

HIGH SPEED RADIX-2 BUTTERFLY STRUCTURE USING NOVEL WALLACE MULTIPLIER

Dissertation submitted in fulfillment of the requirements for the Degree of

MASTERS OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION

By

GARIMA THAKUR

Enrollment No.162005

Under the Supervision of

DR. SHRUTI JAIN

AND

DR. HARSH SOHAL



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY WAKNAGHAT, SOLAN -
173234, INDIA MAY, 2018

TABLE OF CONTENTS

DECLARATION BY THE SCHOLAR.....	v
CERTIFICATE.....	vi
ACKNOWLEDGEMENT.....	vii
ABSTRACT.....	viii
LIST OF FIGURES.....	x
LIST OF TABLES	xiii
ABBREVIATIONS.....	xiv
CHAPTER 1 INTRODUCTION.....	1
1.1 Adders.....	1
1.2 Multiplier.....	2
1.3 FFT.....	3
1.4 Need for Low Power Design.....	3
1.5 Programming Language.....	5
1.6 Research Approach.....	7
1.7 Organization.....	7
1.8 Tools Used.....	8
CHAPTER 2 LITERATURE SURVEY.....	9
CHAPTER 3 ADDERS.....	13
3.1 Basic Adder.....	13
3.1.1 Half Adder.....	13
3.1.2 Full Adder.....	14

3.2 Complex Adder.....	15
3.2.1 Ripple Carry Adder.....	15
3.2.2 Carry Select Adder.....	16
3.2.3 Carry Skip Adder.....	17
3.2.4 Carry Look Ahead Adder.....	18
3.2.5 Kogge Stone Adder.....	21
3.3 Implementation of 4-bit Adders.....	24
3.4 Proposed Adder.....	26
3.5 Comparison Table.....	31
3.5 Conclusion.....	31
CHAPTER 4 MULTIPLIERS.....	32
4.1 Different Multipliers.....	33
4.1.1 Array Multiplier.....	33
4.1.2 Wallace Multiplier.....	36
4.1.3 Vedic Multiplier.....	38
4.2 Implementation of 4-bit Multiplier.....	41
4.3 Proposed Multiplier.....	42
4.4 Conclusion.....	48
CHAPTER 5 FAST FOURIER TRANSFORMS.....	49
5.1 Implementation of 16 and 32 bit efficient Adder.....	49
5.2 Implementation of 16 and 32 bit efficient Multiplier.....	52

5.3 Transforms.....58

 5.3.1 Butterfly Structure.....61

5.4 Conclusion.....63

CONCLUSION.....67

REFERENCES.....68

PUBLICATION.....71

DECLARATION BY THE SCHOLAR

I hereby declare that the work reported in the Masters Of Technology thesis entitled "**High Speed Radix-2 Butterfly Structure using Novel Wallace Multiplier**" submitted at **Jaypee University of Information Technology, Wagnaghat India**, is an authentic record of my work carried out under the supervision of **Dr. Shruti Jain and Dr. Harsh Sohal**. I have not submitted this work elsewhere for any other degree or diploma.

Garima
Garima Thakur

ECE Department

JUIT

Date: 10-05-2018



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

(Established by H.P. State Legislative vide Act No. 14 of 2002)
P.O. Wahnaghat, Teh. Kandaghat, Distt. Solan - 173234 (H.P.) INDIA

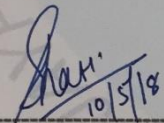
Website: www.juit.ac.in

Phone No. (91) 01792-257999

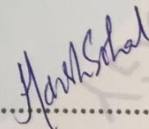
Fax: +91-01792-245362

CERTIFICATE

This is to certify that the work reported in the M.Tech project report entitled “**High Speed Radix-2 Butterfly Structure using Novel Wallace Multiplier**” which is being submitted by **Garima Thakur** in fulfillment for the award of Masters of Technology in Electronics and Communication Engineering by the Jaypee University of Information Technology, is the record of candidate’s own work carried out by her under our supervision. This work is original and has not been submitted partially or fully anywhere else for any other degree or diploma.


10/5/18

Dr. Shruti Jain
Associate Professor
ECE Department
JUIT
Date: 10-05-18



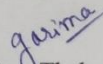
Dr. Harsh Sohal
Assistant Professor
ECE Department
JUIT
Date: 10-05-2018

ACKNOWLEDGEMENT

I might want to offer a profound feeling of gratitude and indebtedness to **Dr. Shruti Jain and Dr. Harsh Sohal**, Department of ECE and supervisor for this work, for their direction, support, inspiration, and consolation all through the finish of this work. Their excitement to listen to the issues, their educative remarks, their suggestions for the convenient and successful completion of this work has been exemplary.

We are appreciative to **Prof. Dr. Samir Dev Gupta**, Director and Head of Department of ECE for his superb support during our work. I might likewise want to thank all of the professors & other supporting members of the department of Electronics and Communication Engineering for their generous help in different courses for the finish of this work. Last however not the slightest, I might genuinely want to thanks, every one of my companions, Ph.D. researchers and friends for their insightful recommendations for fulfilling this undertaking.

Date : 10-05-2018


Garima Thakur

ABSTRACT

In any Central Processing Unit (CPU) the crucial components are Arithmetic and Logic Unit (ALU). ALU can perform different operation like addition, subtraction, multiplication etc. In this thesis addition and multiplication plays an important role because adders and multipliers are the basic building blocks of any Digital Signal Processing applications.

Firstly, in this work adder is used for addition of numbers but it also perform some arithmetic operations. In adders the basic buildings blocks are Half adder and Full adder because they are used for constructing complex adders like Ripple Carry Adder, Carry Select Adder etc. First analyzed efficient adder and used this efficient adder for implementation of modified Carry Increment adder. This modified adder used in many Signal Processing application and increases the speed of the circuit because nowadays delay optimization and power optimization become a very challenging problem with the increase of the portable devices.

Secondly, after adders the crucial component is multiplier. In multipliers the basic block for reducing partial product is adder. The efficient adder used in the multiplier for increasing the speed of circuit and used the circuit in an optimized way. For constructing an efficient multiplier different architecture are proposed so to design an efficient multiplier. A modified multiplier is proposed by using efficient multiplier with efficient adder so, to improve the overall performance of the circuit.

The goal of this is to analyze and compare various adders and multiplication schemes for high-speed and low power operations. Since the various Digital Signal Processing applications, require computationally efficient Multiply and Accumulate operations so the blocks with desired characteristics have to be chosen carefully. Various techniques have been proposed to design multipliers which are efficient in terms of performance, low power consumption and area.

Finally, the efficient adders and multipliers are used in FFT algorithm. FFT is used for signal processing applications. It consists of addition and multiplication operations, whose speed improvement will enhance the accuracy and performance of FFT computation for any applications. FFT are used to covert signal form time domain to frequency domain. In FFT

processing unit butterfly structure is the basic building block and are used for calculating the complex calculation. So, it is important to design an efficient adder and multiplier block and used this efficient block in butterfly structure.

Further work on Low Power Techniques on different multipliers needs to be done in order to make us choose a proper multiplier in accordance with the requirements by making the best possible trade off choice between Speed and Power in different circumstances.

LIST OF FIGURES

Figure 1.1	Logical Circuit of Half Adder.....	2
Figure 1.2	Logical Circuit of Full Adder.....	2
Figure 1.3	Number of transistors on IC (1971-2016).....	4
Figure 1.4	Power density trend versus power design requirements.....	4
Figure 1.5	Design Flow using Verilog	6
Figure 3.1	RTL Schematic of Half Adder	13
Figure 3.2	RTL Schematic of Full Adder	14
Figure 3.3	4 bit Ripple carry adder.....	15
Figure 3.4	RTL Schematic of Ripple carry adder.....	15
Figure 3.5	4-bit Carry select adder	16
Figure 3.6	RTL Schematic of Carry select adder.....	16
Figure 3.7	4-bit Carry skip adder	17
Figure 3.8	RTL Schematic of Carry skip adder.....	18
Figure 3.9	4-bit Weinberger-Smith CLA	20
Figure 3.10	RTL Schematic of Carry Lookahead adder.....	21
Figure 3.11	Block level diagram of a prefix adder.....	22
Figure 3.12	4-bit Kogge-Stone prefix adder	22
Figure 3.13	RTL Schematic of Kogge stone adder.....	23
Figure 3.14	Representation of each KSA block.....	24

Figure 3.15	Block diagram of CIA_RCA.....	27
Figure 3.16	Block diagram of CIA_KSA.....	28
Figure 3.17	RTL Schematic of CIA_RCA.....	29
Figure 3.18	RTL Schematic of CIA_KSA.....	30
Figure 4.1	Block diagram of Multiplier architecture.....	32
Figure 4.2	Partial product array for an $M \times N$ multiplier.....	34
Figure 4.3	4×4 Array multiplier.....	34
Figure 4.4	RTL Schematic of 4 bit Array multiplier.....	35
Figure 4.5	8×8 partial product tree reduction of Wallace multiplier.....	36
Figure 4.6	RTL Schematic of 4 bit Wallace multiplier.....	36
Figure 4.7	RTL Schematic of 4 bit Vedic multiplier.....	40
Figure 4.8	8×8 Array multiplier architecture.....	42
Figure 4.9	8×8 Vedic multiplier architecture.....	43
Figure 4.10	8×8 Wallace multiplier architecture.....	44
Figure 5.1	RTL Schematic of 16 bit KSA.....	48
Figure 5.2	RTL Schematic of 32 bit KSA.....	49
Figure 5.3	RTL Schematic of 16 bit CLA.....	49
Figure 5.4	RTL Schematic of 32 bit CLA.....	50
Figure 5.5	16×16 Wallace multiplier Architecture using KSA.....	53
Figure 5.6	16×16 Wallace multiplier Architecture using CLA.....	53
Figure 5.7	32×32 Wallace multiplier Architecture using KSA.....	55

Figure 5.8 32×32 Wallace multiplier Architecture using CLA.....56

Figure 5.9 Butterfly Structure.....59

Figure 5.10 Flow diagram of Butterfly.....61

Figure 5.11 RTL Schematic of 4 bit butterfly.....62

Figure 5.12 RTL Schematic of 8 bit butterfly.....63

Figure 5.13 RTL Schematic of 16 bit butterfly.....64

Figure 5.14 RTL Schematic of 32 bit butterfly.....65

LIST OF TABLES

Table 3.1 Implementation Table of Full Adder.....	14
Table 3.2 Truth Table of Full Adder.....	19
Table 3.3 Area, Delay and Power calculation of 4-bit Adders.....	25
Table 3.4 Area, Delay and Power calculation of 8-bit CIA.....	28
Table 3.5 Comparison Table of Area, Delay and Power calculation of 8-bit CIA.....	31
Table 4.1 Area, Delay and Power calculation of 4-bit Multipliers.....	41
Table 4.2 Area, Delay and Power calculation of 8-bit Multipliers.....	45
Table 4.3 Comparison Table of Area, Delay and Power calculation of 8-bit Wallace multiplier...	45
Table 4.4 Comparison Table of Area, Delay and Power calculation of 8-bit Vedic multiplier.....	46
Table 4.5 Comparison Table of Area, Delay and Power calculation of 8-bit Array multiplier.....	46
Table 5.1 Area, Delay and Power calculation of 16 bit Adder.....	51
Table 5.2 Area, Delay and Power calculation of 32 bit Adder.....	51
Table 5.3 Area, Delay and Power calculation of 16 bit Wallace multiplier.....	54
Table 5.4 Area, Delay and Power calculation of 32 bit Wallace multiplier.....	56
Table 5.5 Area, Delay and Power calculation of Butterfly structure.....	60

ABBREVIATIONS

VLSI – Very Large Scale Integration

HDL – Hardware Description Language

VHDL – Very High Speed Integrated Circuit Hardware Description Language

ICs – Integrated Circuits

RCA – Ripple Carry Adder

CSA – Carry Save Adder

SQRT – Square Root

CSLA – Carry Select Adder

BEC – Binary To Excess Code

XOR – Exclusive OR

CBL – Common Boolean Logic

KSA – Kogge Stone Adder

DFR – Design Of Reconfigurability

CLA – Carry Look Ahead Adder

DSP – Digital Signal Processor

HA – Half Adder

FA – Full Adder

CSkA – Carry Skip Adder

LUT – Look-Up Tables

CLBs – Configurable Logic Block

SRAM – Static Random Access Memory

RAM – Random Access Memory

FPGA – Field Programmable Gate Array

IOB – Input/Output Block

CIA – Carry Increment Adder

WAP – Write a Programme

DFT- Discrete Fourier Transform

IDFT- Inverse Discrete Fourier Transform

FFT- Fast Fourier Transform

CHAPTER 1

INTRODUCTION

Fast Fourier Transform (FFT) plays a crucial role in many Digital Signal Processing (DSP) applications. In communication systems like Orthogonal Frequency Division Multiplexing (OFDM), FFT is the most important block. It is used to convert a signal from time domain to frequency domain. In various systems there are requirements of high performance FFT to fulfill the demands of next generation with low cost and high speed. There is a need to design an efficient butterfly because it plays an important role in FFT processor. Nowadays, delay and power optimization have become a very challenging problem and portable electronic products are of great demand which need more backup, less area and less weight. So, low power circuit is designed because it directly affects the performance of the circuit. In general purpose processors the most important arithmetic units are adders and multipliers. The emphasis of our work is on minimizing the latency, with the goal being the implementation of the fastest multiplication blocks as possible. When we use digital system on a VLSI chip, much better Signal Processing Systems are implemented with the growth in the scale of integration. A large amount of energy and computation capacity is consumed in the signal processing. Architecture of arithmetic units is chosen carefully to reduce the power consumption and area. Arithmetic unit consists of Adders and Multipliers which are as follows:

1.1 Adders

In digital IC designs the most frequent and essential operation is addition. Addition is done by adders and adders are the most important arithmetic units of any Digital Processing System. The most commonly used blocks in adders are Half Adder (HA) and Full Adder (FA) as shown in Figure 1.1 and 1.2. By using HA and FA many complex adder architectures are constructed. In this report many complex adders are implemented in terms of speed and power. The adder which has high speed is used for implementation of proposed design of adder. Adders are the basic building blocks of many applications. Adders are also used for calculate addresses, table indices and many more.

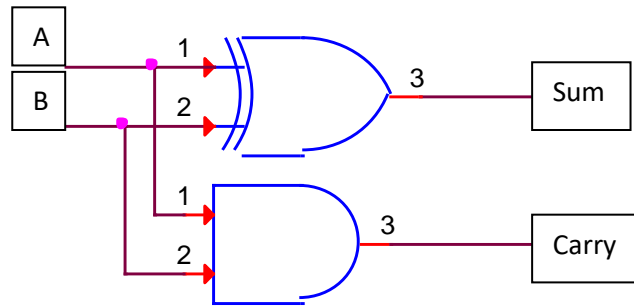


Figure 1.1: Logical circuit of Half Adder

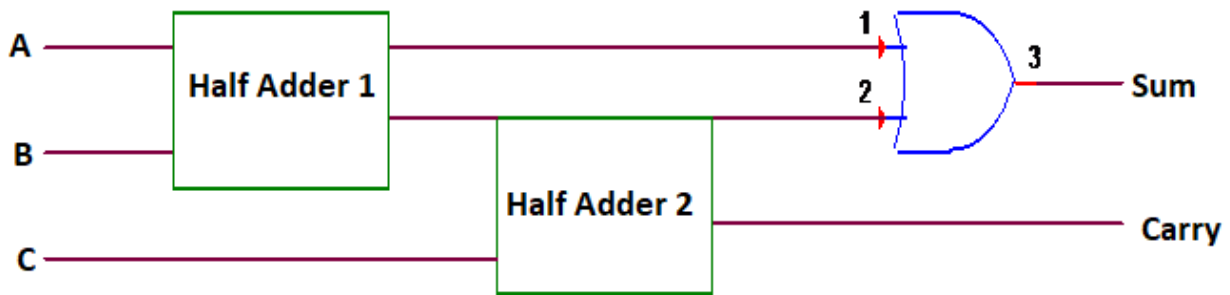


Figure 1.2: Logical circuit of Full Adder

1.2 Multipliers

Multipliers also play a crucial role in the implementation of high performance circuits. These high performance circuits are used in many Digital Signal Processing applications, so there is a need to design an efficient multiplier to meet the requirement of the designer. For designing an efficient multiplier various characteristics are of taken care like: Speed-At high speed multiplier should perform operation, Accuracy- Correct result should given by good multiplier, Area- Less number of LUTs and Slices are occupied by multiplier and Power- The power consumed by the multiplier is less. Adders are used for addition of partial products of efficient multipliers. Three steps are followed for multiplication process: firstly, generation of partial products, second addition of partial products and finally, final addition. In this work the proposed multiplier is implemented using the adder with high speed and low power so to design an efficient multiplier. This efficient multiplier is used in many applications such as Image Processing, DSP, microprocessors etc.

1.3 FFT

In Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IDFT) the complex numbers of arithmetic operations like addition and multiplication are required. For implementation of DFT one of the most efficient ways is FFT because it reduces the usage of arithmetic units. Arithmetic operations of DFT has $O(N \times N)$ order and FFT has $O(N \log N)$ order. Basically, FFT is used to convert signal to frequency domain from time domain. Nowadays, in Wireless Communication researchers employed FFT processor so, to achieve low power, high speed and low area utilization. For N-point data sequence the DFT is shown in Equation 1.1.

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \quad (1.1)$$

For N-point data sequence the IDFT is shown in Equation 1.2.

$$X[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \quad (1.2)$$

$$\text{Where } W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$$

In FFT the most important block used is butterfly block because for increase the performance of the FFT processor firstly, it is important to design the high speed and low power consumption butterfly element. Butterfly is implemented using an efficient multiplier and adder. Different adders and multipliers circuits are designed, implemented and compared as explained in Chapter 2 and Chapter 3. Later proposed adders and multipliers are compared with the existing work.

1.4 Need for Low Power Design

During the last decades there is an increase in the integrated circuits which is predicted by Moore's Law. Figure 1.3 shows that every two year the number of transistors in IC doubles. As the feature size decreases there is reduction of power consumption. But as shown in Figure 1.3 power consumption increases with increase in the transistor count. Low power consumption in portable application is the major constraint. Power consumption and energy efficiency contribute to so many factors like longer operation time, higher workload etc in these situation power consumption become more critical. Reliability is also reduced as increase in on chip temperature due to increase in power consumption. Many problems are raised as power supply is delivered to chip such as noise immunity, power rails design etc. In many high performance applications the cost of cooling and packaging is increasing, so, power reduction is the primary objective of the designers in these applications.

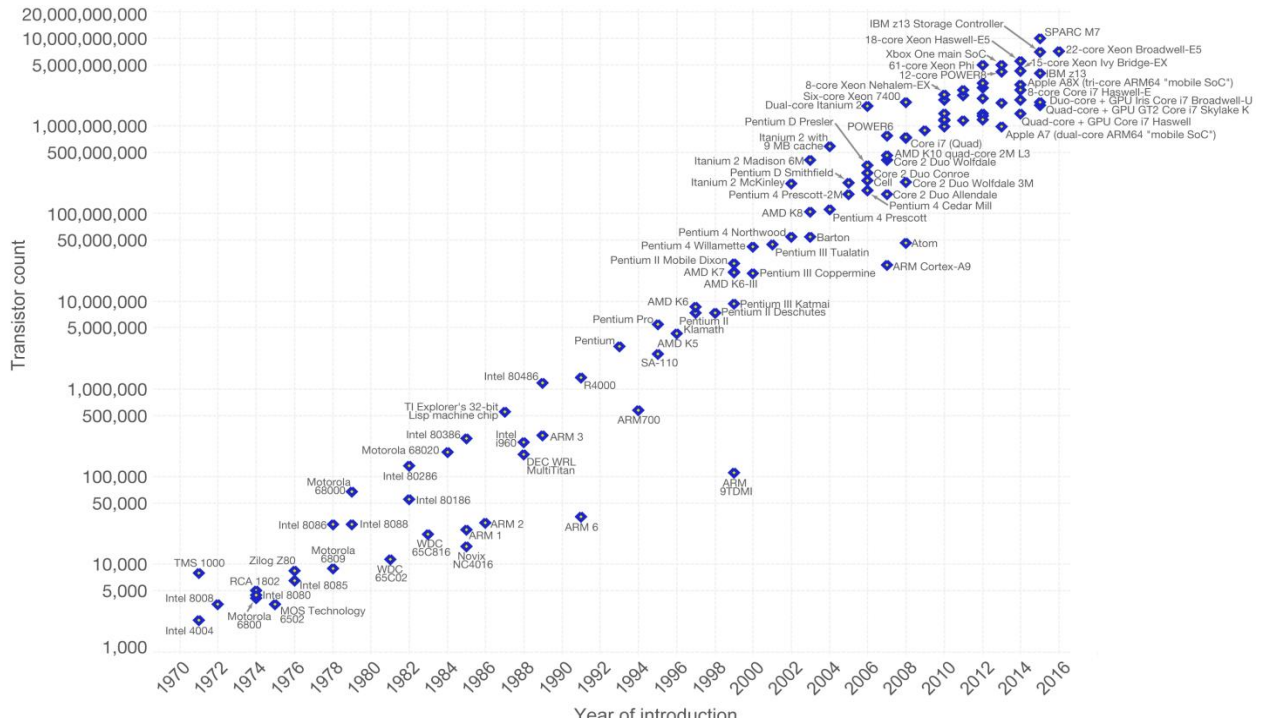


Figure 1.3: Number of transistors on IC (1971-2016)[35]

Therefore, it is important to reduce power technique for current and future integrated circuit. Figure 1.4 shows the graph between power density and power density requirement for modern SoCs. The widening gap is shown in Figure 1.4 which shows the huge challenge for the designers that they face today. Lots of efforts are required for managing power in the design and the effort required is increasing with the new designs to meet new challenges.

