:

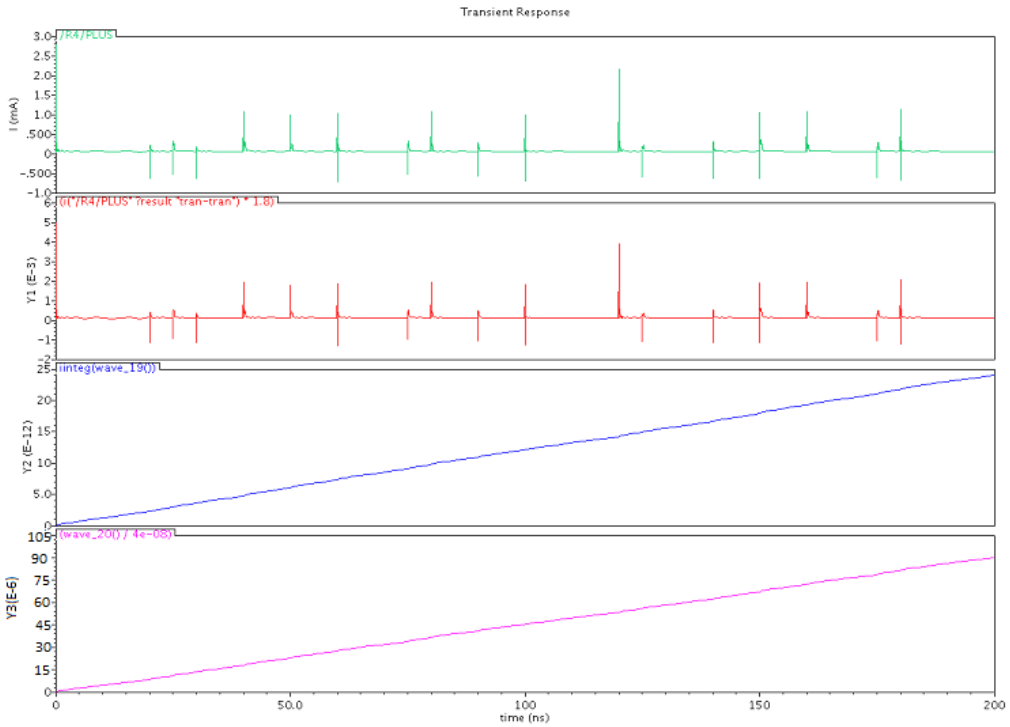


Figure 6.2: Power curve for 1-bit DSL Full Adder

Figure 6.3 shows the power curve for 1-bit CNTL Full Adder below:

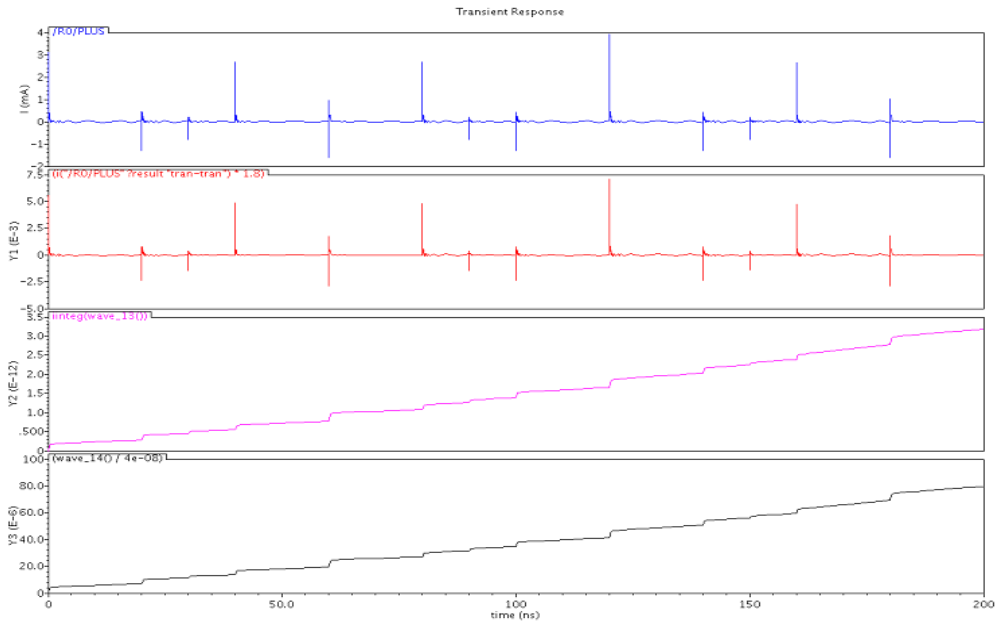


Figure 6.3: Power curve for 1-bit CNTL Full Adder

Table 6.1 shows the comparison of the power of 1-bit Full Adder (DCVS, DSL and CNTL):

Table 6.1: Power comparison of 1-bit Full Adder

Logic Styles	Static Power (µW)	Dynamic Power (µW) (25 MHZ)
DCVS	6.15	19.75
DSL	3.83	14.67
CNTL	1.78	13.23