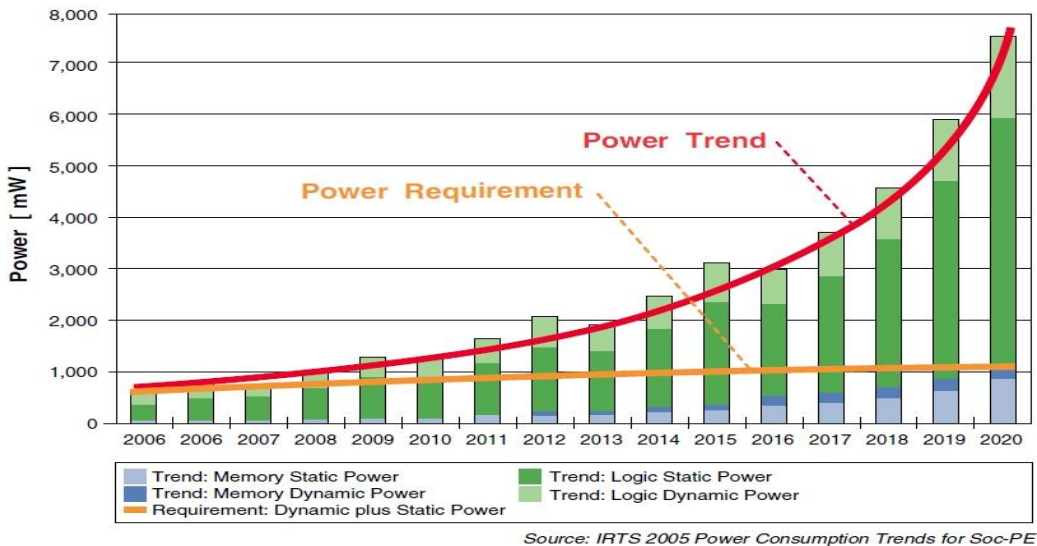


Figure 1.4: Power density trend versus power design requirements[36]

As shown in Figure 1.3 increasing level of device integration and the complexity of the circuits is also increases. So, the primary goal is reduction of the power consumption. As the years goes by, the power dissipation also goes on increasing linearly and because of ever-shrinking size of chips the power density increases exponentially. If the rise of power density exponential then few years later a microprocessor designed who has power equal to nuclear reactor. So, this introduces reliability concerns as rise in power density such as thermal stresses, device degradation induced in hot carrier and it will result in loss of performance. As the increase in the demand of portable devices which is powered by batteries need low power chips this is the another factor.

1.4 Programming Language

Hardware Description Language (HDL) e.g. Verilog, VHDL are used to Write a Program (WAP). We have used Verilog programming for implementation of our digital circuit, because by using a HDL we can describe any digital hardware at any level. For design and verification of digital circuits it is most commonly used at the Register-Transfer Level (RTL) of abstraction Verilog language have many advantages over VHDL i.e. compact language, reduction operator etc. The Figure 1.5 summarizes the high level design flow for an ASIC (i.e. gate array, standard cell) or FPGA. Various steps are used for simulation and these steps are further divided into various other parts:

a) System-Level Verification

In system level verification there is complete verification of system model and simulates aspects. There is a detailed description of the system functionality and alternatively, maybe there is partial description of the system properties. For system-level modeling Verilog is not suited ideally.

b) RTL Design and Test Bench Creation

In RTL design the architecture of the system is stable and it's partitioning. In Verilog the RTL and test cases start capturing. Both the tasks are complementary and for various designs different RTL and test cases captured. If automated synthesis of logic is used then the RTL Verilog should be synthesizable. A disciplined approach is followed by test case.

c) RTL verification

In comparison to gate level simulation the RTL magnitude simulation is faster of one or two order and this shows that by spending more time in simulation it speed-up is best when more simulation is done. Against the specification the validation of functionality is done by RTL Verilog.

d) Look-Ahead Synthesis

In this synthesis design process will be done early, so the designers evaluate speed and area accurately and researchers check how the Verilog synthesis will be done, until the completion of functional simulation the production of main synthesis run. If the validation of design functionality is not done it is pointless to invest efforts and time in synthesis.

e) System Level

Verilog is addressed by system Verilog because it is not suited ideally for simulation at system level. VHDL is user-defined types in which designer allowed to work in the domain of problem. But Verilog is a pre-defined type in which designer allowed to work for scholastic simulation and build-in language features allowed for queuing, modeling performance and throughput.

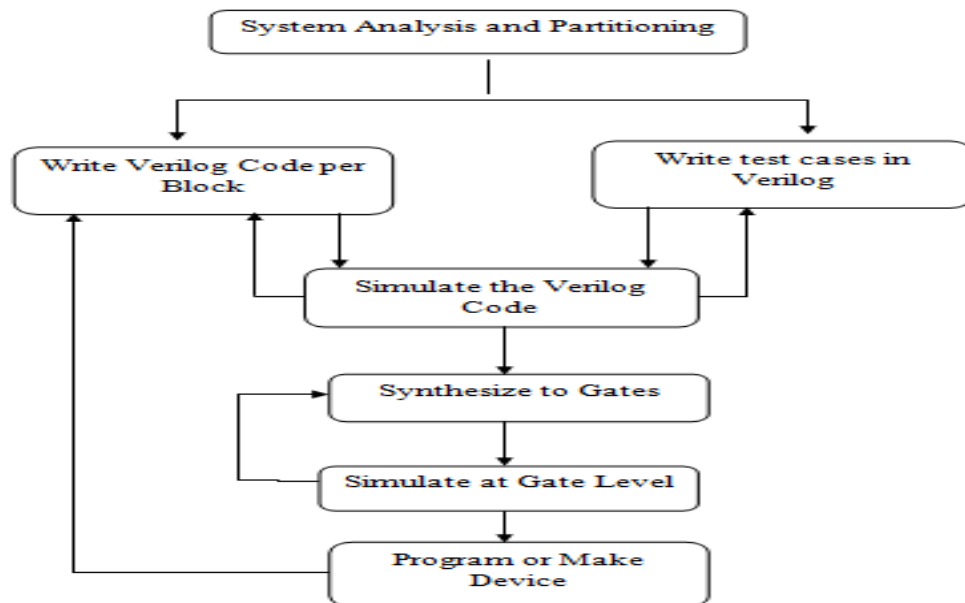


Figure 1.5: Design Flow using Verilog

1.3 Research Approach

The basic idea behind our work is to design as efficient Butterfly Structure having low power and less delay because in FFT processor the basic block is Butterfly. We need to design efficient Butterfly block so it will be the processor for fast computation in many applications. In Butterfly the important blocks are adders and multipliers. An efficient multiplier block is implemented in which the basic building block is adder. Nowadays, power optimization and delay optimization have become a very challenging problem. Firstly, our focus is on designing and implementation of adder in which we have examined different adders. The best adder is one with a minimum delay, low power consumption and finds a proper relation between LUTs, slices, power and delay. Secondly, the best high speed adder is used for designing an efficient multiplier. Different multipliers are studied, designed and implemented using Verilog HDL. After getting all the results we find out the best multiplier with high speed and low power consumption. Finally, this efficient multiplier and high speed adder is used for the implementation of Butterfly Structure. Our future work is dedicated for reduction of more power consumption by reducing the number of logic gates.

1.4 Organization

CHAPTER 2: LITERATURE SURVEY – This chapter explains the various types of algorithms used in adders and multipliers to design optimized circuits. The algorithm which is best suited to design high performance circuits is used.

CHAPTER 3: ADDERS – This chapter explains the different types of adders and their implementation. An optimized adder is used to design the high speed circuit.

CHAPTER 4: MULTIPLIERS – This chapter explains different types of multipliers their implementation and results. An optimized adder and multiplier combine to construct high speed circuits.

CHAPTER 5: FFT – This chapter explains the implementation of butterfly using efficient multipliers and adders. Butterfly plays a crucial role in the processors.

1.5 Tools Used

Simulation Software:

- Xilinx 14.1 ISE design suite

Power Calculation

- XPower Analyzer

CHAPTER 2

LITERATURE SURVEY

We have studied a lot of papers some are listed :

Akhter S. *et al.* 2017[1] in this paper various digital adders are used for comparative analysis of Vedic multiplier. Using CBL adder the 8-bit Vedic multiplier is 20% faster than BEC and is approximately 5% faster in terms of delay than RCA-CSA, SQRT-CSA and RCA. With the increase in the width size they have calculated outcome in terms of speed, area and leakage power.

Gowreesrinivas K. V. *et al.* 2016[2] in this paper using different types adders and by incorporating Vedic multiplier, a new type of multiplier is developed. The new developed multiplier i.e. Single Precision Floating Point (SPFP) have drawback of optimization of speed and area. By reducing interconnections and complexity the overall performance can be improved. It is observed that using combination of prefix sklansky adder and Vedic multiplier (VM) is best in comparison to other multipliers because it is best in terms of speed and complexity.

Gokhale G. R. *et al.* 2015[3] V M is implemented in this paper by means of lesser number gates, area, which is required by proposed CSLA. The Booth multiplier has more area, low speed compare to proposed V M, so it is superior. In the architecture of Vedic multiplier the addition block plays a important role for increasing and decreasing the performance of the circuit.

Murugeswari S. *et al.* 2014[4] in this a low power and an area efficient modified Wallace and truncated multiplier is implemented by using full adder which is based on mux. In the end it is concluded that reduction in area of modified truncated multiplier shows improvement in device utilization compared to modified Wallace multiplier.

Anjana R. *et al.* 2014[5] in this paper they designed a modified novel high speed multiplier as a result of combining Kogge stone adder with the multiplier to design the fastest multiplier.

S. Rajaram *et al.* 2011[6] in this paper proposed multiplier has less delay than the conventional multiplier. Proposed multiplier is Wallace multiplier which used Parallel prefix adder at the final stage, so there is improvement in multiplier.

R. B. S. Kesava *et al.* 2016[7] in this paper a simple approach is proposed for Wallace multiplier(WM) using Carry select adder(CSLA) , so to reduce area. They implement CSLA with BEC in Wallace tree multiplier to occupying less power, less area and memory when compared to WM using CSLA and WM.

S.Srikanth *et al.* 2016 [8] in this paper by using multiplexers and XOR gate, proposed a full adder. In Wallace tree multiplier , by putting the proposed full adder in the decrease stage an average delay, power and area reduction achieved compared to existing method respectively is achieved.

D. Paradhasaradhi *et al.* 2014[9] an area efficient proposed WM is implemented in this paper by using CBL which is based on SQRT (square root) CSLA. There is reduction of delay and area by reducing the number of gates. Copying of adder cells are removed in the usual CSLA by sharing CBL term.

Gurjar P. *et al.* 2011[10] simulated and synthesized the different adders. The parameters like area and speed and the usefulness of fast adders is analyzed by simulated results. In the end, for 8-bit and 16-bit adders the caught parameters are analyzed. This paper infers that in conditions of delay and area consumption the Carry skip adder the efficient adder.

Bais K. *et al* 2016 [11] mentioned relationship of speed and area of different adders for various number of bits. From the delay comparison of adders, it is clear that Kogge Stone adder (KSA) is the fastest adder because it is parallel prefix adder. In the end they drive a conclusion that the speed and area cannot be optimized at the same time. If one parameter is improved the other

definitely shows degradation. In this paper we understand that KSA was the best when considered for delay but for area, as the number of bits of operands increases KSA occupies more area due to increase in parallel prefix stage.

Nandini M. *et al* 2015[12] discussed different kinds of prefix adders particularly Spanning adder, Sparse Kogge stone adder, Ladner fisher adder, Brent Kung adder and KSA. Correct practicality of every individual module was tested. This paper has resulted in reduced delay and power in the development of adders design. After analysis calculated KSA and Sparse KSA different parameters is being compared with the other adder. After the comparison less combinational delay of Kogge Stone adder with 12.499ns and less amount of power consumption of Ladner fisher adder with 0.26089 mW. In future using parallel prefix adders all the proposed architectures are designed.

Kulkarni R. *et al* 2015[13], discussed the performance of different adders. Characterization of different adders and implementation on an FPGA is done. Later than observe the outcome of comparisons, for two 8 bit addition numbers, CLA is superior. For three and four 8 bit numbers addition Carry save adder(CSA) with last stage built by RCA is preferable. In future work, low area as well as delay is required to design unique adder and to meet the requirement of current industry.

Mitre A. *et al* 2015[14], this paper compared completely different addition rule for various performance parameters i.e. power, area and speed for different adders such as Ripple carry adder(RCA), CSA, Carry select adder, Carry look ahead adder(CLA) and KSA. By merging Kogge stone and Carry select algorithms a high speed adder is then designed and works significantly faster than the rest.

Kumar A. *et al* 2013[15] explained that Ripple carry adder design is basic and it is appropriate for just addition of less width operand since delay run straightly with the width of operand. Linear area required by Carry skip adder which is not really bigger than area required by the RCA . As compare to other the delay of CLA is less and much faster than RCA. For high speed

multiplication and accumulation, they can use Carry-look ahead adder(CLA) for 32 bit multiplier-accumulator unit. In fact the speed of multiplier has approximately twice the speed with carry-look ahead adder.

SaiKumar M. *et al* 2013[16], in this paper, presentation parameters of adders such as delay, area are compare, determined and design of various adders are discussed. Better performance in terms of area and delay is achieved through Carry increment adder (CIA) in comparison to additional adder topologies. For the later use, design of unique adder is needed which will provide small area, speed and meet up the requirements of current VLSI industry.

Suba C. *et al* 2014[17], design for reconfigurability (DFR) technique is presented by this paper for CLA.DFR proposal which has planned to isolate an extensive CLA keen on different little part ones. A small amount of area and delay penalty is incurred by the DFR scheme. The CLA has the smallest amount delay-area result. It is proper for conditions where together low power and fastness application. It is not feasible to apply CLA in steady delay for the wider-bit adders as there will be low speed, larger power consumption and substantial loading capacitance.

Singh A. K. *et al* 2017[28], in this paper, for higher radix FFT an efficient algorithm of butterfly unit was implemented. Taking various issues of FFT implementation an capable Butterfly block is implemented firstly and used this Butterfly block in FFT. Vedic multiplication result analyses in terms of slices and LUTs and implement an efficient butterfly in comparison to Booth multiplication (BM) technique. The design using CLA require large amount of hardware but faster than using RCA.

Rashmi M. J. *et al* 2014[29], in this paper, two algorithms are implemented which was based on DIT-FFT. First, algorithm was on the bases of Radix-2 FFT and second on the bases of Split Radix FFT. Both algorithms are compared for device utilization, speed and taking width size of 16 bits. Large amount of memory usage and increase in the delay as increases the single length of the proposed architecture using radix-2 FFT. This drawback was overcome by Split Radix FFT algorithm.

CHAPTER 3

ADDERS

In digital IC designs the most frequent and essential operation is addition. Addition is done by adders and adders are the most important arithmetic units of any Digital Processing System. The most commonly used blocks in adders are HA and FA. By using HA and FA many complex adder architecture are implemented.

3.1 Basic Adder

There are two types of basic adder which are used i.e. Half Adder and Full adder. These adders are the most important building block of any circuit.

3.1.1 Half Adder (HA) : HA is used for addition of two 1 bit number, which give sum bit(s) and carry bit (c). Sum bit is given by XORing A and B, Carry bit is given by ANDing A and B which is expressed by Equation 3.1 and 3.2 respectively.

$$\text{Sum}(s) = A \text{ XOR } B \tag{3.1}$$

$$\text{Carry}(c) = A \text{ AND } B \tag{3.2}$$

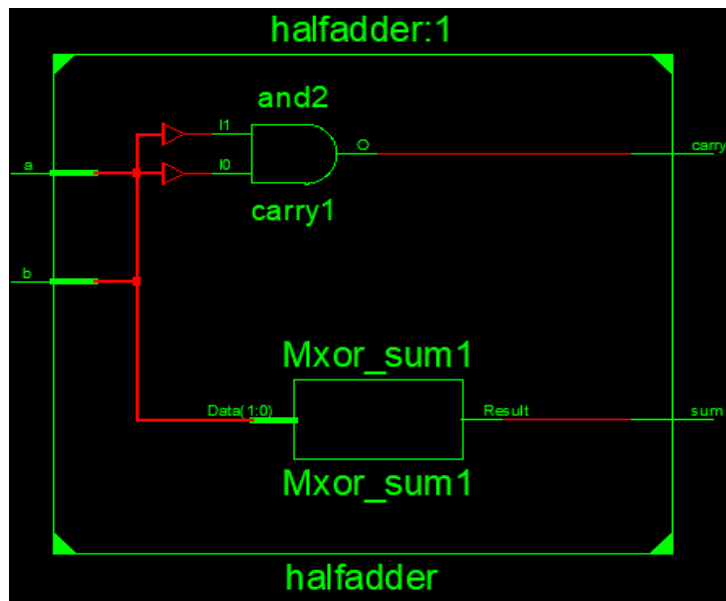


Figure 3.1: RTL Schematic of Half Adder

3.1.2 Full adder (FA) : FA adds three 1 bit number, which produces sum bit(s) and carry (c). By cascading two half adder it also produce sum and carry bit. Sum and Carry bit is expressed by Equation 3.3 and 3.4 respectively.

$$\text{Sum}(s) = A \text{ XOR } B \text{ XOR } C_{in} \tag{3.3}$$

$$\text{Carry}(c) = (A \text{ AND } B) \text{ OR } (B \text{ AND } C_{in}) \text{ OR } (A \text{ AND } C_{in}) \tag{3.4}$$

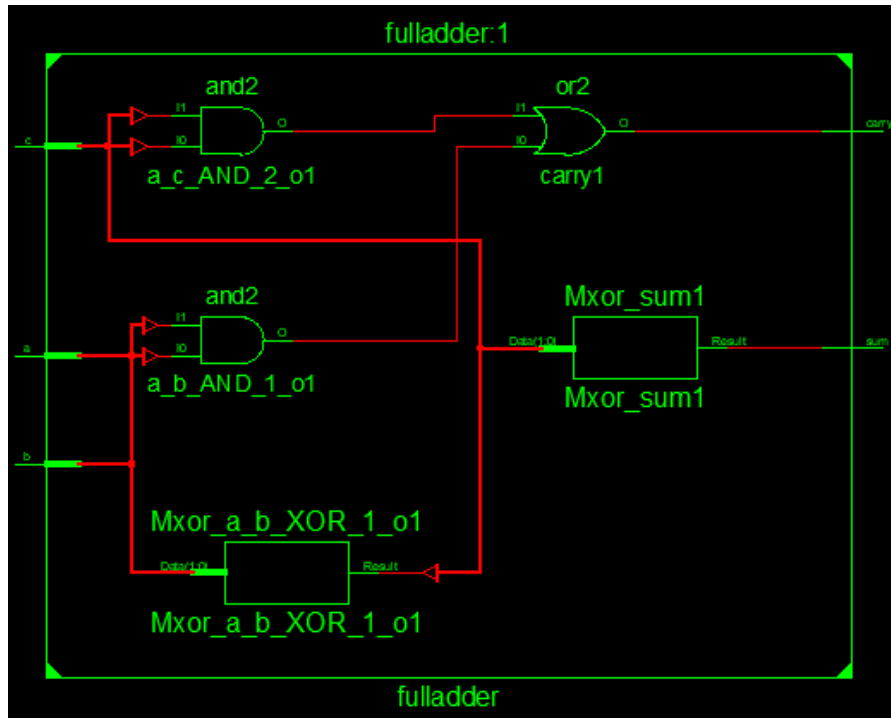


Figure 3.2: RTL Schematic of Full Adder

Table 3.1: Implementation Table of Full Adder

Inputs			Half Adder1 O/P		Half Adder1 O/P		Final Output	
A	B	C _{in}	S1	C1	S2	C2	S	C _{out}
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	1	0
0	1	0	1	0	1	0	1	0
0	1	1	1	0	0	1	0	1
1	0	0	1	0	1	0	1	0
1	0	1	1	0	0	1	0	1
1	1	0	0	1	0	0	0	1
1	1	1	0	1	1	0	1	1

3.2 Complex Adder

Complex Adders are constructed by using basic adders i.e. HA and FA. Design circuitry of these adders is complex because they take number of blocks to design a circuit.