Power curves for 4-bit Full Adder are given below:

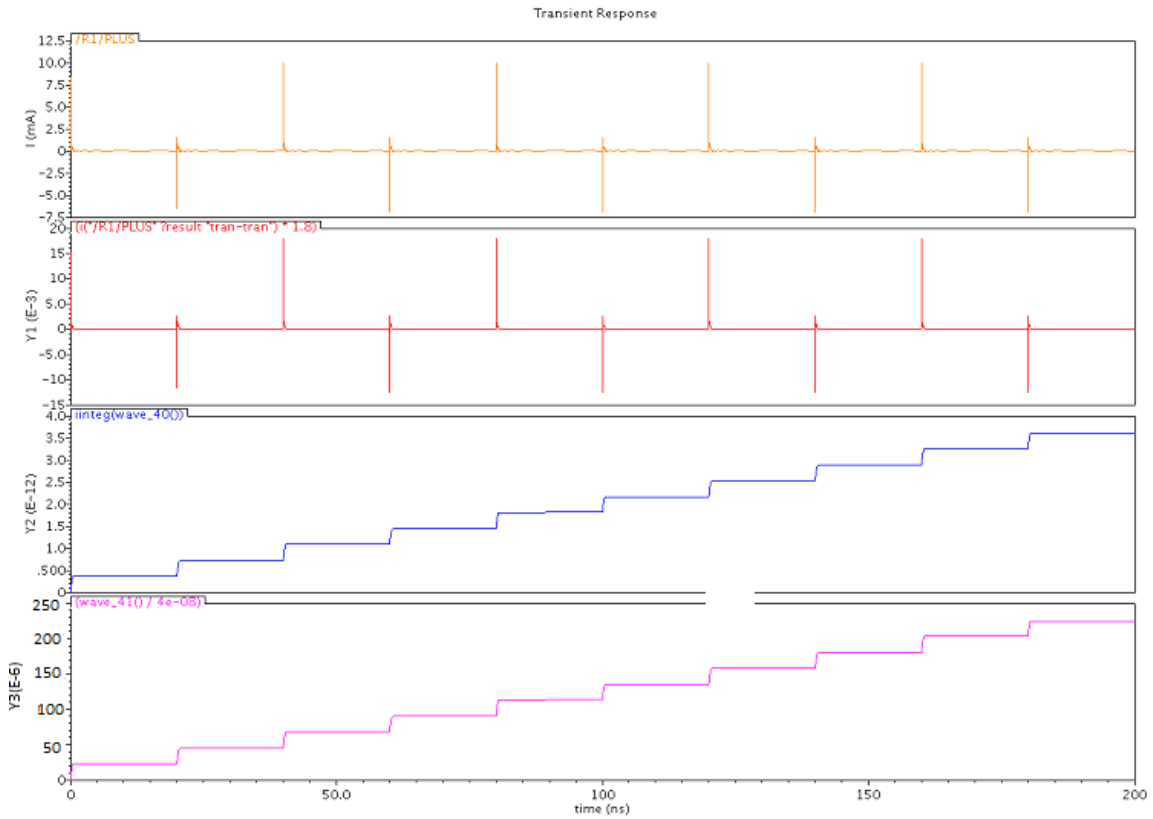


Figure 6.4: Power curve for 4-bit DCVS Full Adder

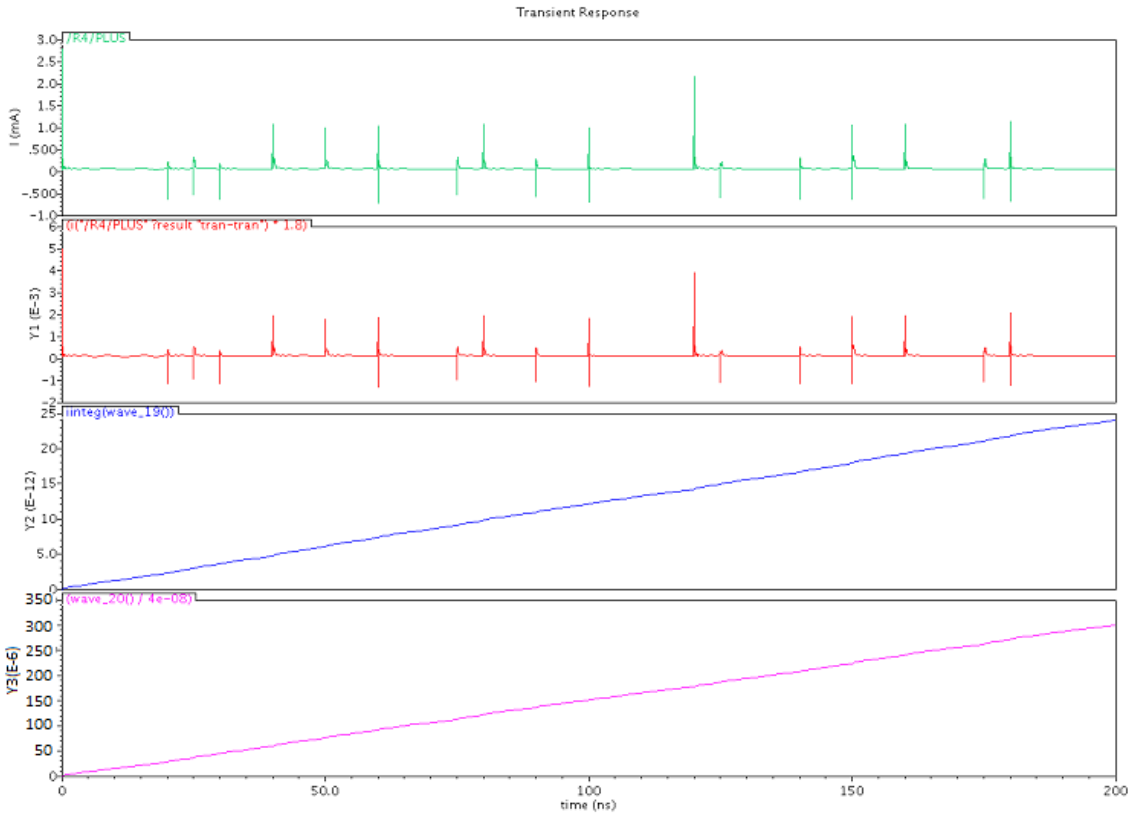


Figure 6.5: Power curve for 4-bit DSL Full Adder

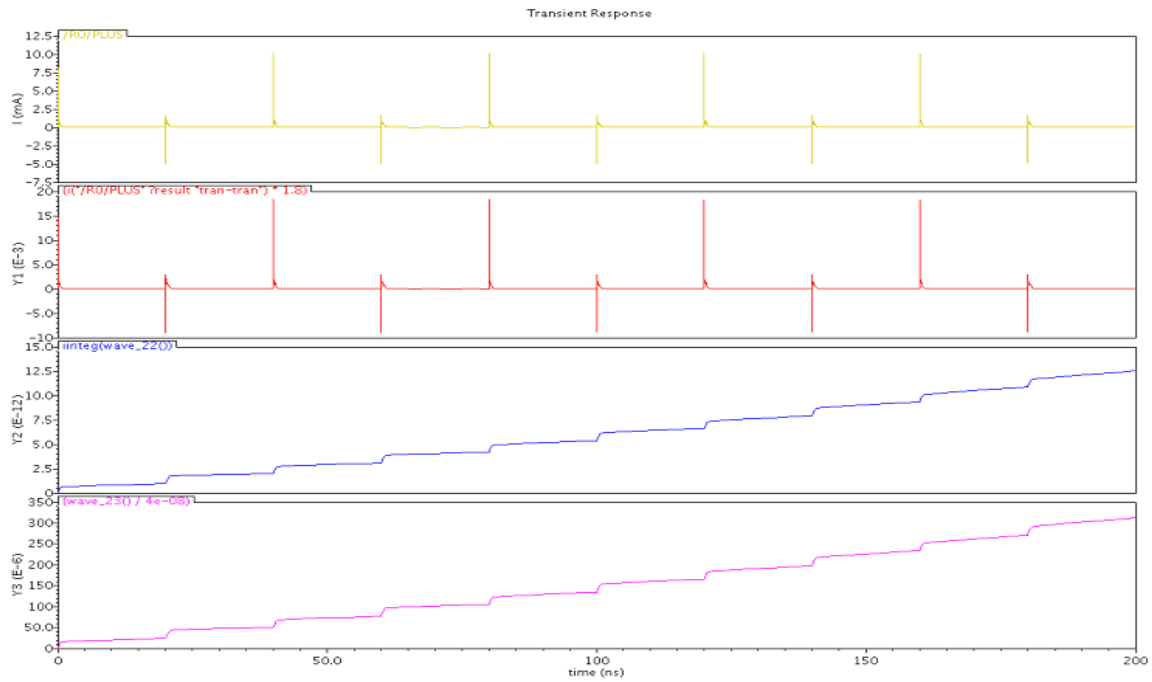


Figure 6.6: Power curve for 4-bit CNTL Full Adder

Table 6.2 shows the comparison of the power of 4-bit Full Adder (DCVS, DSL and CNTL):

Table 6.2: Power comparison of 4-bit Full Adder

Logic Styles	Static Power (µW)	Dynamic Power (µW) (25 MHZ)
DCVS	25.19	66.59
DSL	11.24	50.23
CNTL	5.48	45.62

As we can see from Table 6.1 and Table 6.2 the power of 4-bit Full Adder is approximately four times the power of 1-bit Full Adder.