3.2.1 Ripple Carry Adder (RCA) : Multi-bit addition is performed by the RCA and moreover the processing delay is also increased by it. RCA is formed by cascading full adders in series. Each FA generates a carry which is provided to next FA and the process goes on, which is shown in Figure 3.3. The delay is increased, when the numbers of bits go on increasing. The advantage of RCA is its easy implementation and simple design.

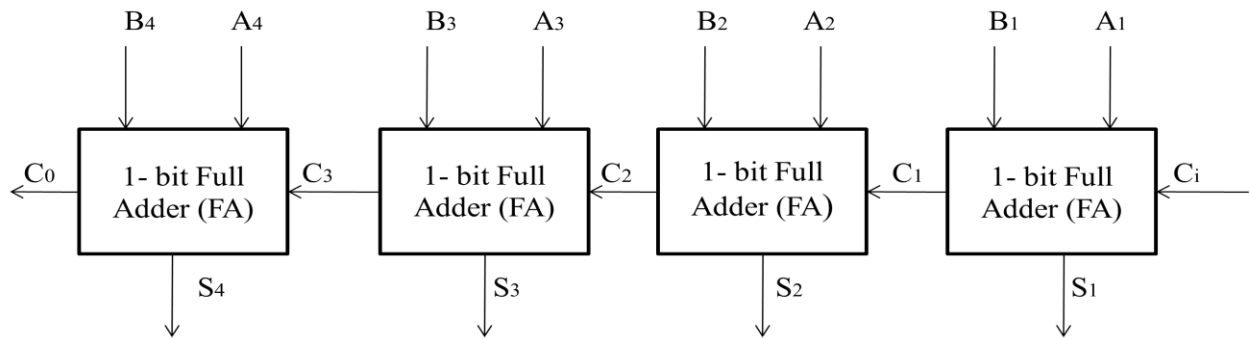


Figure 3.3: 4 bit Ripple Carry Adder

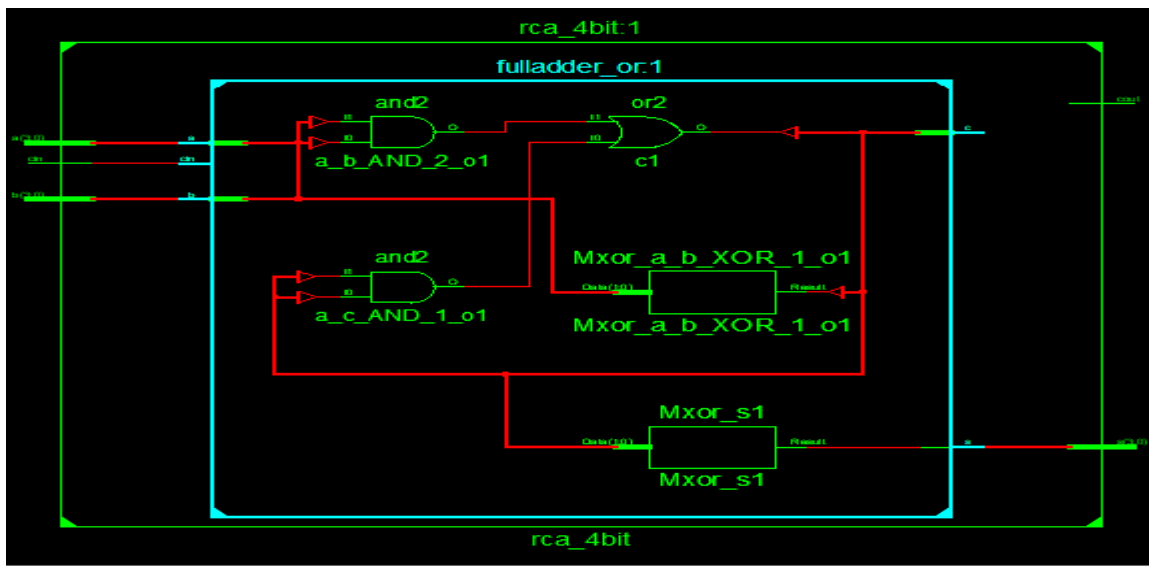


Figure 3.4: RTL Schematic of Ripple Carry Adder

3.2.2 Carry Select Adder (CSLA) : CSLA is formed by two RCA and 2:1 multiplexer. Independent generation of sum and carry i.e. $c_{in}=1$ and $c_{in}=0$ are executed parallelly in CSLA. Correct sum along with correct carry-out is then selected by the multiplexer, depending on real carry-out of previous section. CSLA is further divided into two blocks i.e. uniform and variable block. More hardware is used by CSLA even though it gives less delay compared the ripple carry adder. 4- bit CSLA is shown in Figure 3.5 and RTL Schematic in Figure 3.6..

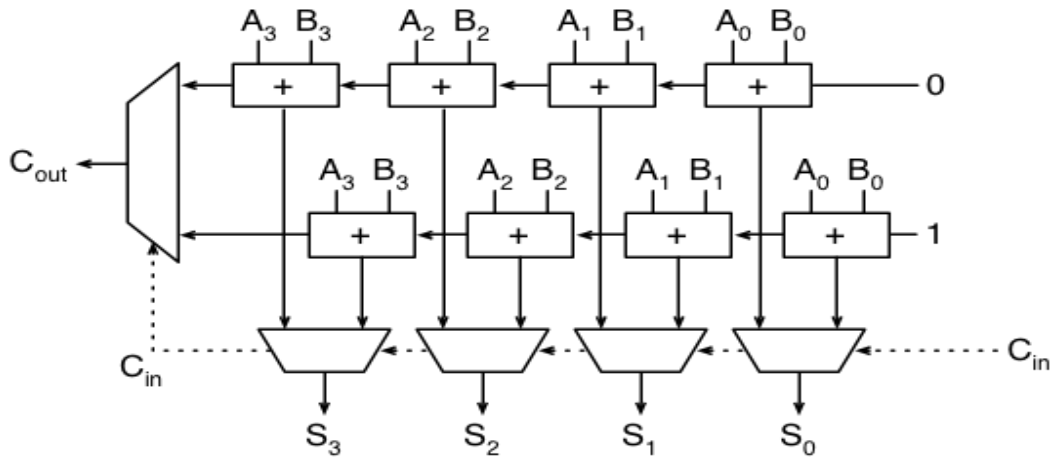


Figure 3.5: 4-bit Carry Select Adder

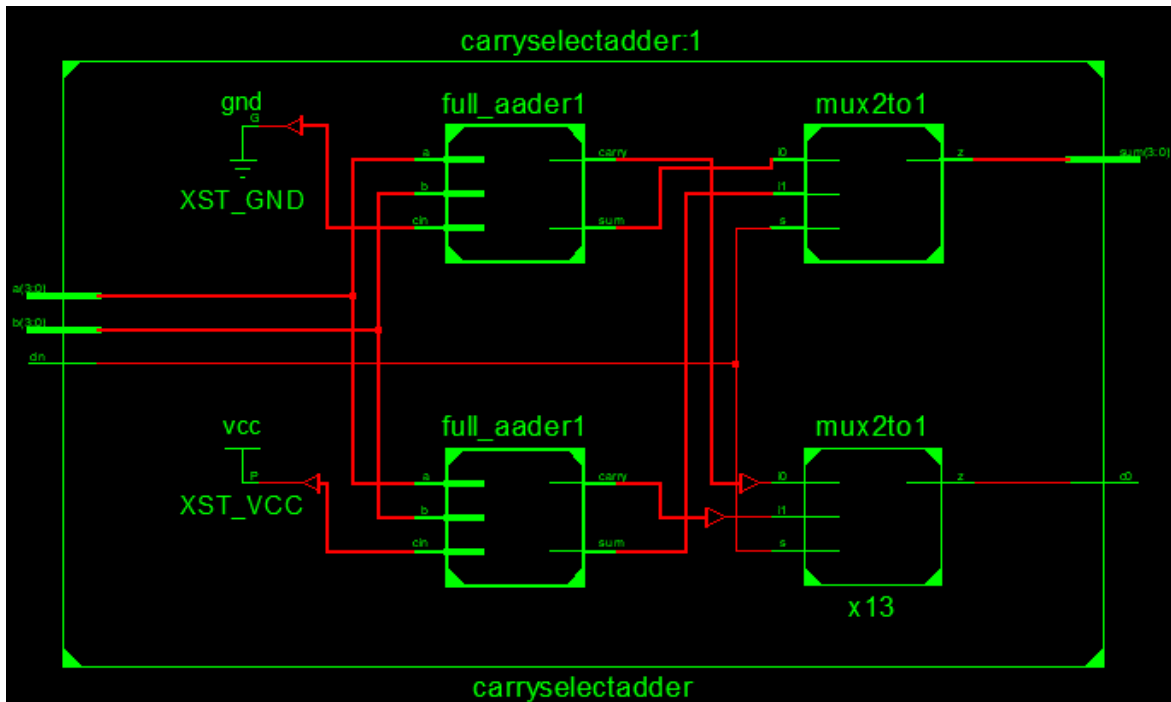


Figure 3.6: RTL Schematic of Carry Select Adder

3.2.3 Carry Skip Adder (CSkA) : This adder enhances delay of RCA with small effort. As the name indicates, CSkA uses skip logic in the propagation of carry. It is designed to speed up a wide adder by adding the propagation of carry bit around a portion of the entire adder. The carry-in bit designated as C_i . The output of RCA (the last stage) is C_{i+4} . The carry skip circuitry consists of two logic AND gate which accepts the carry-in bit and compares it with the group of propagated signals using the individual propagate values.

$$P_{[i,i+3]} = (P_{i+3}) \cdot (P_{i+2}) \cdot (P_{i+1}) \cdot P_i \quad (3.5)$$

Output stage is produced when the output from the AND gate is ORed with C_{out} of RCA. Final carry is expressed by output stage which is represented by Equation 3.6.

$$\text{Carry} = C_{i+4} + (P_{i,i+3}) \cdot C_i \quad (3.6)$$

If $P_{[i,i+3]} = 0$, then the Carry-out of the group is determined by the value of C_{i+4} . However, if $P_{[i,i+3]} = 1$ then the Carry-in bit is $C_i = 1$, then the group carry-in is automatically sent to the next group of adders. The design of schematic of CSkA is shown in Figure 3.7.

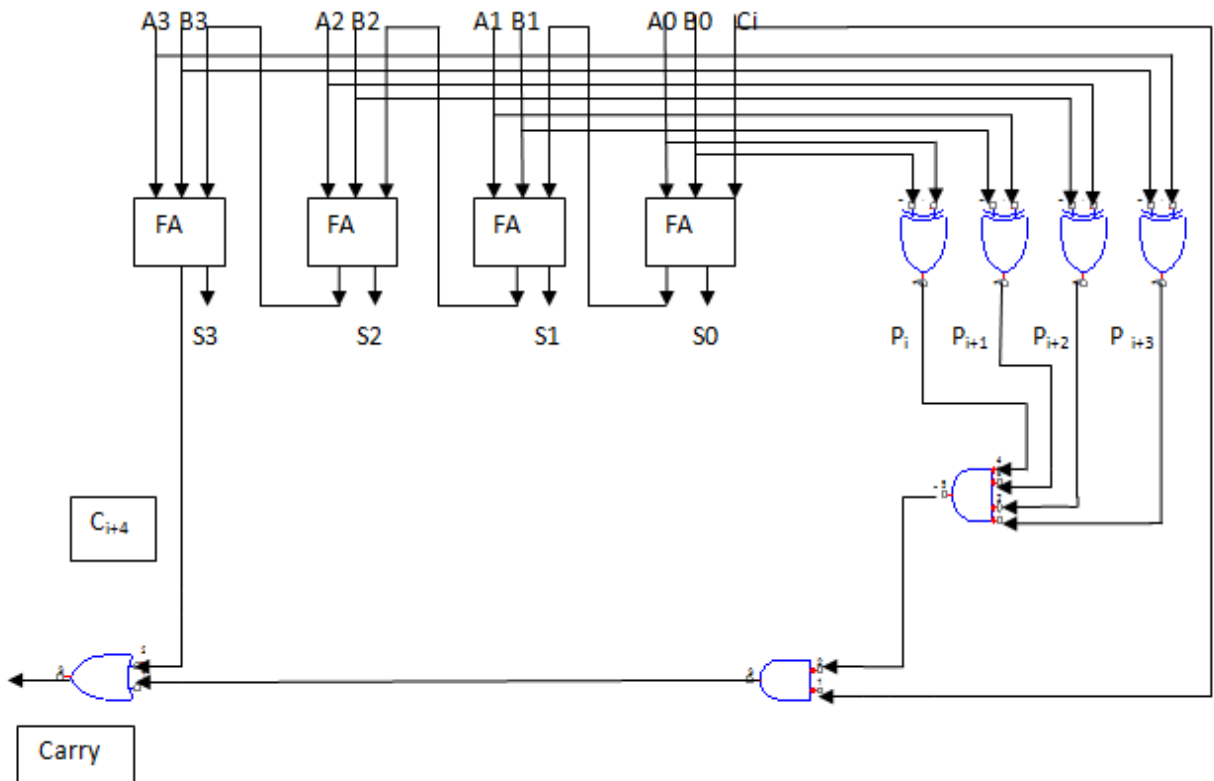


Figure 3.7: 4-bit Carry Skip Adder

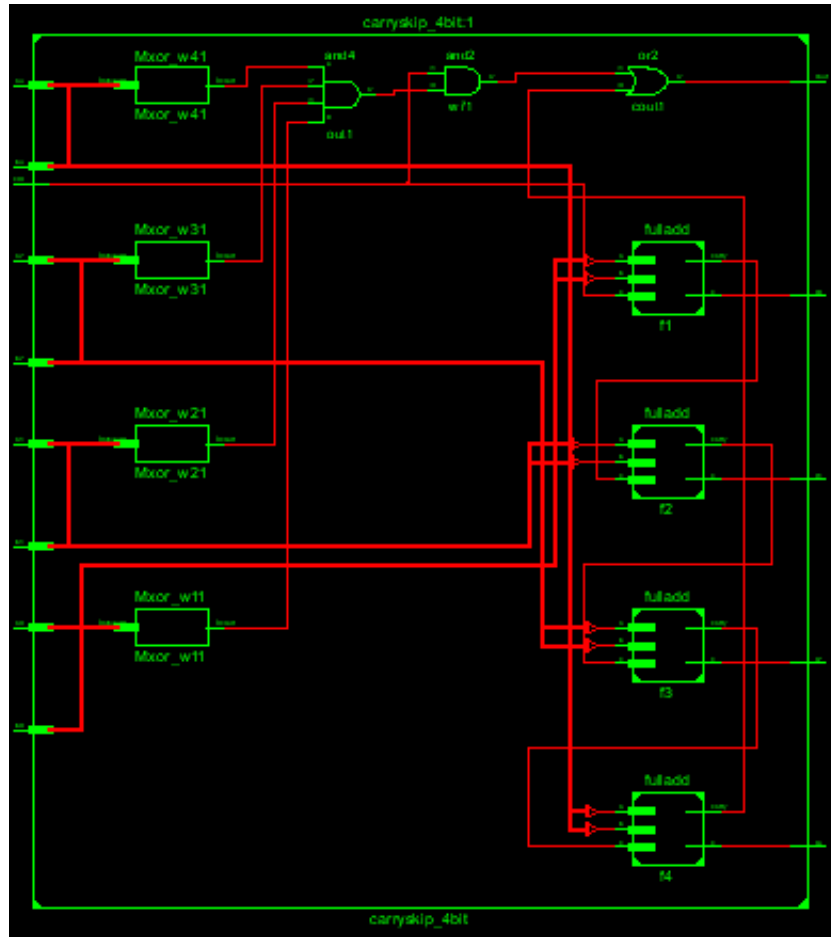


Figure 3.8: RTL Schematic of Carry Skip Adder

3.2.4 Carry Look Ahead Adder (CLA) The RCA is simple and easy to implement, but it suffers from serious delay issues. This is because the next stage of full adder needs to wait for Carry bit from the previous stage full adder. The CLA solves this problem by calculating the carry signals in advance, based on the input signals. CLA technique is to drive the 'Sum' and 'Carry' outputs by using intermediate terms defined as 'Generate (G)' and 'Propagate (P)' terms. In the case of propagate the 'Carry-out' depends on the 'Carry-in' and in the case of generate the 'Carry-out' independent of the 'Carry-in'.

The Table 3.2 illustrates the concept of Propagate and Generate more clearly. The output 'Sum' and 'Carry' of the full adder in terms of P and G, can be observed from Table 3.2 as expressed by Equation 3.7 and Equation 3.8:

$$S_i = P_i \oplus C_i \quad (3.7)$$

$$C_{i+1} = G_i + (P_i \cdot C_i) \quad (3.8)$$

Table 3.2: Truth Table of a Full Adder

	A	B	Cin	Sum	Cout	
G=P=0	0	0	0	0	0	Cout=0 (no dependence on Cin)
	0	0	1	1	0	
	0	1	0	1	0	
P=1	0	1	1	0	1	Cout=Cin
	1	0	0	1	0	
	1	0	1	0	1	
G=1	1	1	0	0	1	Cout=1 (no dependence on Cin)
	1	1	1	1	1	

Weinberger and Smith proposed a method for fast carry generation which states that the carry need not depend on the previous carry which is shown in Figure 3.9. Generate term produces a carry-out independent of the carry-in, i.e. no matter what the carry-in, the carry-out is always '1', when both of the inputs A and B are '1' thus $G=A \cdot B$. The Propagate term transfers the input Carry as output Carry when only one of the inputs is high. The carry generation is done by first calculating Generate (g_i) and Propagate (p_i) which is explained by Equation 3.9 and Equation 3.10 respectively.

$$g_i = A_i \cdot B_i \quad (3.9)$$

$$p_i = A_i \oplus B_i \quad (3.10)$$

Carry is generated by Equation 3.11 as:

$$C_i = g_{i-1} + p_{i-1} C_{i-1} \quad (3.11)$$

After the generation of carry, the sum is calculated using the Equation 3.12.

$$S_i = A_i \oplus B_i \oplus C_i \quad (3.12)$$

For wide adders where $N > 16$ (N is the input operand size), the delay of the carry look-ahead adders becomes dominated by the delay of passing the carry through the look-ahead stages and the implementation need high fan-in gates.

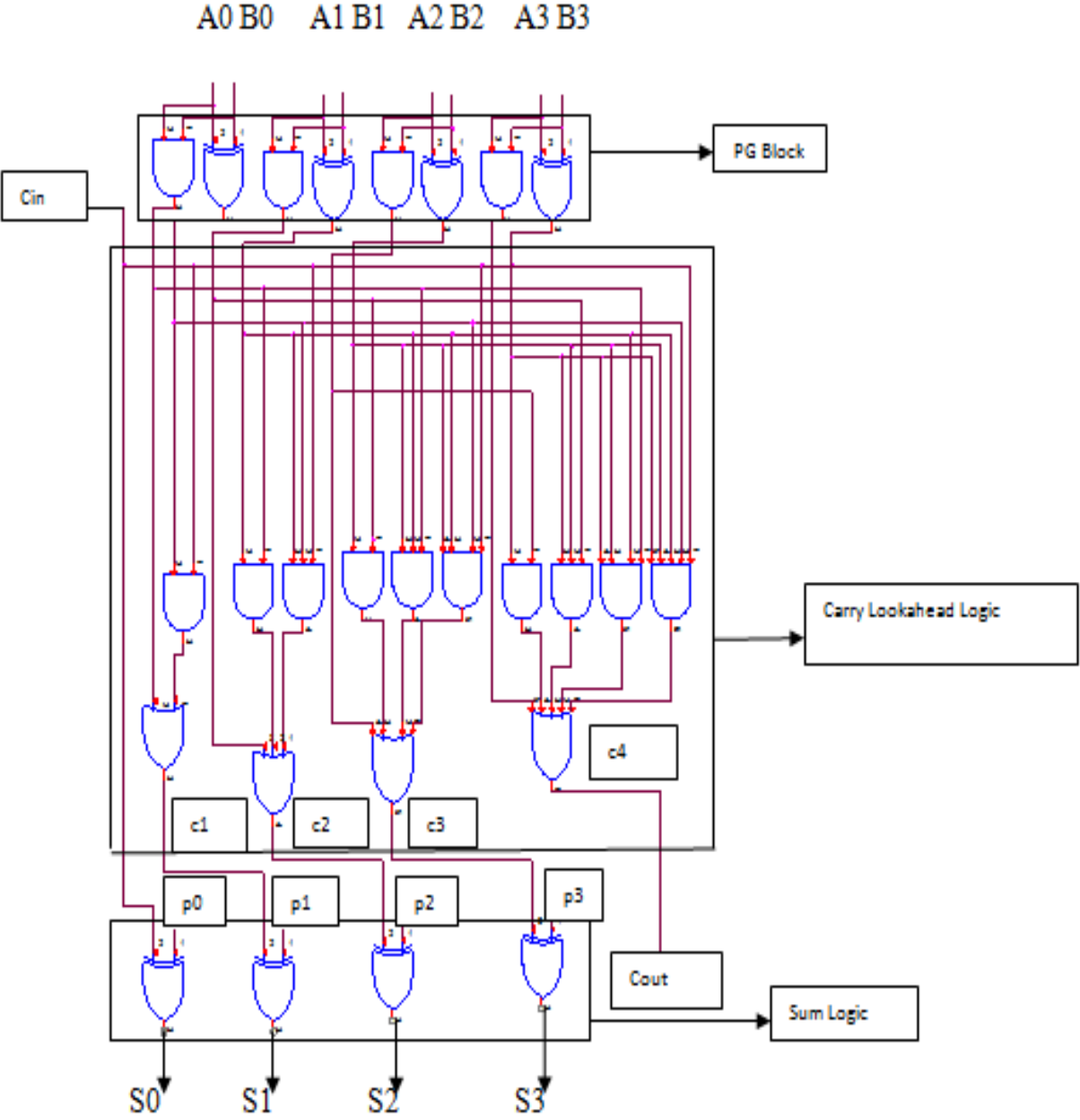


Figure 3.9: 4-bit Weinberger-Smith CLA

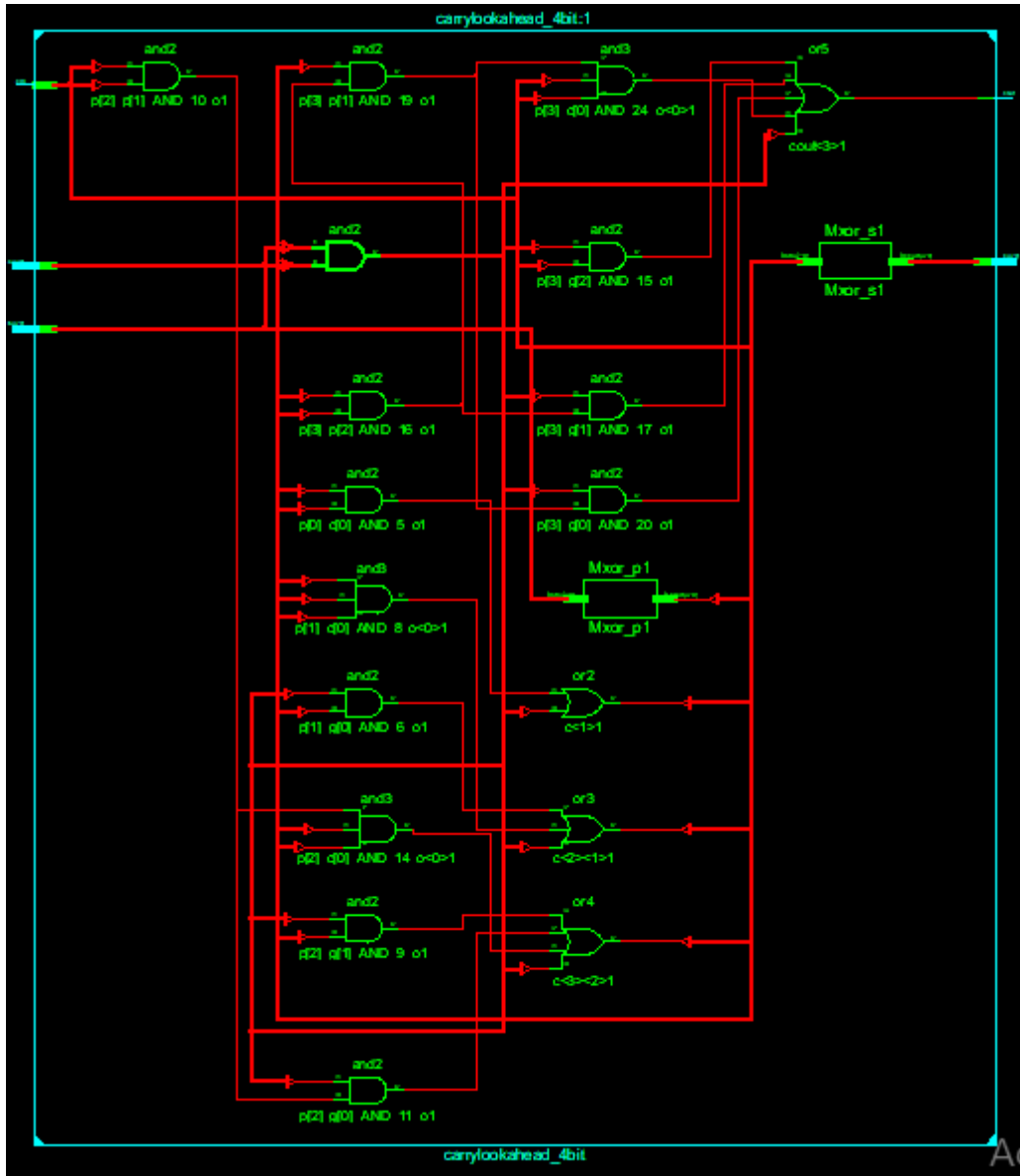


Figure 3.10: RTL Schematic of Carry Lookahead adder

3.2.5 Kogge Stone Adder (KSA) : It is basically a prefix based adder. Prefix adder includes three stages i.e: pre-computation stage, prefix network stage and post-computation stage which is shown in Figure 3.11.

1. *Pre-Computation* -It computes the carry 'Propagate' and carry 'Generate' bits for each input pair as given by Equation 3.13 and Equation 3.14.

$$\text{Generate, } G_i = A_i \text{ AND } B_i \quad (3.13)$$

$$\text{Propagate, } P_i = A_i \text{ XOR } B_i \quad (3.14)$$

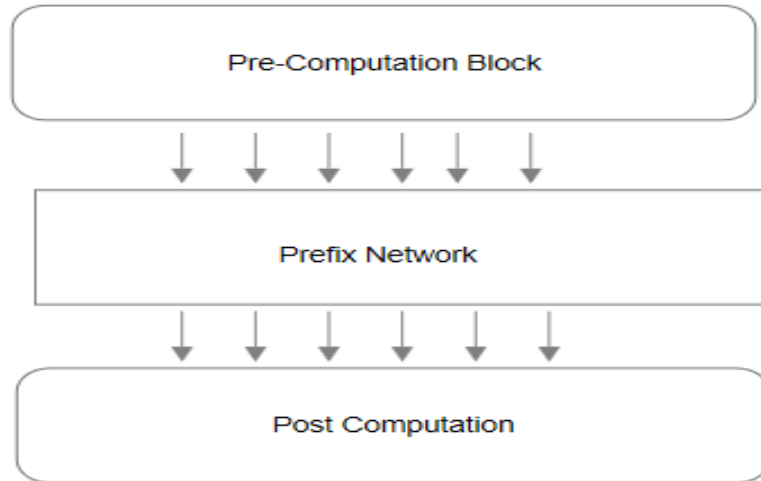


Figure 3.11: Block level diagram of a prefix adder

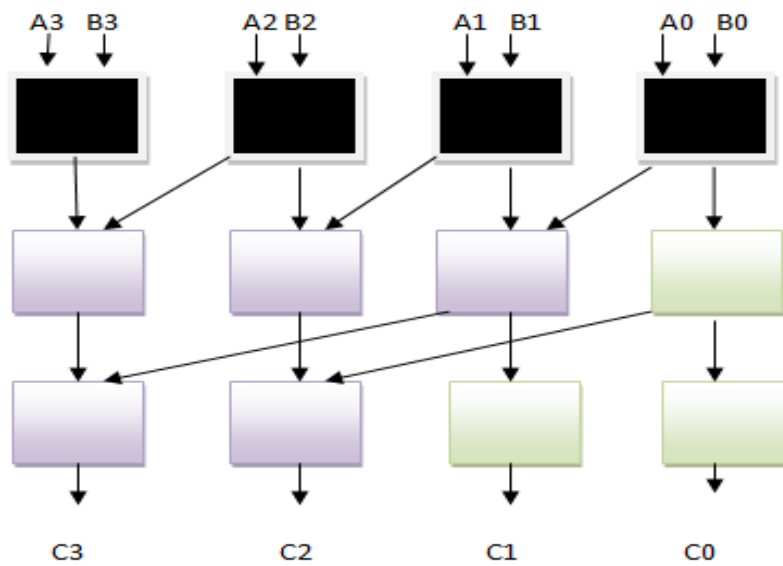


Figure 3.12: 4-bit Kogge-Stone prefix adder

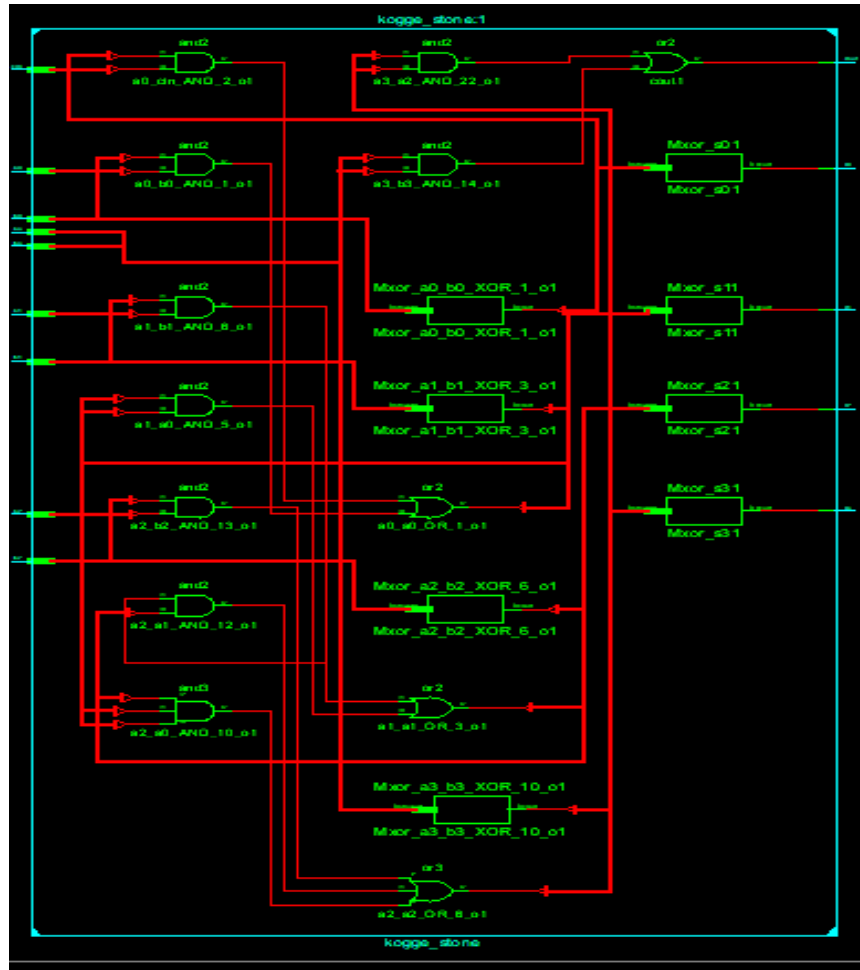


Figure 3.13: RTL Schematic of Kogge stone adder

2. *Prefix Network* - It computes the final carry from the carry 'Propagate' and carry 'Generate' bits which is expressed by Equation 3.15 and Equation 3.16.

$$\text{Propagate, } P = P_i \text{ AND } P_{i\text{prev}} \quad (3.15)$$

$$\text{Generate, } G = (P_i \text{ AND } G_{i\text{prev}}) \text{ OR } G_i \quad (3.16)$$

3. *Post Computation* - It computes the final Sum from carry generated in the prefix network stage. Final sum and final carry is expressed by Equation 3.17 and Equation 3.18 respectively.

$$\text{Sum, } S_i = P_i \text{ XOR } C_{i-1} \quad (3.17)$$

$$\text{Carry, } C_i = G_i \quad (3.18)$$

The KSA is the parallel prefix form that takes more area to implement, but has a lower fan-out at each stage. KSA started being used in multi-bit addition for faster addition but wiring congestion is often a problem. The KSA tree structure of 4 bit has been shown in Figure 3.8.

Figure 3.14 shows the colored representation of Figure 3.12. This figure also explains the equations for calculating Propagate and generates terms.

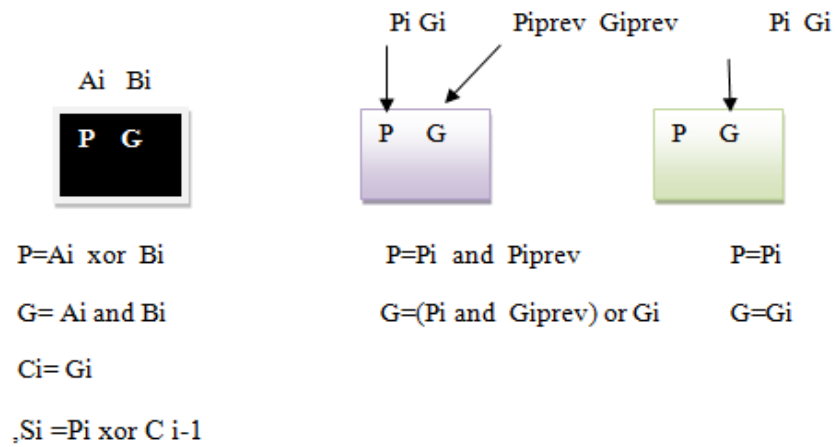


Figure 3.14: Representation of each KSA block

3.3 Implementation of 4-bit Adders

For implementation of 4-bit adders we have used Xilinx ISE 14.1 Design Suite, area and delay values are calculated from synthesis report while Power is calculated by Power analyzer in which we have calculated IOs Power and Leakage Power. The terms used in Table 3.3 are explained as follows:

- Look-Up Tables (LUT):-** In Configurable Logic Block (CLBs) function generators are implemented using LUT. When LUT's inputs are given then a block of SRAM is indexed. The output of LUT depends on whatever value is in indexed location in its SRAM. This is because when chip is powered up, contents have to initialize and as RAM is volatile.
- Slices:** - In FPGA slices are the basic building block components. Before mapping the logic of design, number of elements which each slice contain make up. All of the Flip

flop and LUT's are packed into slices after mapping, not necessarily filling the slices. In the map report, even partially used slice is counted in the "occupied slices".

- c) **Input/output Block (IOB):-** In FPGA device, input and output functions are implemented from the grouping of basic elements. Such collection and grouping of basic elements is termed as an IOB.
- d) **Delay:** - Delay is the time required for the input to be propagated to the output. In other words, the delay of a logic gate is defined as the time it takes for the effect of a change in input to be visible at the output
- **Router delay:** - Router delay can be ~40% of total delay.
 - **Logic delay:** - Logic delay can be more than 50% of total delay.
- e) **Power:** - Power dissipation of two types a) static b) dynamic.
- Static power dissipation- Static power lost is due to current leakage in the transistors of an FPGA.
 - Dynamic power dissipation- Dynamic power consumption is caused by signal alteration.

Table 3.3 shows the comparison of different adders for various performance parameters.

Table 3.3: Area, Delay and Calculation of different 4-bit adders

Sr. No.	Design	No. of 4 I/P LUT	No. of occupied slices	No. of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
				I- Buf	O -- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	4 bit RCA	8	4	9	5	7.306	2.768	0.021	0.034	0.543
2.	4 bit Carry Skip adder	10	6	9	5	6.602	2.134	0.012	0.034	0.401
3.	4 bit Carry Select	11	6	9	5	6.637	2.099	0.021	0.034	0.471

	adder									
4.	4 bit Carry look ahead adder	8	4	9	5	7.306	2.576	0.008	0.034	0.415
5.	4 bit Kogge stone adder	7	4	9	5	6.602	2.006	0.008	0.034	0.361

3.4 Proposed Adder

Among the parallel adders *Carry Increment Adder (CIA)* has the best delay performance which is one of the most important parameter in the high speed devices. The resulting CIA cuts the circuit size down by 23% with no change in performance. CIA is preferred for large word lengths (upto 128 bits) as the power delay product is smallest among all the known adder architectures. An 8-bit CIA adder includes two blocks of adder each 4 bit. In CIA only one partial sum is calculated and incremented if necessary, according to the input carry but in Carry select adder, from each group computing two partial sums and selecting the correct one. We have implemented modified 8-bit CIA using KSA which provides less delay than already implemented 8-bit CIA. As we have seen from Table 3.3, among all the 4-bit adders KSA has the best performance in terms of delay i.e. 8.608ns. Therefore, we have used KSA to implement 8-bit CIA.

- a) **CIA using RCA:** CIA consists of incremental circuitry and RCA's. A desired number of 4-bit inputs add by RCA and generating partitioned sum and partitioned carry. Using HA's in ripple carry sequence with an in order the incremental circuit is calculated. For example, two 4-bit RCA is required to implement an 8-bit CIA. From the first block of RCA, we directly get the 4-bit sum of CIA. And the first RCA block carry output is given as input to the C_{in} of an incremental circuit. The incremental circuit consists of half adders and the second RCA

block the partial sum obtained is given to incremental circuit. The block diagram of an 8-bit CIA_RCA is shown in Figure 3.15.

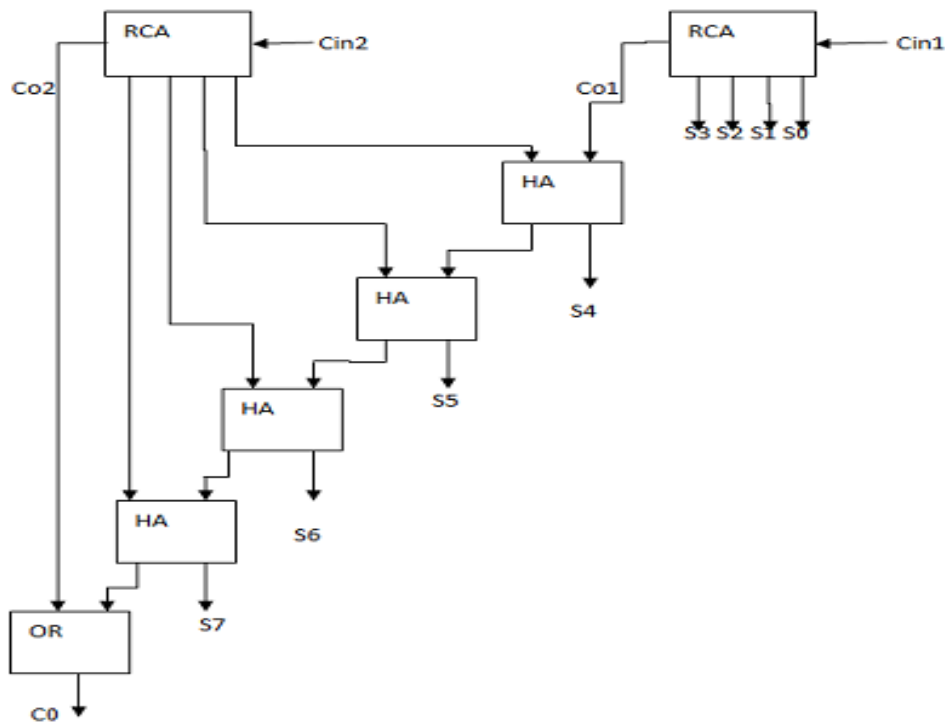


Figure 3.15: Block diagram of CIA_RCA

- b) **CIA using KSA:** The KSA replaces the ripple carry adder, in order to reduce the delay of the regular CIA. The modified 8-bit CIA using KSA is shown in Figure9. KSA suffer from complexity in prefix network due to an increase in number of wiring and logic cells. A delay efficient KSA is proposed. Among the parallel adders carry increment adder has the best performance which is one of the most important parameter in the high speed devices. The proposed design is a new concept and to the best of our knowledge it has not been proposed earlier by any researcher. In this sub section, we present the modified Carry increment adder i.e. CIA_KSA. We know that RCA design is simple and implementation is easy, but it suffers from worst propagation delay. It is proved that KSA performs better than RCA in term delay at the expense of increased design complexity. We have modified CIA_RCA by replacing the RCA block with KSA block. Because of the property of KSA, the overall delay performance will be improved. As similar to CIA_RCA incremental circuit can be calculated

using HA's in ripple carry sequence with an in order. The block diagram representation of CIA_KSA is shown in Figure 3.16 and RTL schematic in Figure 3.18.