6.2 Post Layout Simulations

The post layout power curves for 4-bit Full Adder are given below:

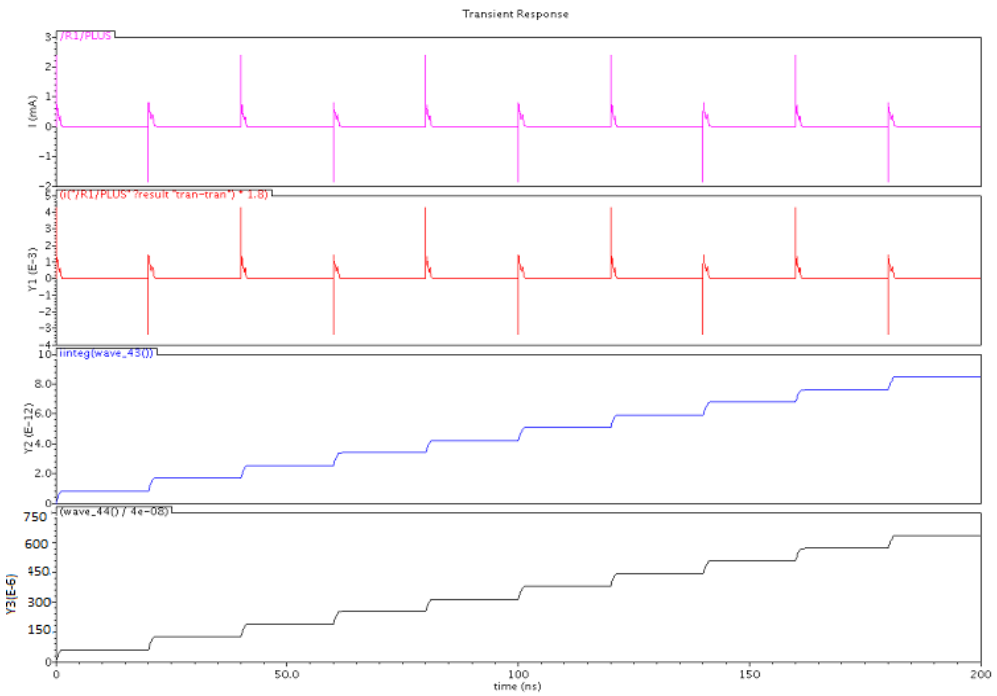


Figure 6.7: Post Layout power curve for 4-bit DCVS Full Adder

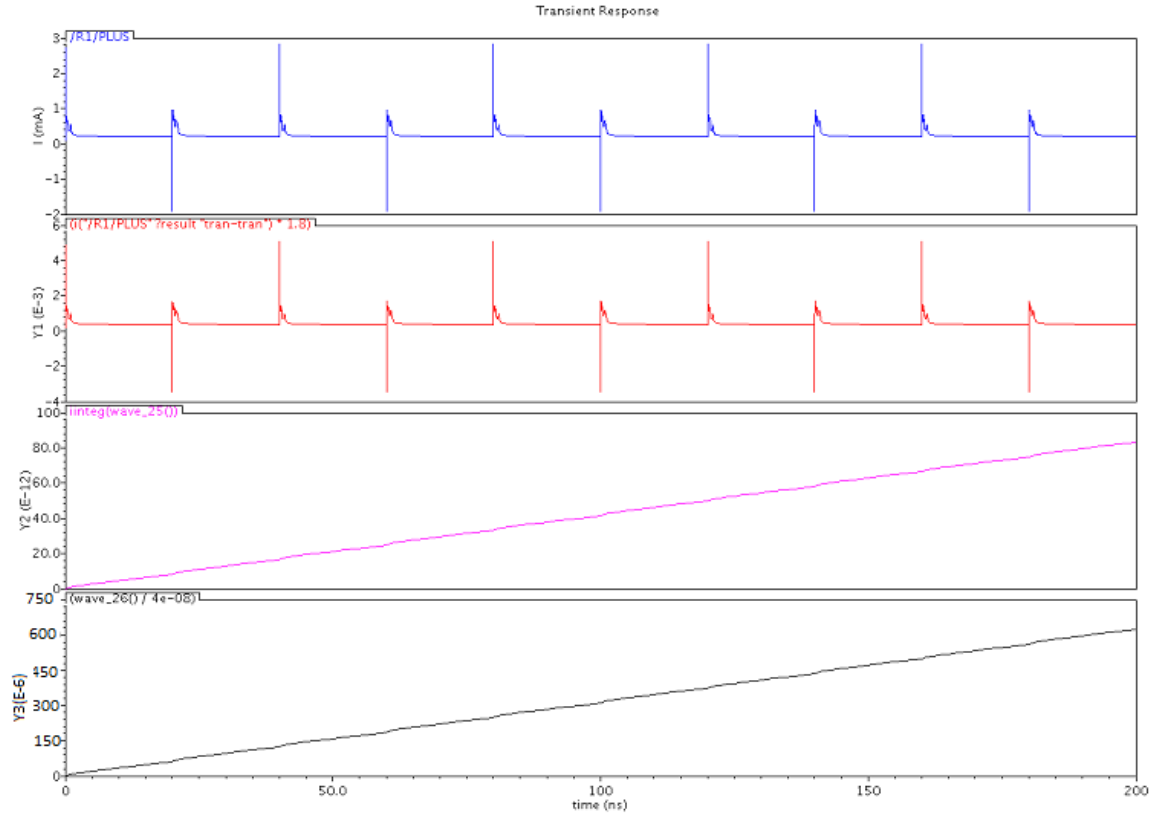


Figure 6.8: Post Layout power curve for 4-bit DSL Full Adder

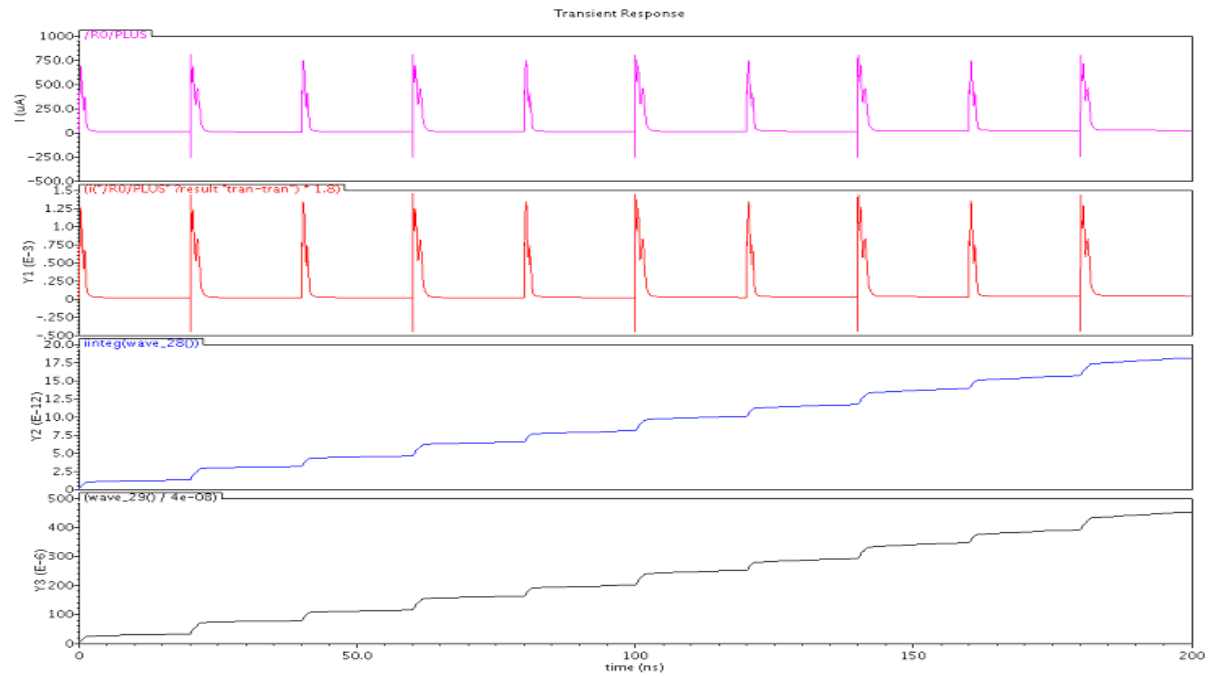


Figure 6.9: Post Layout power curve for 4-bit CNTL Full Adder

Table 6.3 shows the post layout power comparison for the three 4-bit Full Adder:

Table 6.3: Post Layout power comparison of 4-bit Full Adder

Logic Styles	Static Power (µW)	Dynamic Power (µW) (25 MHZ)
DCVS	25.19	72.24
DSL	11.24	56.47
CNTL	5.48	51.43

6.3 Simulations Based on Schematic for 8-bit Multipliers

Power curve for 8-bit CNTL multiplier is given below:

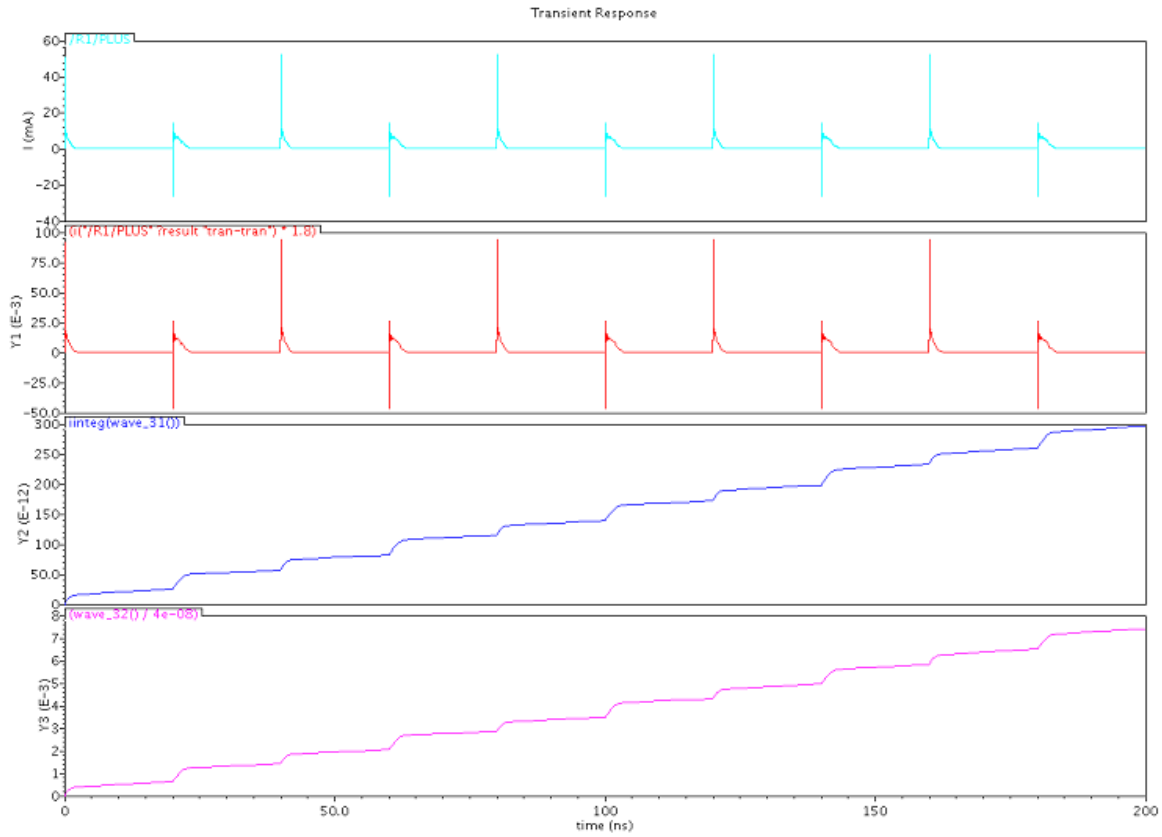


Figure 6.10: Power curve for 8-bit CNTL Multiplier

Power curve for 8-bit DSL multiplier is given below:

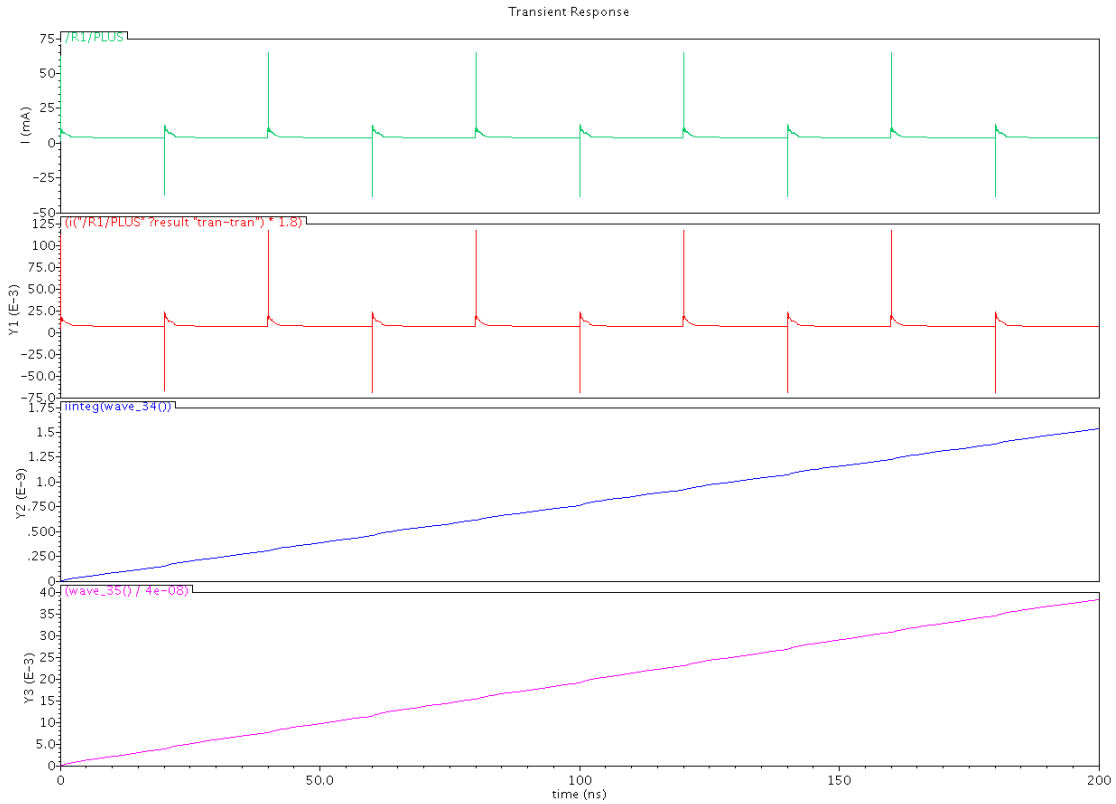


Figure 6.11: Power curve for 8-bit DSL Multiplier

Power curve for 8-bit DCVS multiplier is given below:

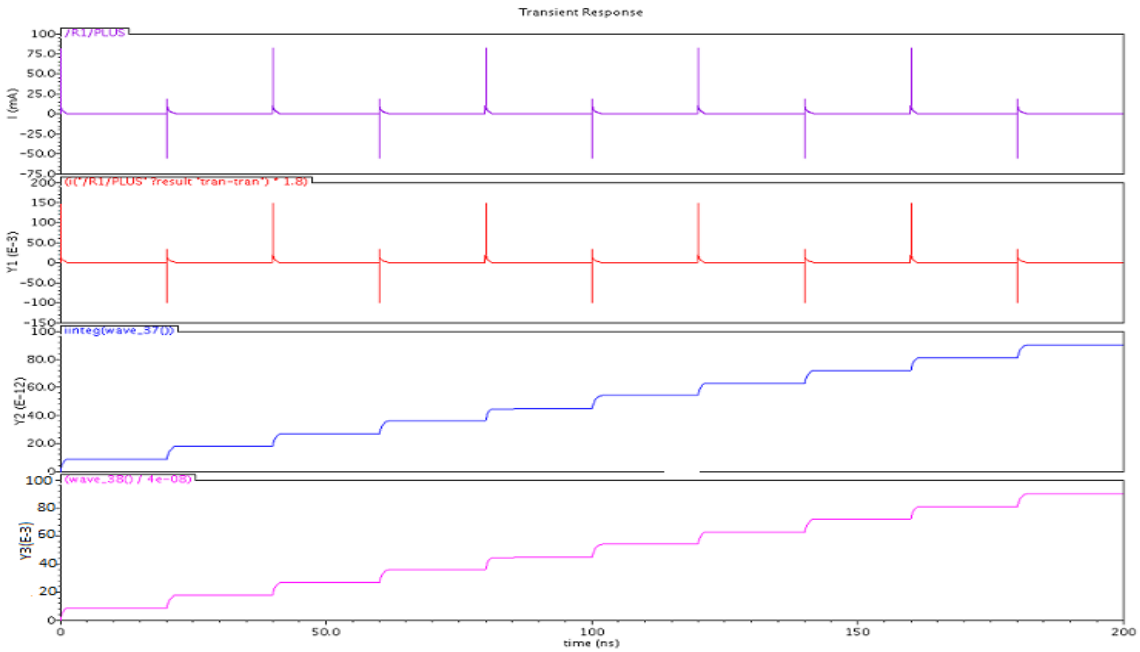


Figure 6.12: Power curve for 8-bit DCVS Multiplier

Table 6.4 shows the power comparison for the three 8-bit Multipliers:

Table 6.4: Power comparison of 8-bit Multipliers

Logic Styles	Power Dissipated(mW) (At 25 MHZ)
DCVS	14.7
DSL	6.66
CNTL	2.42

6.4 Post Layout Simulation 8-bit CNTL Multiplier

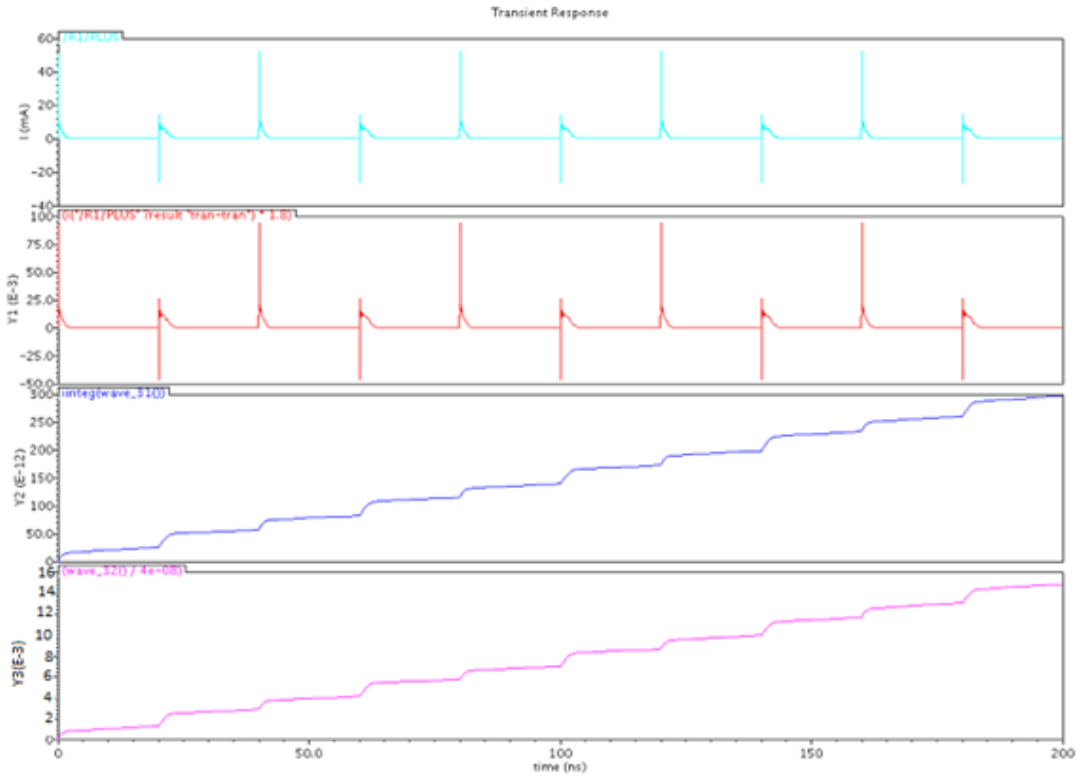


Figure 6.13: Post Layout power curve for 8-bit CNTL Multiplier

The power after post layout simulation of 8-bit multiplier comes out to be 3.07 mW which is more than the schematic level simulation because parasitic capacitances come into picture in post layout simulation.

CHAPTER 7

CONCLUSIONS AND FUTURE SCOPE

7.1 CONCLUSIONS

The primary goal of this thesis work is to show a successful try in terms of reduction of power dissipation. The low power CMOS cell structures as like a two-input AND gate, a two-input OR gate, Half Adders and eventually the 8-bit Multipliers are designed using different non-clocked logic styles that are DCVS, DSL, CNTL. Out of these CNTL is the best logic style on the basis of power.

The entire CMOS cell structures which are designed in this thesis work are designed in Cadence Schematic Editor in Virtuoso Analog Design environment using standard UMC 0.18 • m Technology. Because of the main concern is power dissipation so after the schematic design and the simulation, power dissipation for all the logic styles are compared with each other. All the circuits operate at a supply voltage of 1.8 V.

With CNTL technique, the circuit consumes less dynamic power and hence is power efficient. It saves approximately 70% of power as compared to other techniques.

7.2 FUTURE SCOPE OF WORK

Ø This thesis work is based on combinational circuit design but these techniques can also be used in sequential circuits. In low power VLSI design CNTL technique has very good scope for future because of less power consumption.

REFERENCES

[1] A. P. Chandrakasan, S. Sheng and R. W. Brodersen, "Low Power CMOS Digital Design," *IEEE Journal of Solid State Circuits*, Vol. 27, No. 4, pp. 473-484, April 1999.

[2] K. Bernstein, K. M. Carrig, Patrick R. Hansen " High Speed CMOS Design Styles" 5th edition, Kluwer Academic Publishers, 2001.

[3] L.G Heller, *et. al*., "Cascode Voltage Switch Logic: A Differential CMOS Logic Family", *Proceedings of 1984 IEEE International Solid State Circuits Conference*, pp. 16-17.

[4] Leo C.M.G. Pfennings, *et. al.*, "Differential Split Level CMOS Logic for Subnanosecond Speeds," *IEEE Journal of Solid State Circuits*, Vol. SC-20, No.5, October 1985.

[5] J. S. Wang, *et. al*., "CMOS Non-threshold Logic (NTL) and Cascode Nonthreshold Logic (CNTL) for High-Speed Applications," *IEEE Journal of Solid State Circuits*, Vol.24, No.3, June 1989, pp. 779-786.

[6] 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.

[7] K.Z. Peckmestzi "Multiplexer-Based Array Multipliers" *IEEE Transactions on Computers*, Vol.48, No.1, January 1999.

[8] Gary W. Bewdick, "Fast Multiplication Algorithms and Implementation" Thesis work for degree of masters in Stanford University, February 1994.

[9] N. Weste and K. Eshraghian, "Principal of CMOS Digital Design", 2nd Edition, Pearson Education, 2005.

[10] R. J. Baker, H. W. Li, and D. E. Boyce, "CMOS Circuit Design, Layout, and Simulation," IEEE Press Series on Microelectronic Systems, 2nd Edition, 1998.

[11] J. M. Rabaey, A. Chandrakasan, B. Nikolic, "Digital Integrated Circuits", 2nd Edition, Prentice Hall, 2002.

[12] Sung-Mo Kang, Yusuf Lablebici, "CMOS Digital Integrated Circuits: Analysis and Design", 3rd Edition Tata Mc-Graw Hill, 2003.

[13] H. J. M. Veendrick, "Short-circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits," *IEEE Journal of Solid State Circuits* , pp. 468-473, August 1984.

[14] Kaushik Roy, Sharat C. Prasad, "Low-Power CMOS VLSI Circuit Design", John Wiley & Sons, Inc, 1st Edition 2000.

[15] John P. Uyemura, "Introduction to VLSI Circuits and Systems," Wiley, John & Sons, Inc., 2002.

[16] Douglas A.Pucknell and Kamran Eshraghian, "Basic VLSI Design" 3rd Edition Prentice Hall of India, 2005.