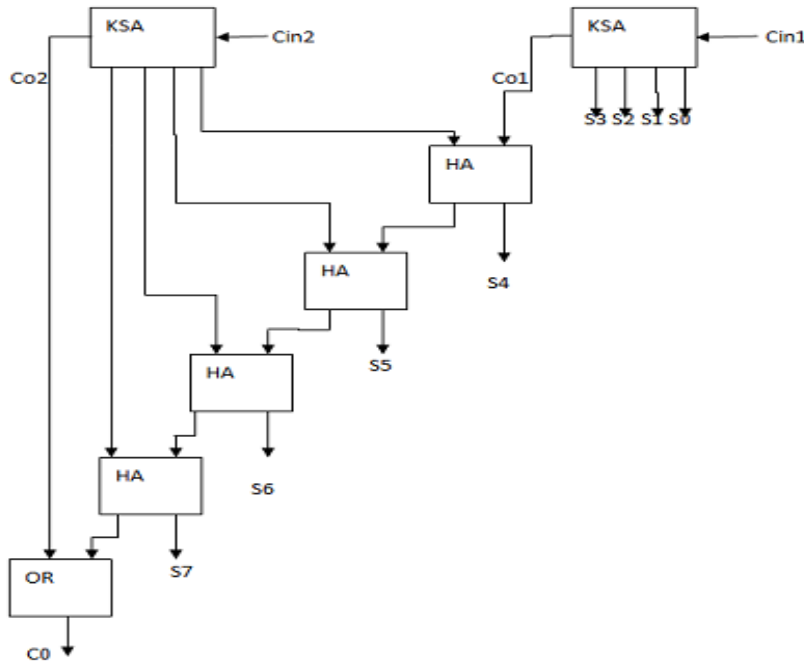


Figure 3.16: Block diagram of CIA_KSA

From the synthesis report, the performance parameters like area and delay are obtained and from power analyzer power is calculated which is shown in Table 3.4. It can be observed that the proposed design for 8bit Carry increment adder has better delay performance which is the desired goal of this research work.

Table 3.4: Area, Delay and Power calculation of 8 bit CIA

Sr. No.	Design	No. of 4 I/P LUT	No. of occupied slices	No. Of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
				I- Buf	O- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	8bit CIA_RCA	19	11	18	9	9.418	4.502	0.008	0.034	0.585
2.	8bit Proposed CIA_KSA	21	12	18	9	8.714	3.631	0.021	0.034	0.666

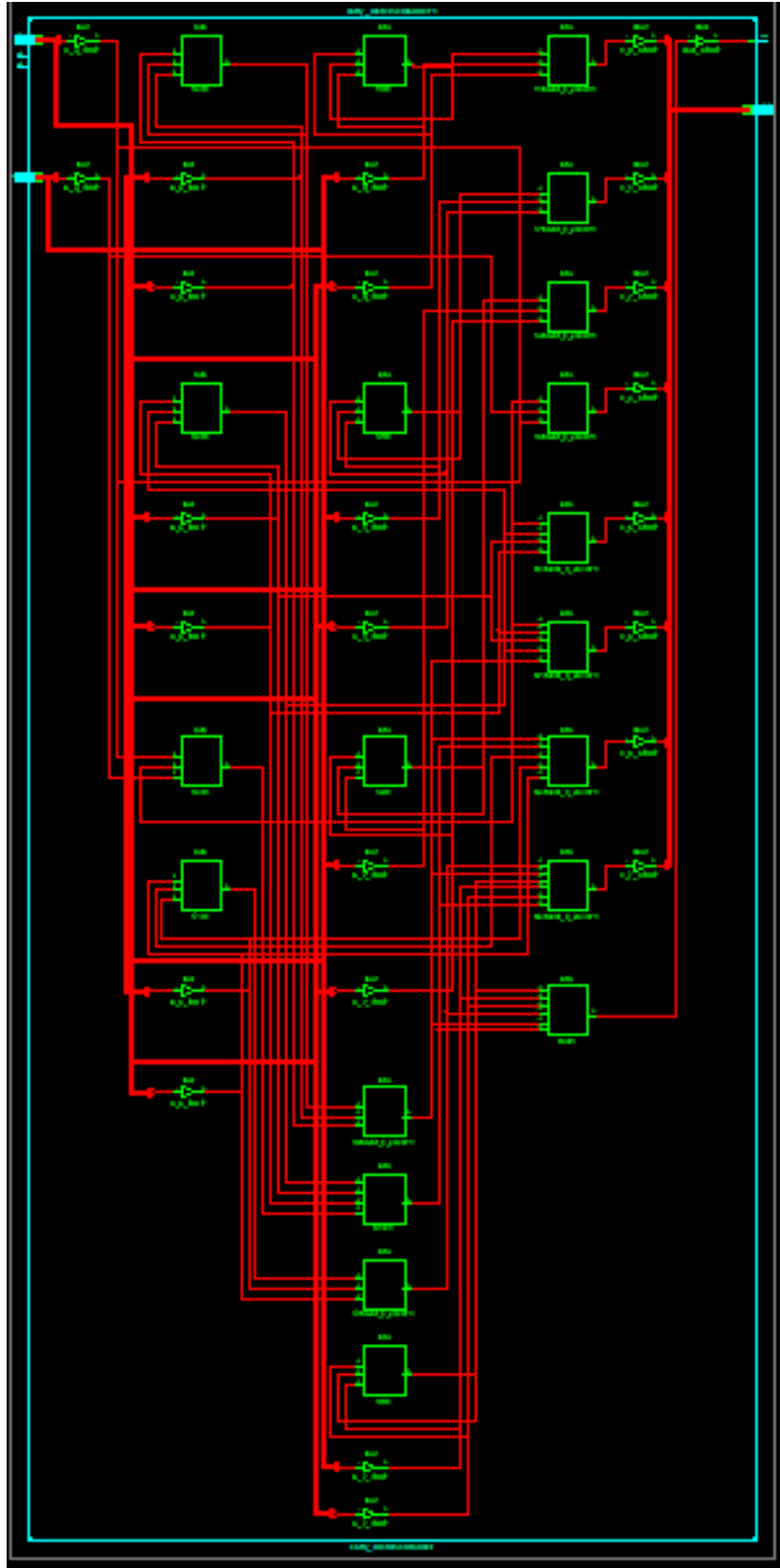


Figure 3.17: RTL Schematic of CIA_RCA

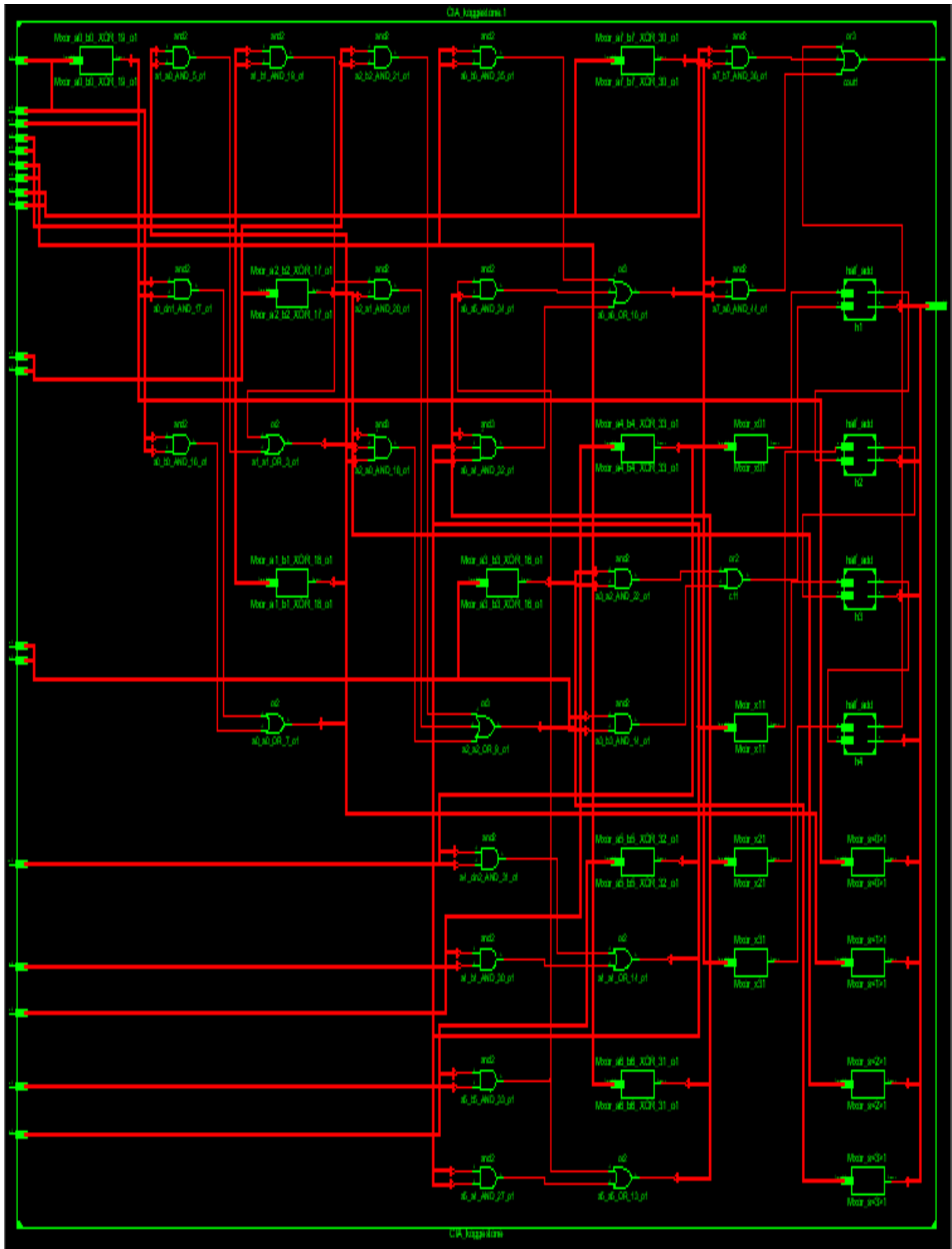


Figure 3.18: RTL Schematic of CIA_KSA

3.5 Comparison Table

Table 3.5 signifies the comparison of designed 8-bit CIA with the existing adder. Our proposed circuit gives the best delay i.e. 12.345 ns for CIA_KSA in comparison with Devi A.B. *et al.* 2016 [11] whose delay is 14.59 ns for CIA_RCA and 13.54 ns for CIA_CLA.

Table 3.5: Delay and Power calculation of 8-bit CIA using KSA

	No. Of occupied Slices	LUTs	Delay(ns)	Power(W)
Proposed Work CIA_KSA	12	21	12.345	0.054
Devi AB <i>et al.</i> 2016 [20] CIA_RCA	13	20	14.59	0.041
Devi AB <i>et al.</i> 2016 [11] CIA_CLA	12	19	13.54	0.041

3.6 Conclusion

The performance of any circuit in VLSI design limits by the constituent factors like power, delay and area. In this chapter a modified carry increment adder is proposed using KSA instead of ripple carry adder. Without affecting the circuit the delay performance of the circuit is improved by replacing the 4-bit RCA with a proposed 4-bit KSA. But the proposed CIA_KSA has the disadvantage of more power consumption. The design is tested and verified by Verilog HDL coding and simulation is carried out by in Xilinx ISE 14.1 design suite and synthesized for Spartan 3E FPGA. The delay performance of KSA is better than RCA but as operand size increases (32-bits and above). KSA suffers from complexity due to an increase in the number of logic cells and wiring. Future work may be dedicated to studying the complexity of CIA_KSA when the number of bits was increased.

CHAPTER 4

MULTIPLIERS

Multiplication is the mathematical operation that at its simplest is an abbreviated process of adding an integer to itself, a specified number of times and can be measured as a chain of repeated additions. The number which is to be added is called the multiplicand, the number of times which is added is called the multiplier and the result being given is known as the product. Multiplication is an important fundamental function in arithmetic operations. Many researchers have tried and are trying to design multiplier which offers either of the following- high speed, less area and low power consumption. We describe different types of multipliers: Array multiplier, Wallace tree multiplier, Vedic multiplier. Designer mainly concentrates on efficient circuit design. Characteristics of an efficient multiplier: *Speed*-At high speed multiplier should perform operation, *Accuracy*- Correct result should be given by good multiplier, *Area*- Less number of LUTs and Slices are occupied by multiplier and *Power*- The power consumed by the multiplier is less.

Three main steps of multiplication process:-

1. Generation of partial product
2. Addition of partial product
3. Final addition

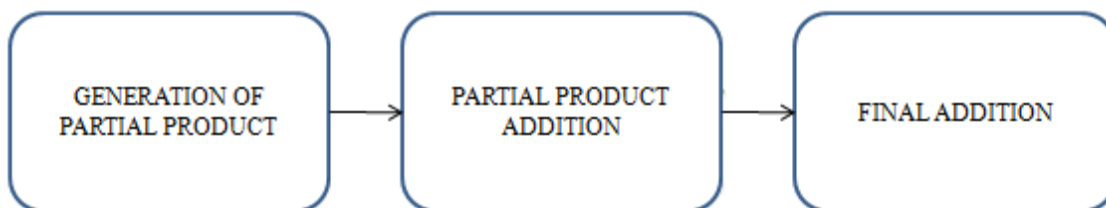


Figure 4.1: Block diagram of Multiplier architecture

Block diagram consists of three stages, in the first stage partial products are generated by multiplying bit by bit of multiplier and multiplicand. In the next stage there is addition of generated partial product, this stage is complex and the speed of circuit is derived and last stage

generated the output result by added the two-row outputs. Parallel multipliers are the most rapid multiplier type. The earlier performance of multiplier is enhanced to developed number of technique.

Let the multiplicand and multiplier be A and B:

$$A = a_{(M-1)}.a_{(M-2)} \dots a_1 a_0 = \sum_{i=0}^{M-1} a_i \cdot 2^i \quad (4.1)$$

$$B = b_{(N-1)}.b_{(N-2)} \dots b_1 b_0 = \sum_{i=0}^{N-1} b_i \cdot 2^i \quad (4.2)$$

The value of their product $P = A \times B$ is given by Equation 4.3:

$$P = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (a_i b_j \cdot 2^{i+j}) \quad (4.3)$$

Equation 4.4 and 4.5 expressed signed binary number and Equation 4.6 defined the product of A and B.

$$A = -a_{M-1} \cdot 2^{M-1} + \sum_{i=0}^{M-2} a_i \cdot 2^i \quad (4.4)$$

$$B = -b_{N-1} \cdot 2^{N-1} + \sum_{i=0}^{N-2} b_i \cdot 2^i \quad (4.5)$$

The product $P=A \times B$ is given by Equation 4.6:

$$P = (-a_{M-1} \cdot 2^{M-1} + \sum_{i=0}^{M-2} a_i \cdot 2^i) \times (-b_{N-1} \cdot 2^{N-1} + \sum_{i=0}^{N-2} b_i \cdot 2^i) \quad (4.6)$$

4.1 Different Multipliers

4.1.1 Array Multiplier

It is regular in structure and to go from one block to adjacent block short wires are used. In VLSI its layout is efficient and simple. N partial product is generated when there is multiplication of multiplier and multiplicand bit by bit as expressed by Equation 4.3. Multiplication is depends on Add/Shift algorithm. Figure 4.2 shows the $M \times N$ multiply operation of array multiplier and by ANDing multiplicand and multiplier partial products are generated. 4×4 array multiplier is shown in Fig.4.3.

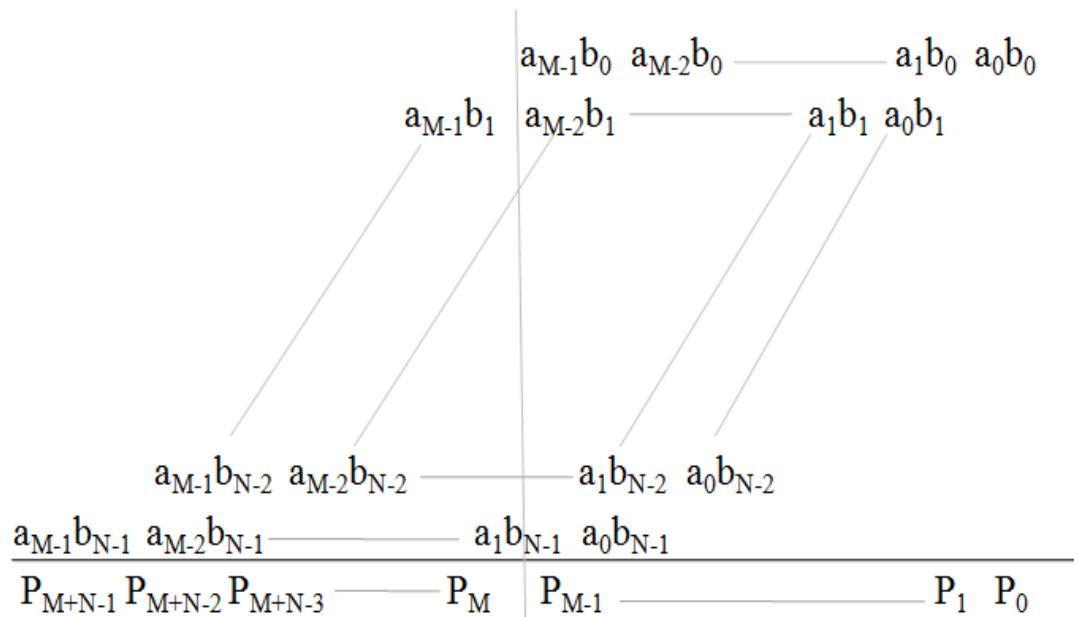


Figure 4.2: Partial product array for an $M \times N$ multiplier

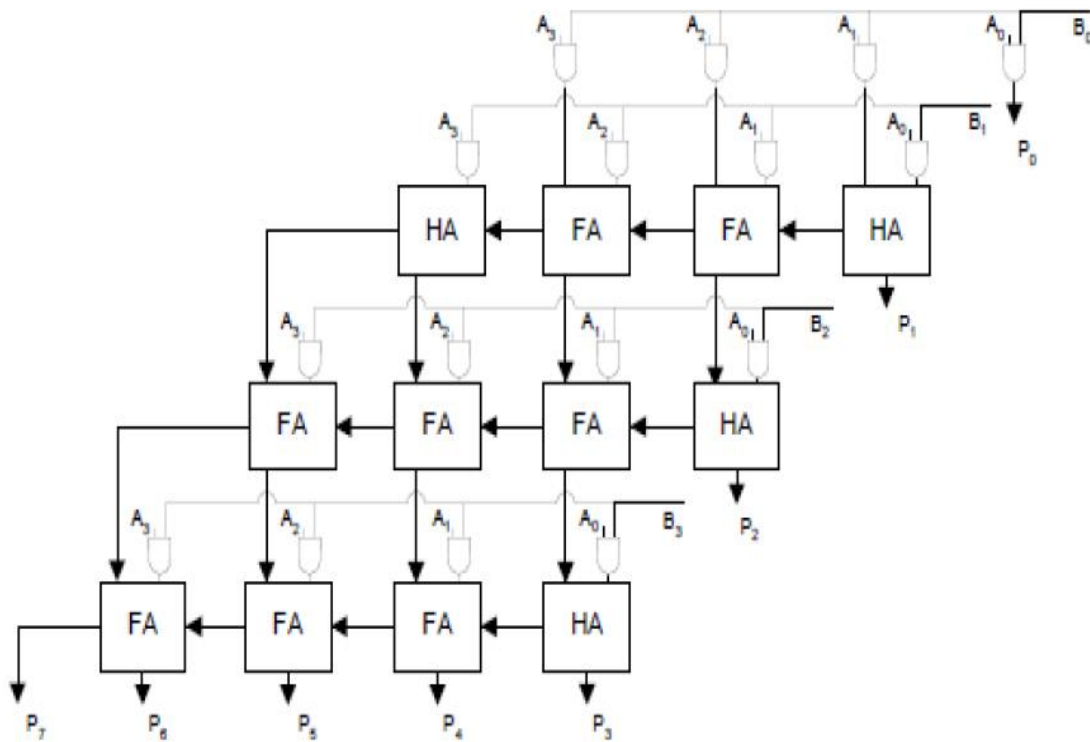


Figure 4.3: 4×4 Array multiplier

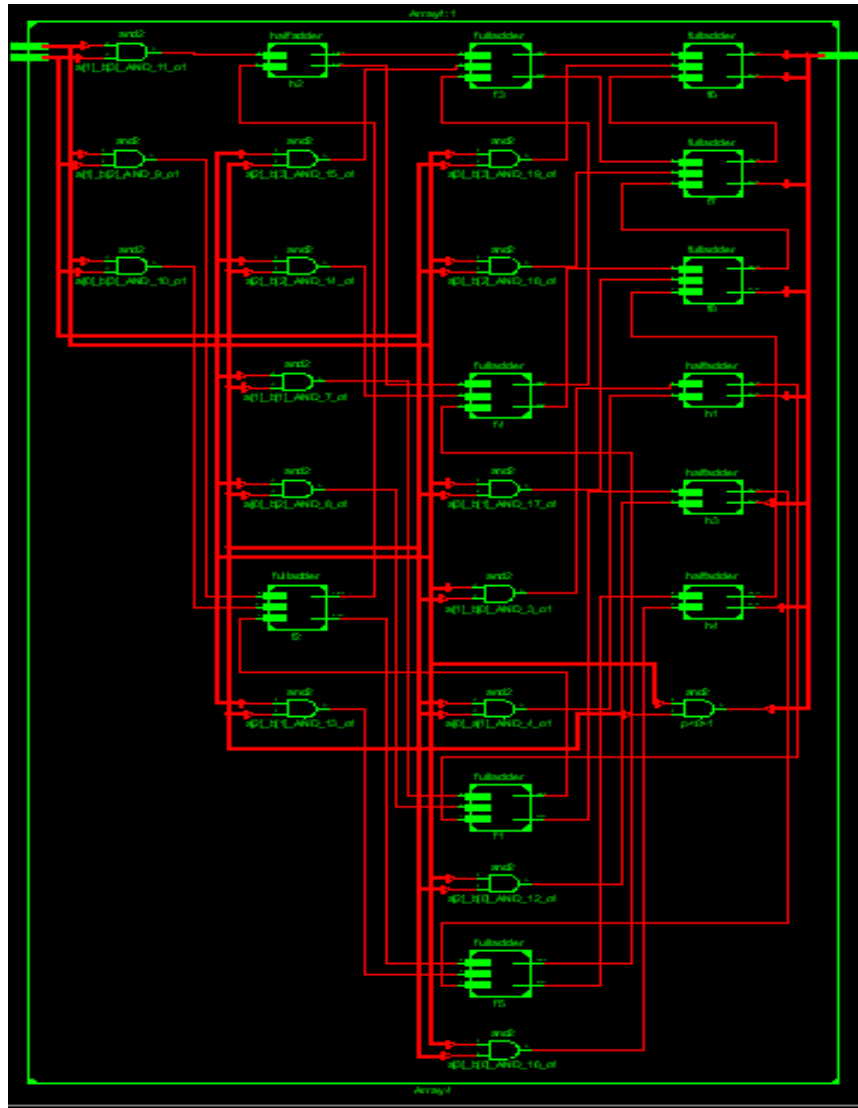


Figure 4.4: RTL Schematic of 4 bit Array multiplier

4.1.2 Wallace multiplier

In this multiplier there is parallel addition of generated partial products, so it takes less time for accumulation than array multiplier because in array multiplier the partial products are added in series. 8×8 bit partial product reduction is shown in Figure 4.5. In this Figure the two circled dots represent HA and tree circled dots represent FA. After four stages partial product is reduced to two rows. To reduce tree structure there are so many ways but only one method of reduction is shown.

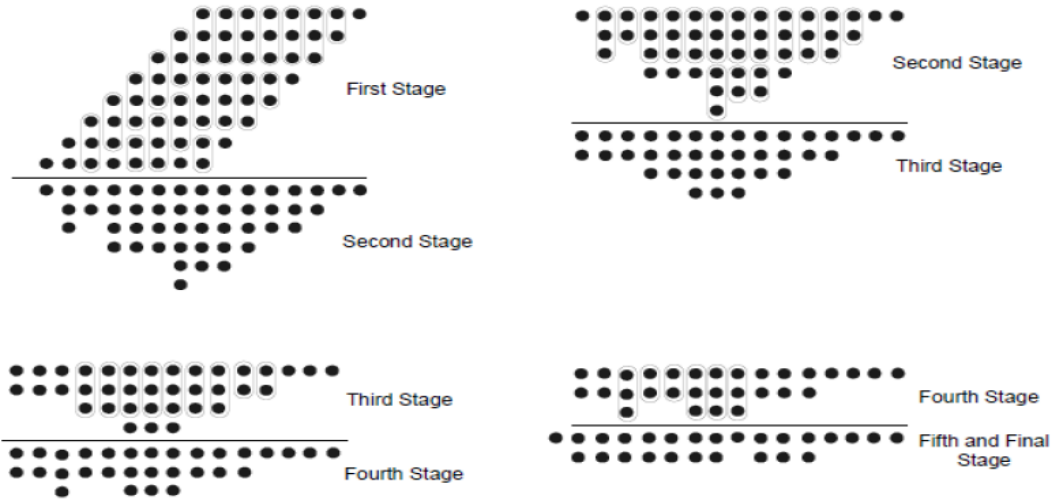


Figure 4.5: 8x8 partial product tree reduction of Wallace multiplier

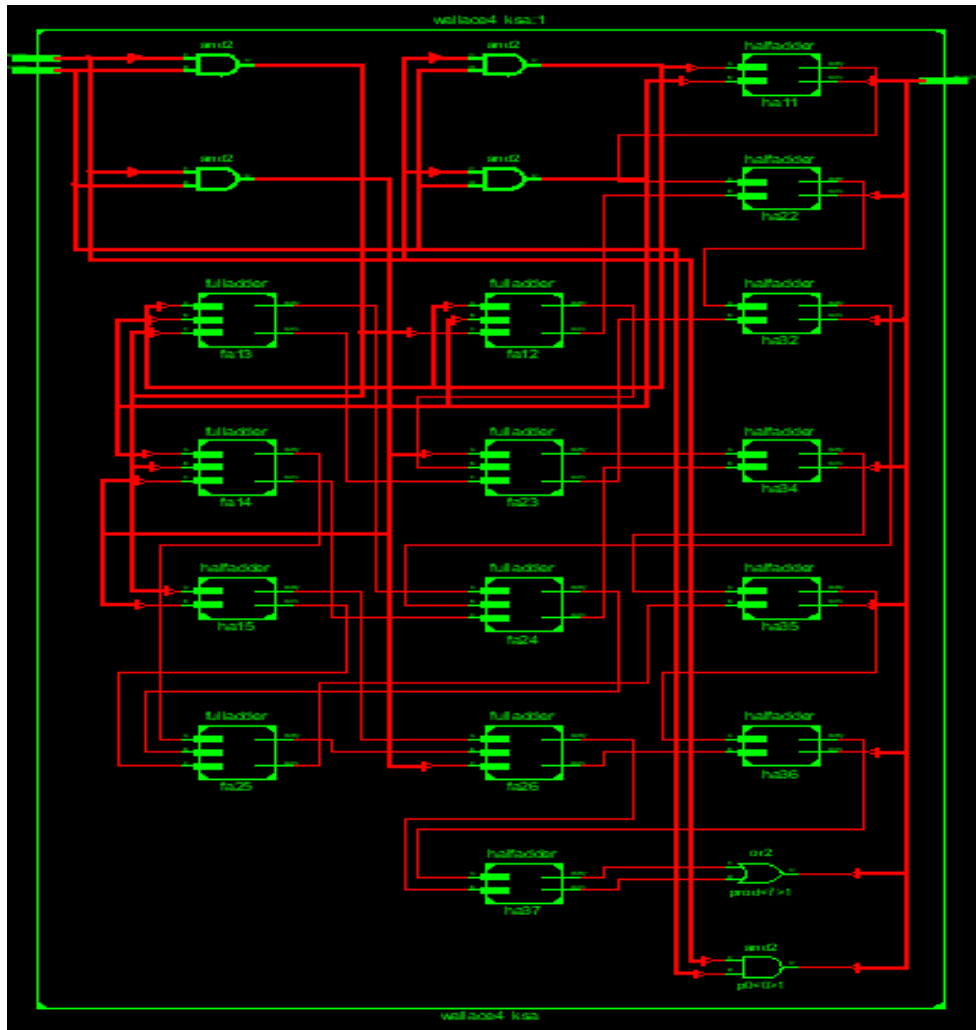


Figure 4.6: RTL Schematic of 4 bit Wallace multiplier

4.1.3 Vedic Multiplier

The word “Vedic” is derived from the word “Veda” which means the store house of knowledge. Veda consist of 16 sutras which encapsulate the branches of Mathematics- geometry, calculus, arithmetic, trigonometry etc. These sutras are : *Shunyamanyat(Anurupye)*, *Chalana-Kalanabyham*, *Ekadhikina Purvena*, *Ekanyunena Purvena*, *Gunakasamuchyah*, *Gunitasamuchyah*, *Nikhilam Navatashcaramam Dashatah*, *Paraavartya Yojayet*, *Puranapuranaabhyam*, *Sankalana-vyavakalanabhyam*, *Shesanyankena Charamena*, *Shunyam Saamyasamuccaye*, *Sopaantyadvayamantyam*, *Urdhva-tiryakbyham*, *Vyashtisamanstih*, *Yaavadunam*.

Vedic Multiplier using “UrdhvaTiryakbyham” Sutra:

In Sanskrit literature the ‘Urdhva’ means ‘vertically’ and ‘Tiryakbyham’ means ‘crosswise’. UrdhvaTiryakbyham is applicable to all cases of multiplication. In one step the algorithm produces sum and partial product. Once the number of bits is increased, this multiplier is advantageous as compared to other multipliers in terms of area and gate delay increases slowly.

For example: 131×121

Step	Explanation	Process	Result
1.	The numbers that lie on ones place are multiplied vertically and output is generated and stored result in ones place of the final result	$\begin{array}{r} 131 \\ \downarrow \\ \underline{121} \\ 1 \end{array}$	Result=1 Carry=0

2.	The numbers that lie on ones and tens place are multiplied by crossover multiplication and result is stored on tens place	$\begin{array}{r} 1\ 3\ 1 \\ \times 1\ 2\ 1 \\ \hline 5\ 1 \end{array}$	<p>Result=3+2=5 Carry=0</p>
3.	The numbers that lie on ones and hundred place are multiplied by crossover multiplication and number that lie on hundred place are multiplied by vertical multiplication, result of these multiplication are summed and final result stored in hundred place.	$\begin{array}{r} 1\ 3\ 1 \\ \times 1\ 2\ 1 \\ \hline 8\ 5\ 1 \end{array}$	<p>Result=1+6+1=8 Carry=0</p>
4.	The numbers that lie on tens and hundred place are multiplied by crossover multiplication and result is stored on thousand place	$\begin{array}{r} 1\ 3\ 1 \\ \times 1\ 2\ 1 \\ \hline 5\ 8\ 5\ 1 \end{array}$	<p>Result=3+2=5 Carry=0</p>

5.	Finally, Vertical multiplication of two numbers on hundred place are multiplied, 1 bit output is generated and stored result in ten thousand place of the final result	$ \begin{array}{r} 131 \\ \downarrow \\ \underline{121} \\ 15851 \end{array} $	Result=1 Carry=0
----	--	---	---------------------

Nikhilam Sutra

It literally means “all from 9 and last from 10” and when large numbers are involved it is more efficient. When the original number is larger the multiplication complexity is lesser. To perform the multiplication the compliment of the large number is find out from its nearest base.

For example: 131×121

Nearest Base = 100

$131 - 100 = 31$

$121 - 100 = 21$

$$\begin{array}{r}
 131 \quad 31 \\
 \swarrow \quad \searrow \\
 \underline{121 \quad 21} \\
 15|2| \\
 \underline{\quad 651} \\
 15851 \rightarrow \text{Result}
 \end{array}$$

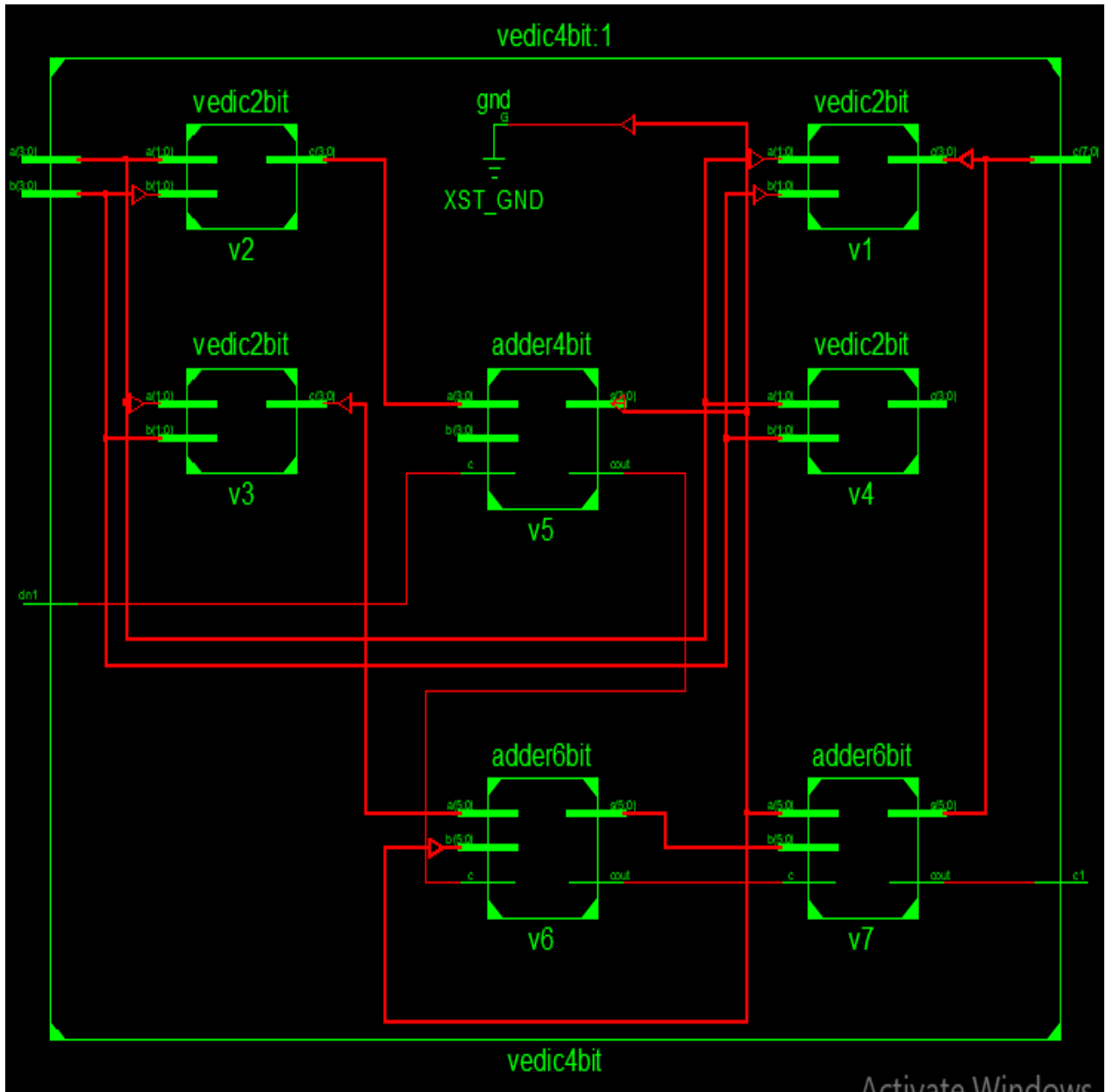


Figure 4.7: RTL Schematic of 4 bit Vedic multiplier

4.2 Implementation of 4- bit multipliers

For implementation of 4-bit multipliers we have used Xilinx ISE 14.1 Design Suite, area and delay values are calculated from synthesis report while Power is calculated by Power analyzer in

which we calculated IOs Power and Leakage Power. The comparison of different multipliers in terms of area, delay and power is shown in Table 4.1

Table 4.1: Area, Delay and Power calculation of 4-bit Multipliers

Sr. No.	Design	No. of 4 I/P LUT	No. of occupied slices	No. of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
				I- Buf	O – Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	4 bit Array multiplier	29	17	8	8	9.171	4.486	0.001	0.034	0.4779
2.	4 bit Wallace multiplier	33	19	8	9	7.947	3.928	0.001	0.034	0.4156
3.	4 bit Vedic multiplier	39	22	9	9	8.837	3.995	0.029	0.034	0.8084

4.3 Proposed Design

8-bit multipliers are implemented using Kogge stone adder(KSA). Among all the adders KSA is best in term of performance i.e. delay, speed and it is basically a prefix based adder. We have implemented Array multiplier, Vedic multiplier, Wallace multiplier using KSA for different performance parameters. In term of delay Wallace multipliers have best delay i.e. 18.024ns but there are increased in power consumption. On the other hand in Array multiplier and Vedic multiplier there is decrease in the speed and decrease in the power consumption. Each multiplier has its own advantage and disadvantage depending on logic we are using.

8×8 Array Multiplier

8 by 8 Array multiplier is implemented by considering two 8-bits binary numbers $A[7:0]$ and $B[7:0]$. To implement 8 X 8 Array multiplier, 4 X 4 Array multipliers are used to generate partial products. For addition of generated partial product, three KSA of 8 bit are used. We are taking four 4 X 4 Array multiplier block, in the first block least significant bits(LSBs) of A and B are multiplied to generate $S[3:0]$ of final result. In second block most significant bits(MSBs) of A is multiplied with LSBs of B to generate input bits for first block of KSA and in third block LSBs of A is multiplied with MSBs of B to generate input bits for first block of KSA. In fourth block, MSBs of A and B are multiplied to generate input bits for third block of KSA. Then take first two KSA and the carry generated from these adders are ORed. By ORing these two KSA a carry is generated which is applied a input to next KSA. In some blocks of KSA zero inputs are applied according to the requirement. KSA arrangement are made in such way that the speed of working is increased. Finally $sum[15:0]$ and carry(C_3) is generated and architecture of 8 X 8 Array multiplier is shown in Figure 4.8.

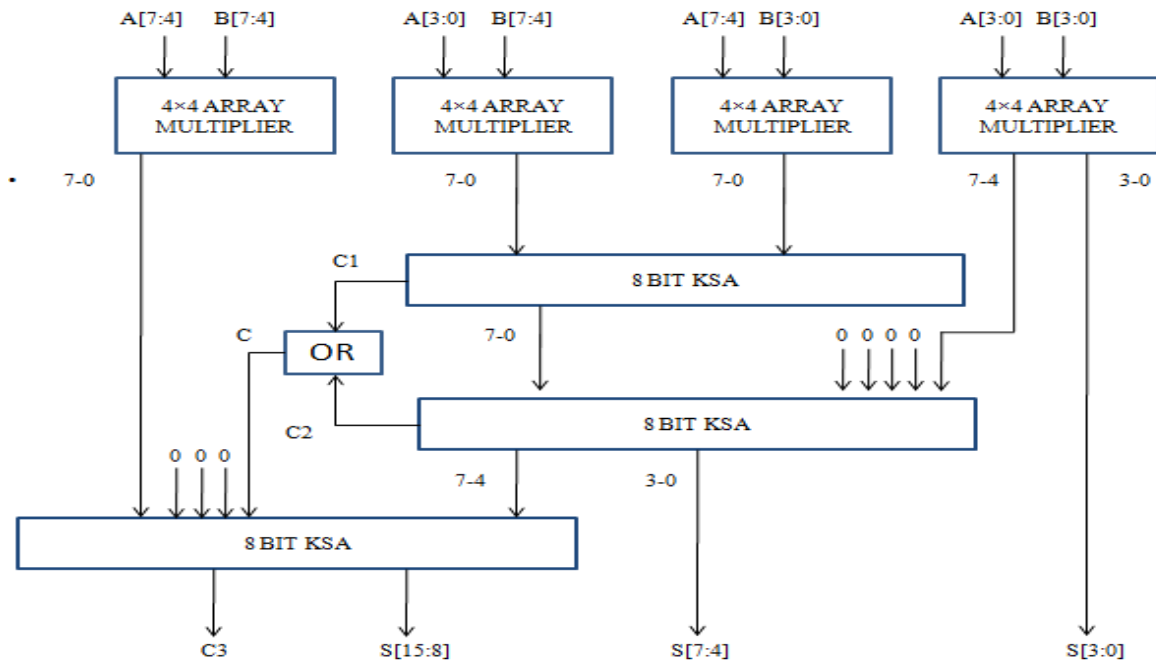


Figure 4.8: 8×8 Array multiplier architecture

8 X 8 Vedic Multiplier

8 by 8 Vedic multiplier is implemented by considering two 8-bits binary numbers $A[7:0]$ and $B[7:0]$. To implement 8 X 8 Vedic multiplier, 4 X 4 Vedic multipliers are used to generate partial products. For addition of generated partial product, three KSA of 8 bit are used. We are taking four 4 X 4 Vedic multiplier block, in the first block least significant bits(LSBs) of A and B are multiplied to generate $S[3:0]$ of final result. In second block most significant bits(MSBs) of A is multiplied with LSBs of B to generate input bits for first block of KSA and in third block LSBs of A is multiplied with MSBs of B to generate input bits for first block of KSA. In fourth block, MSBs of A and B are multiplied to generate input bits for third block of KSA. Then take first two KSA and the carry generated from these adders are Ored. By ORing these two KSA a carry is generated which is applied a input to next KSA. In some blocks of KSA zero inputs are applied according to the requirement. KSA arrangement are made in such way that the speed of working is increased. Finally $sum[15:0]$ and carry($C3$) is generated and architecture of 8 X 8 Vedic multiplier is shown in Figure 4.9.

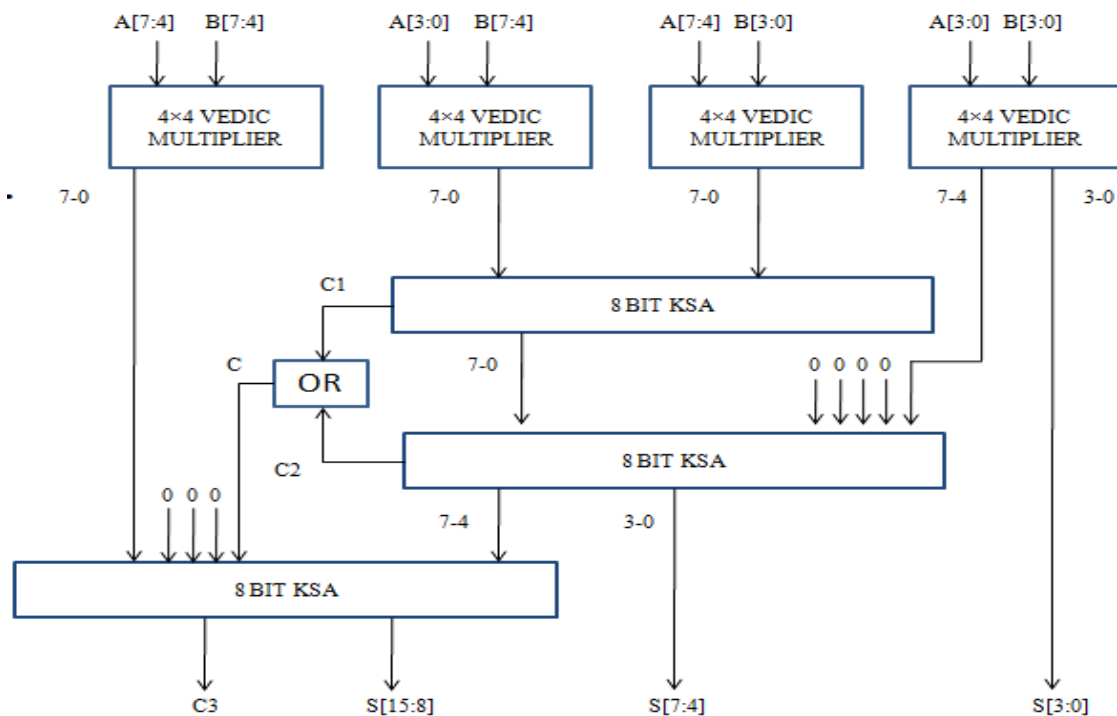


Figure 4.9: 8x8 Vedic multiplier architecture

8 X 8 Wallace Multiplier

8 by 8 Wallace multiplier is implemented by considering two 8-bits binary numbers $A[7:0]$ and $B[7:0]$. To implement 8 X 8 Wallace multiplier, 4 X 4 Wallace multipliers are used to generate partial products. For addition of generated partial product, three KSA of 8 bit are used. We are taking four 4 X 4 Wallace multiplier block, in the first block least significant bits (LSBs) of A and B are multiplied to generate $S[3:0]$ of final result. In second block most significant bits(MSBs) of A is multiplied with LSBs of B to generate input bits for first block of KSA and in third block LSBs of A is multiplied with MSBs of B to generate input bits for first block of KSA. In fourth block, MSBs of A and B are multiplied to generate input bits for third block of KSA. Then take first two KSA and the carry generated from these adders are ORed. By ORing these two KSA a carry is generated which is applied a input to next KSA. In some blocks of KSA zero inputs are applied according to the requirement. KSA arrangement is made in such way that the speed of working is increased. Finally $sum[15:0]$ and carry($C3$) is generated and architecture of 8 X 8 Wallace multiplier is shown in Figure 4.7 and RTL schematic in Figure 4.10.

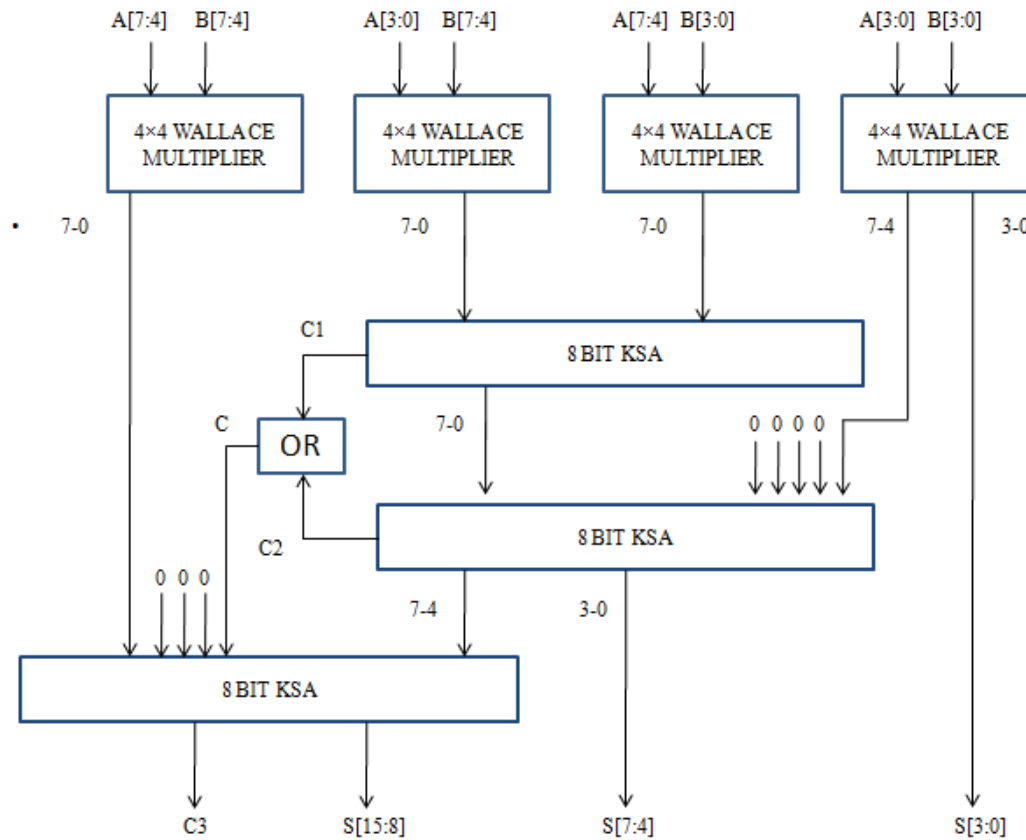


Figure 4.10: 8x8 Wallace multiplier architectur

Table 4.2: Area, Delay and Power calculation of 8 bit Multipliers

Sr. No.	Design	No. of 4 I/P LUT	No. of occupied slices	No. Of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
				I – Buf	O- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	Wallace_KSA	183	104	19	17	11.285	6.739	0.012	0.034	0.8291
2.	Array_KSA	171	98	19	17	13.121	7.850	0.001	0.034	0.7339
3.	Vedic_KSA	216	120	17	17	14.011	8.104	0.001	0.034	0.7740

Table 4.3 gives the comparison of designed 8-bit Wallace multiplier with the existing multiplier. Our proposed circuit of which multipliers gives the less delay i.e. 18.024 ns in comparison Rajaram S *et al.* 2011[6] whose calculated delay is 27.457 ns and Thomas A *et al.* 2016[33] whose delay is 39 ns. We have also calculated power which is less i.e. 46mW then Murugeswari S. *et al.* 2014[4] whose power is 264mW, 231mW while Rajaram S *et al.* 2011[6] has not reported any power.

Table 4.3: Area, Delay and Power calculation of 8 bit Wallace Multiplier

	Width	No. of occupied slices	No. of LUTs	Delay(ns)	Power(mW)
Proposed work Using KSA	8	104	183	18.024	46
Rajaram S <i>et al.</i> 2011[6]	8	-	-	27.457	-
Murugeswari S. <i>et al.</i> 2014[4] Using Full adder	8	87	163	17.223	264
Murugeswari S. <i>et al.</i> 2014[4] Using MUX based Full adder	8	84	155	17.789	231
Thomas A <i>et al.</i> 2016[33]	8	-	133	39	-

Table 4.4 gives the comparison of designed 8-bit VM with the existing multipliers. Our proposed circuit of which multiplier gives the best delay i.e 22.115ns in comparison Gokhale GR *et al.* 2015[3] whose delay is 44.358ns and Thomas A *et al.* 2016[33] whose delay is 34 ns using RCA and 30 ns using CLA. We have also calculated Power which is 35mW while Gokhale GR *et al.* 2015[3] and Anjana R *et al.* 2014[5] has not reported any power. Anjana R *et al.* 2014[5] calculated difference between logic delay and router delay which is 5.588ns and our proposed circuit difference between logic delay and router delay is 5.907 which is more but the no. of LUTs required is less than Anjana R *et al* 2014[5] .

Table 4.4: Area, Delay and Power calculation of 8 bit Vedic Multiplier

	Width	No. of LUTs	Area(gate count)	Delay(ns)	Power(W)
Proposed work Using KSA	8	216	-	22.115	0.035
Gokhale GR <i>et al.</i> 2015[3]	8	-	1293	44.358	-
Anjana R <i>et al.</i> 2014[5]	8	309	-	5.588	-
Thomas A <i>et al.</i> 2016[33] Using RCA	8	166	-	34	-
Thomas A <i>et al.</i> 2016[33] Using CLA	8	167	-	30	-

Table 4.5 gives the comparison of designed 8-bit AM with the existing multipliers. Our proposed circuit of which multiplier gives the best delay i.e. 20.971 ns in comparison to Maiti A *et al.* 2016[34] whose delay is 25.3 ns and Thomas A *et al.* 2016[33] whose delay is 44ns. We also calculated power which is more i.e 35 mW in comparison to Maiti A *et al.* 2016[34] whose power is 0.0606 mW.

Table 4.5: Area, Delay and Power calculation of 8 bit Array Multiplier

	Width	No. of LUTs	Delay(ns)	Power(mW)
Proposed work Array Multiplier	8	171	20.971	35
Maiti A <i>et al.</i> 2016[34] Using CMOS	8	-	25.3	0.0606
Thomas A <i>et al.</i> 2016[33]	8	126	44	-

It can be observed that the proposed design for 8bit Wallace Multiplier has better delay performance which was the desired goal of this research work.

4.4 Conclusion

The performance of any circuit in VLSI design limits by the constituent factors like power, delay and area. In this chapter Array multiplier, Vedic multiplier and Wallace multiplier are implemented using KSA. It is concluded that KSA have less delay and power as compared to other adders, so it is best suited for implementation of modified multiplier. Wallace multiplier has less delay compared to other multipliers but there is increase in power consumption. The design is tested and verified by Verilog HDL coding and simulation is carried out by in Xilinx ISE 14.1 design suite and synthesized for Spartan 3E FPGA. KSA suffers from complexity due to an increase in the number of logic cells and wiring. Future work may be dedicated to decrease in power consumption of Wallace multiplier.

CHAPTER 5

FAST FOURIER TRANSFORMS

Fast Fourier Transform (FFT) is used for Signal Processing applications. It consists of addition and multiplication operations, whose speed improvement will enhance the accuracy and performance of FFT computation for any applications. FFT are used to covert signal from time domain to frequency domain. In FFT processing unit Butterfly Structure is the basic building block and is used for the calculating complex calculation. So, it is important to design an efficient adder and multiplier block and used that efficient block in Butterfly Structure.

5.1 Efficient Adder using 16 bit and 32 bit

Efficient adder for 4 bit and 8 bit has been already implemented in Chapter 2. We have analyzed that KSA and CLA are the best suited adders in terms of delay and power. Now we need to implement 16 bit and 32 bit KSA and CLA so, that they can be used in high performance applications.

KSA

RCA has drawback that its delay goes on increasing as number of bits increases. To overcome this problem KSA is used. For high performance application large amount of bits are used for doing multiple calculation and tasks. So, there is a need to implement 32 bit KSA which is implemented by using 16 bit KSA. Firstly, it is important to implement 16 bit KSA. In KSA as the number of bits goes on increasing speed is increases but the drawback is that complexity increases. The KSA is the parallel prefix form that takes more area to implement. The RTL schematic of KSA for 16 bit and 32 bit are shown in Figure 5.1 and Figure 5.2.

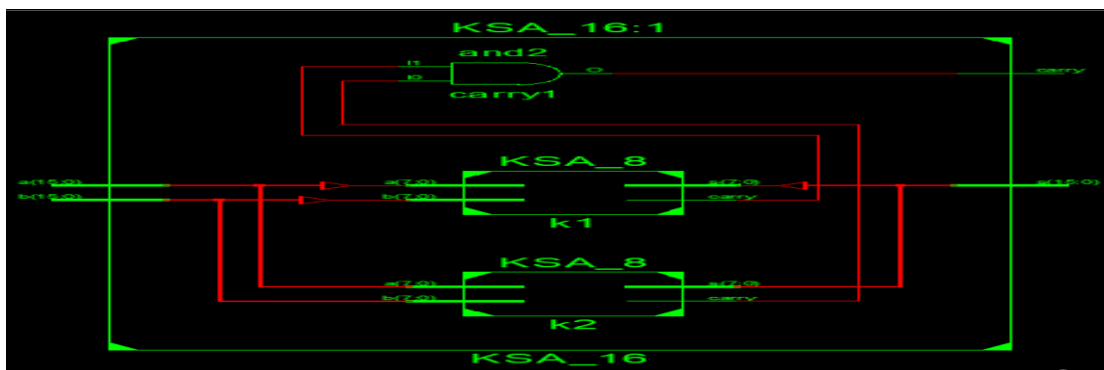


Figure 5.1: RTL Schematic of 16 bit KSA

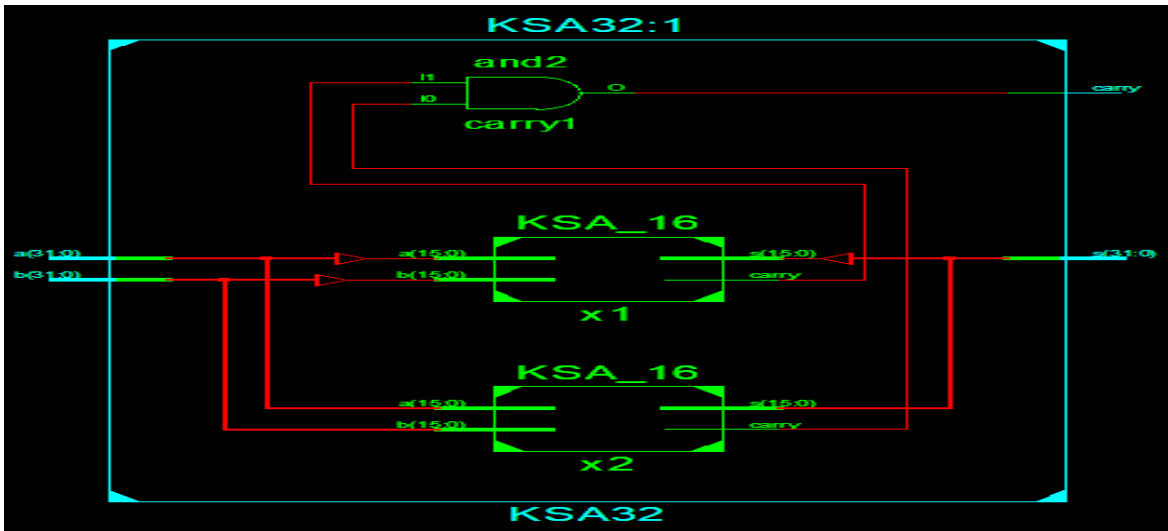


Figure 5.2: RTL Schematic of 32 bit KSA

In Section 3.2.4 it is already discussed about CLA. Figure 5.3 shows the RTL schematic of CLA for 16 bit and Figure 5.4 shows the RTL schematic of 32 bit CLA.

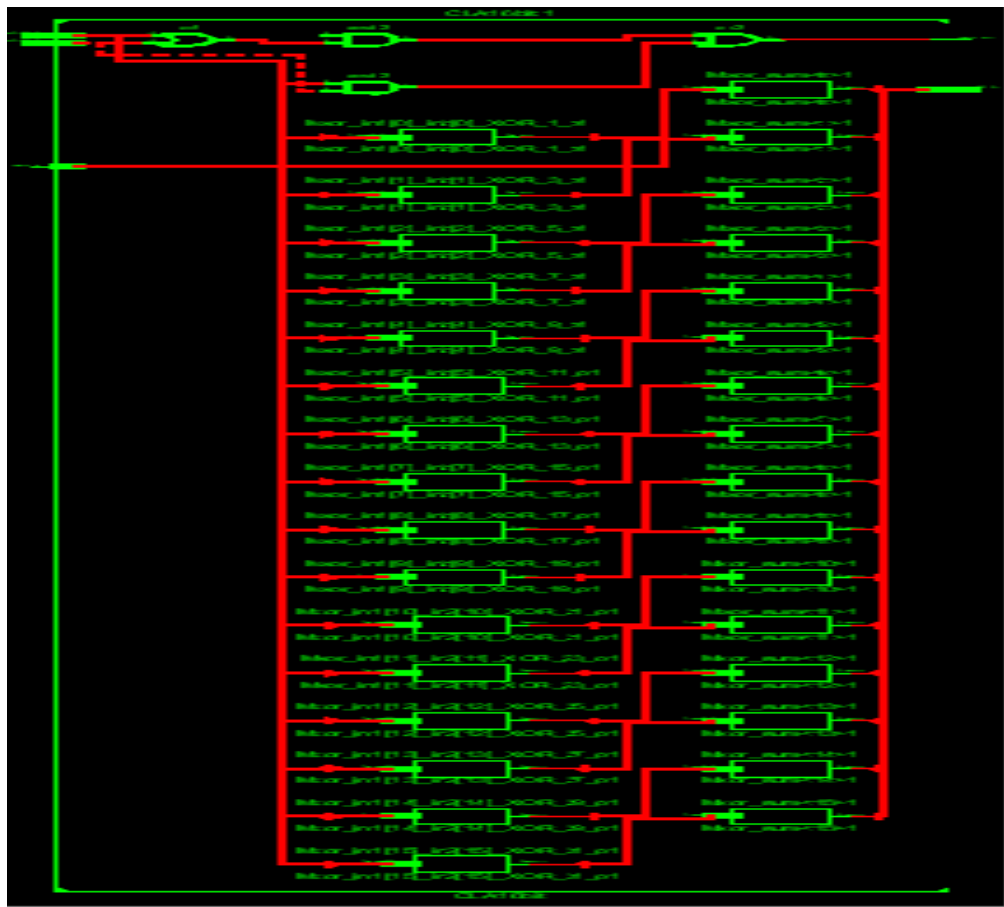


Figure 5.3: RTL Schematic of 16 bit CLA

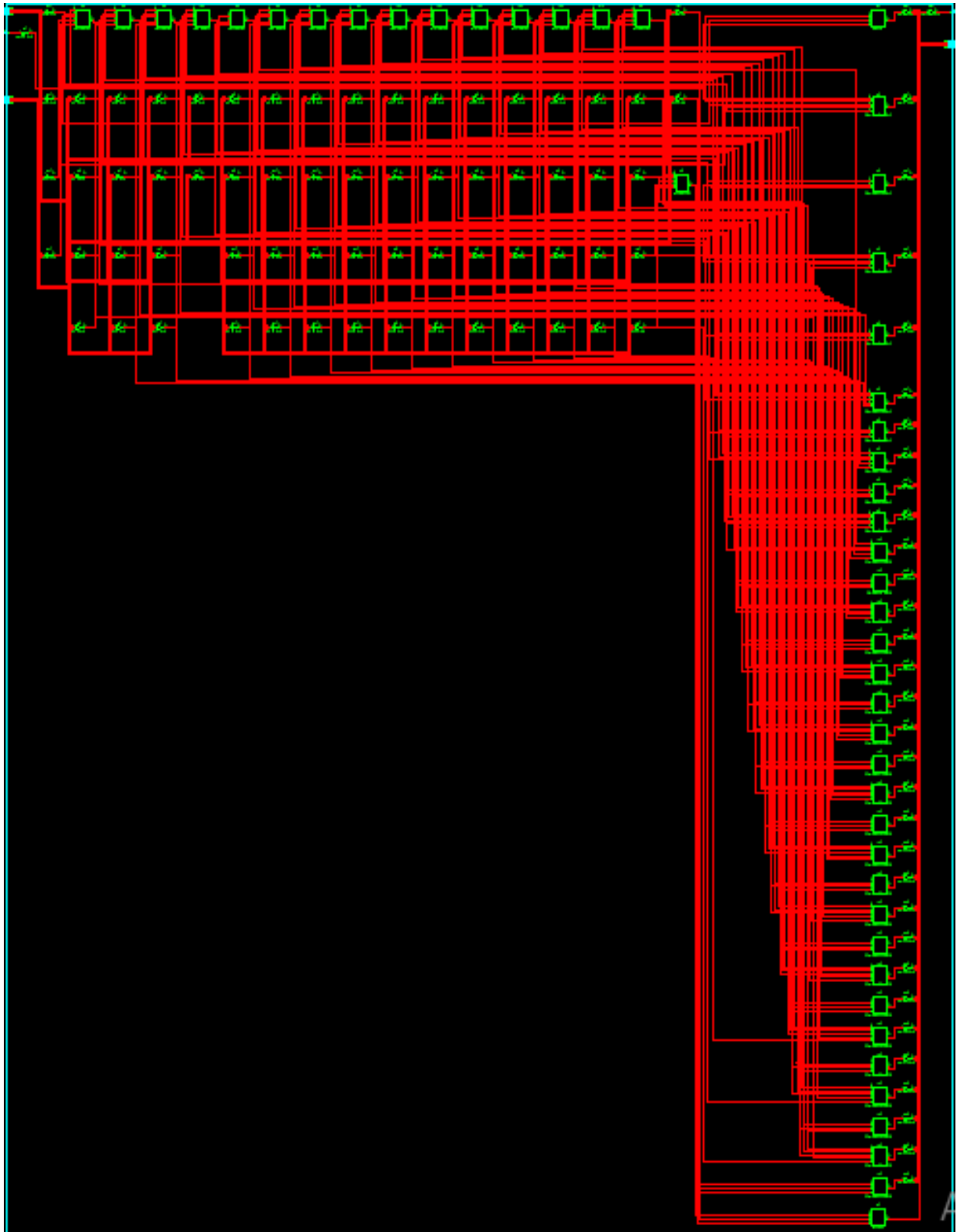


Figure 5.4: RTL Schematic of 32 bit CLA

From the synthesis report, the performance parameters like area and delay are obtained and from power analyzer power is calculated which is shown in Table 5.1. It can be observed that 16bit KSA has less delay compared to 16 bit CLA.

Table 5.1: Area, Delay and Power Calculation of 16 bit Adders

Sr. No.	Design	No. of Slice LUT	No. Of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
			I –Buf	O- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	KSA	35	32	17	4.406	2.853	0.017	0.081	0.7113
2.	CLA	25	33	17	5.636	6.910	0.017	0.081	1.2295

From the synthesis report, the performance parameters like area and delay are obtained and from power analyzer power is calculated which is shown in Table 5.2. It can be observed that 32 bit KSA has less delay compared to 32 bit CLA. From Table 5.1 and 5.2 it is observed that KSA is best suited adder for signal processing applications.

Table 5.2: Area, Delay and Power Calculation of 32 bit Adders

Sr. No.	Design	No. of Slice LUT	No. Of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
			I –Buf	O- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	KSA	71	64	33	4.402	3.345	0.017	0.081	0.7592
2.	CLA	49	65	33	7.276	12.120	0.017	0.081	1.9008

5.2 Efficient Multiplier using 16 and 32 bit

From Chapter 3 we have analyzed that Wallace multiplier is best suited multiplier for high speed application because its delay is less as compared to other multipliers. We have already implemented 4 bit and 8 bit Wallace multiplier in Chapter 3. Now we need to implement 16 bit

and 32 bit Wallace multiplier using high speed adders i.e. KSA and CLA, the best results are used for further processing. As we have already find out in Section 5.1 that KSA have less delay compared to CLA but we need to check for multiplier also which is best suited. Firstly, we implement 16 bit Wallace multiplier using KSA and CLA then 32 bit Wallace multiplier.

16 bit Wallace Multiplier using KSA

16 by 16 Wallace multiplier is implemented by considering two 16-bits binary numbers $A[15:0]$ and $B[15:0]$. To implement 16 X 16 Wallace multiplier, 8 X 8 Wallace multipliers are used to generate partial products. For addition of generated partial product, three KSA of 16 bits are used. We are taking four 8 X 8 Wallace multiplier block, in the first block least significant bits (LSBs) of A and B are multiplied to generate $S[7:0]$ of final result. In second block most significant bits(MSBs) of A is multiplied with LSBs of B to generate input bits for first block of KSA and in third block LSBs of A is multiplied with MSBs of B to generate input bits for first block of KSA. In fourth block, MSBs of A and B are multiplied to generate input bits for third block of KSA. Then take first two KSA and the carry generated from these adders are ORed. By ORing these two KSA a carry is generated which is applied a input to next KSA. In some blocks of KSA zero inputs are applied according to the requirement. KSA arrangement is made in such way that the speed of working is increased. Finally $sum[31:0]$ and carry(C3) is generated and architecture of 16 X 16 Wallace multiplier is shown in Figure 5.5.

16 bit Wallace Multiplier using CLA

16 by 16 Wallace multiplier is implemented by considering two 16-bits binary numbers $A[15:0]$ and $B[15:0]$. To implement 16 X 16 Wallace multiplier, 8 X 8 Wallace multipliers are used to generate partial products. For addition of generated partial product, three CLA of 16 bits are used. We are taking four 8 X 8 Wallace multiplier block, in the first block least significant bits (LSBs) of A and B are multiplied to generate $S[7:0]$ of final result. In second block most significant bits(MSBs) of A is multiplied with LSBs of B to generate input bits for first block of CLA and in third block LSBs of A is multiplied with MSBs of B to generate input bits for first

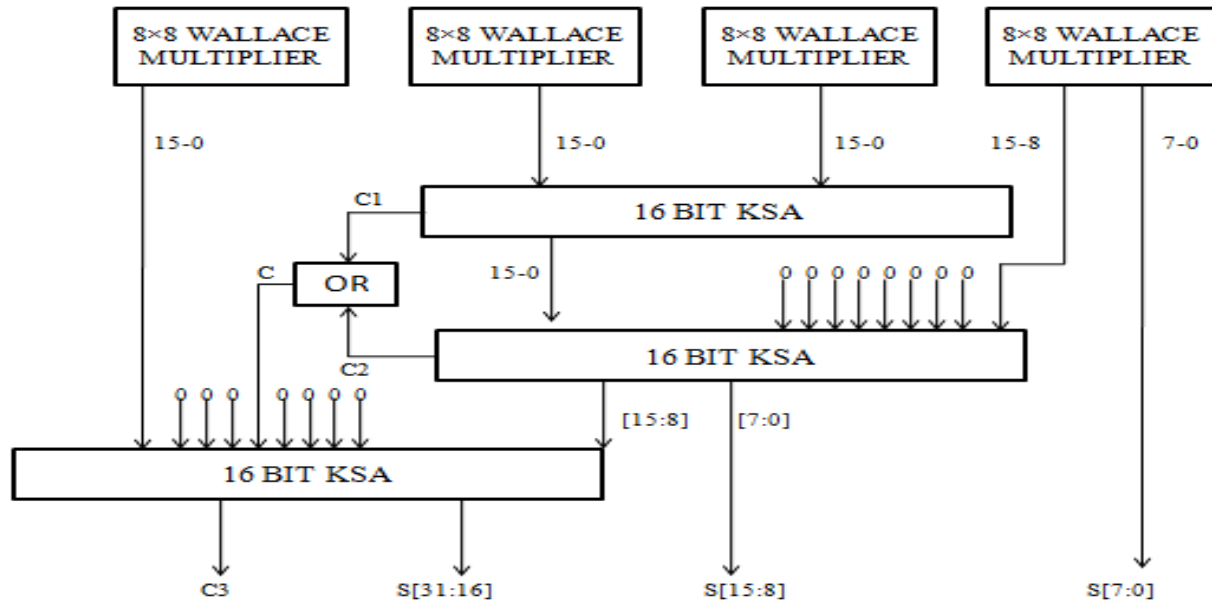


Figure 5.5: 16×16 Wallace multiplier architecture using KSA

block of CLA. In fourth block, MSBs of A and B are multiplied to generate input bits for third block of CLA. Then take first two CLA and the carry generated from these adders are ORed. By ORing these two CLA a carry is generated which is applied a input to next CLA. In some blocks of CLA zero inputs are applied according to the requirement. CLA arrangement is made in such way that the speed of working is increased. Finally sum[31:0] and carry(C3) is generated and architecture of 16X 16 Wallace multiplier is shown in Figure 5.6.

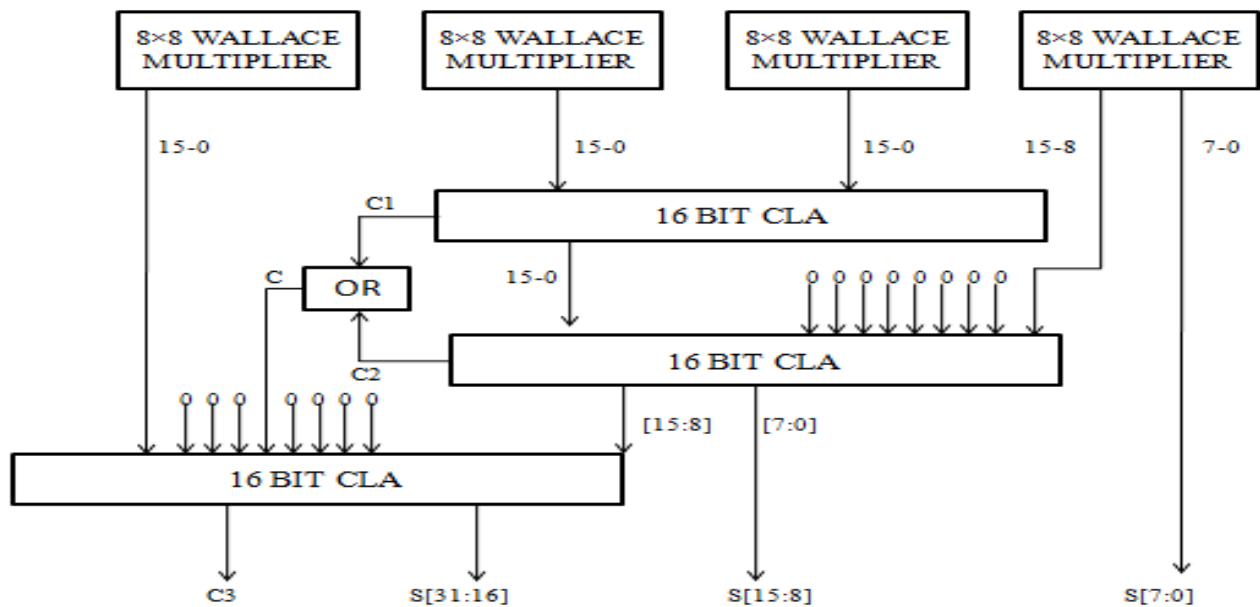


Figure 5.6: 16×16 Wallace multiplier architecture using CLA

From the synthesis report, the performance parameters like area and delay are obtained and from power analyzer power is calculated which is shown in Table 5.3. It can be observed that 16bit Wallace multiplier using KSA has less delay and less amount of energy consumed compared to 16 bit CLA.

Table 5.3: Area, Delay and Power calculation of 16 bit Wallace Multiplier

Sr. No.	Design	No. of Slice LUT	No. Of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
			I -Buf	O- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	Wallace_KSA	610	32	32	6.641	14.369	0.017	0.081	2.0589
2.	Wallace_CLA	543	32	33	7.654	17.900	0.017	0.081	2.5042

32 bit Wallace Multiplier using KSA

32 by 32 Wallace multiplier is implemented by considering two 32-bits binary numbers A[31:0] and B[31:0]. To implement 32 X 32 Wallace multiplier, 16 X 16 Wallace multipliers are used to generate partial products. For addition of generated partial product, three KSA of 32 bits are used. We are taking four 16 X 16 Wallace multiplier block, in the first block least significant bits (LSBs) of A and B are multiplied to generate S[15:0] of final result. In second block most significant bits(MSBs) of A is multiplied with LSBs of B to generate input bits for first block of KSA and in third block LSBs of A is multiplied with MSBs of B to generate input bits for first block of KSA. In fourth block, MSBs of A and B are multiplied to generate input bits for third block of KSA. Then take first two KSA and the carry generated from these adders are ORed. By ORing these two KSA a carry is generated which is applied a input to next KSA. In some blocks of KSA zero inputs are applied according to the requirement. KSA arrangement is made in such way that the speed of working is increased. Finally sum[63:0] and carry(C3) is generated and architecture of 32X 32 Wallace multiplier is shown in Figure 5.7.

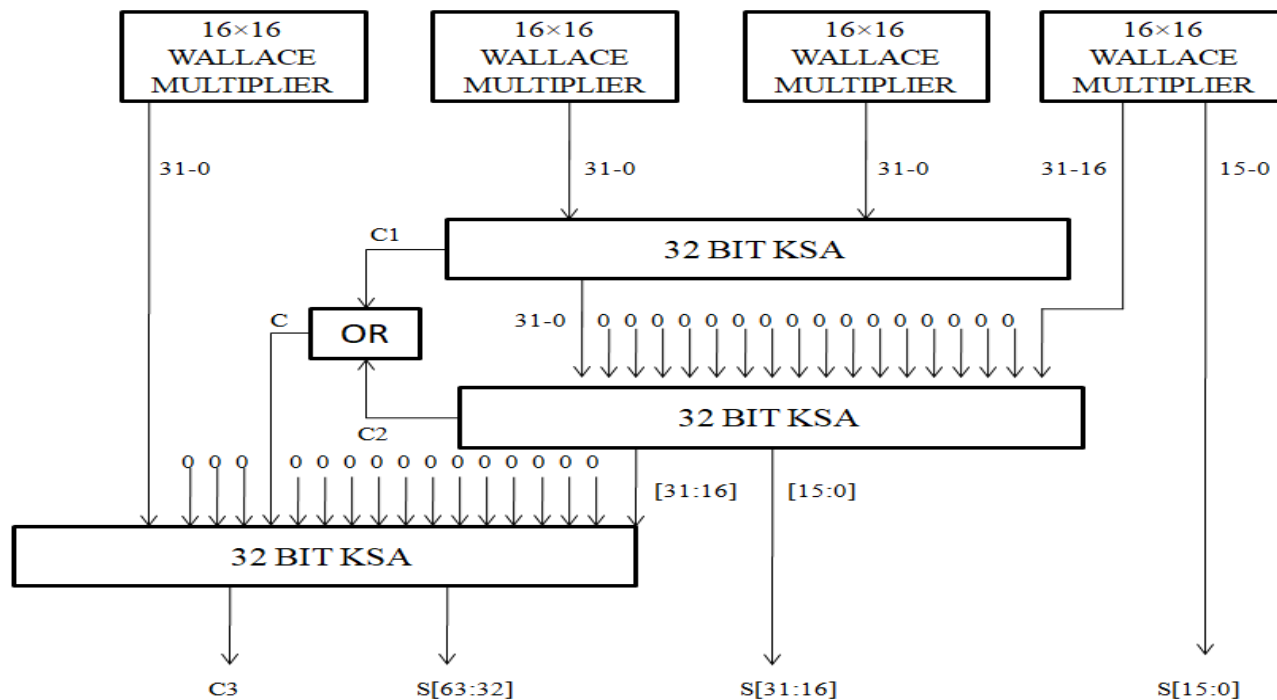


Figure 5.7: 32x32 Wallace multiplier architecture using KSA

32 bit Wallace Multiplier using CLA

32 by 32 Wallace multiplier is implemented by considering two 32-bits binary numbers $A[31:0]$ and $B[31:0]$. To implement 32 X 32 Wallace multiplier, 16 X 16 Wallace multipliers are used to generate partial products. For addition of generated partial product, three CLA of 32 bits are used. We are taking four 16 X 16 Wallace multiplier block, in the first block least significant bits (LSBs) of A and B are multiplied to generate $S[15:0]$ of final result. In second block most significant bits(MSBs) of A is multiplied with LSBs of B to generate input bits for first block of CLA and in third block LSBs of A is multiplied with MSBs of B to generate input bits for first block of CLA. In fourth block, MSBs of A and B are multiplied to generate input bits for third block of CLA. Then take first two CLA and the carry generated from these adders are ORed. By ORing these two CLA a carry is generated which is applied a input to next CLA. In some blocks of CLA zero inputs are applied according to the requirement. CLA arrangement is made in such way that the speed of working is increased. Finally $sum[63:0]$ and carry($C3$) is generated and architecture of 32X 32 Wallace multiplier is shown in Figure 5.8.

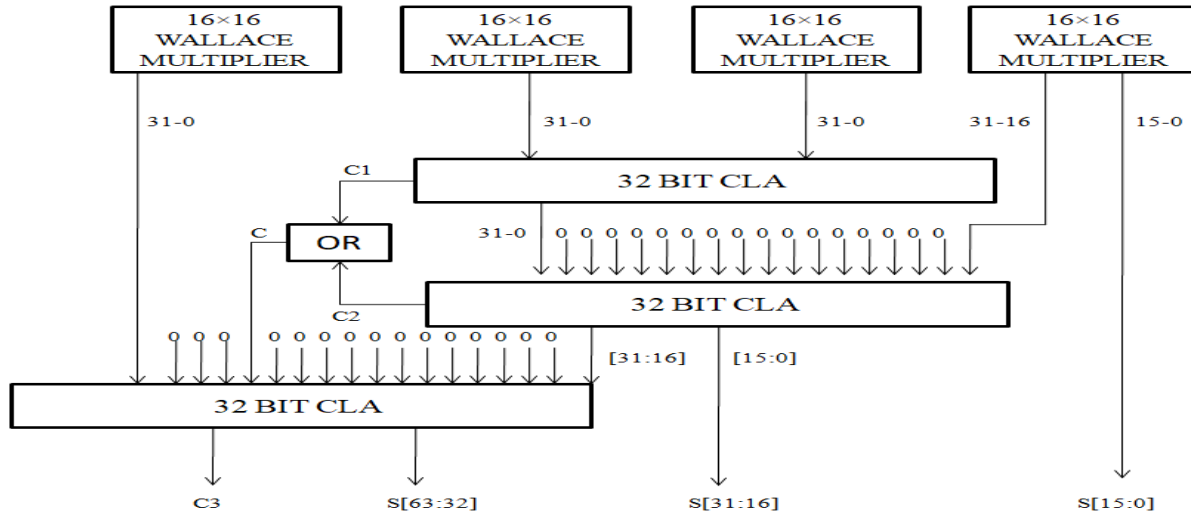


Figure 5.8: 32x32 Wallace multiplier architecture using CLA

From the synthesis report, the performance parameters like area and delay are obtained and from power analyzer power is calculated which is shown in Table 5.4. It can be observed that 32 bits Wallace multiplier using KSA has less delay and less amount of energy consumed compared to 32 bit CLA.

Table 5.4: Area, Delay and Power calculation of 32 bit Wallace Multiplier

Sr. No.	Design	No. of Slice LUT	No. Of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
			I-Buf	O- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	
1.	Wallace_KSA	2109	64	64	7.455	18.134	0.017	0.113	3.326
2.	Wallace_CLA	1819	64	65	11.314	33.189	0.017	0.113	5.78539

It is observed that Wallace multiplier implementation using KSA is best suited multiplier and KSA is best suited adder to enhance the accuracy and performance of FFT computation for any application. So, in butterfly structure these efficient multiplier and adder are used.

5.3 Transforms

Transforms are not that much important but they are used to covert calculations into simple and more convenient form. Either in time domain or in frequency domain the signal analysis and computation is possible. Transforms like Fourier are used to tell us about the property and frequencies present in our system and for transforming a continuous signal into frequency domain. In the digital computers there is computation of DFT and its inverse but they have complexity in computation. For instance, the word length of the input sequence is N and the total number of arithmetic operation required for the computation is proportional to N^2 . If $N=2000$, then required operations are millions. In most of the applications such numbers are prohibitive. So, in 1965 the discovery of Fast Fourier Transform (FFT) was announced by Cooley and Tukey. FFT is one of the most important algorithm used in many applications of DSP such as frequency estimation, communication etc. because this algorithm is efficient and highly elegant. Consider the one of the most basic radix transform i.e. radix-2 transform in which it requires N to be power of 2. Radix transform is used when the number is prime and at that time DFT has a regular pattern and size r . The number r is called the radix of the FFT algorithm.

Equation 5.1 shows the DFT equation:

$$X_p = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi}{N}np} \quad (5.1)$$

Now, split the Equation 5.1 into even and odd parts as shown into Equation 5.2:

$$X_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{N}(2n)p} + \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{N}(2n+1)p} \quad (5.2)$$

$$\text{Where } \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{N}(2n)p} = \text{Even part} \quad (5.3)$$

$$\sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{N}(2n+1)p} = \text{Odd part} \quad (5.4)$$

From Equation 5.4 take $e^{-j\frac{2\pi}{N}p}$ outside the summation as shown in Equation 5.5:

$$e^{-j\frac{2\pi}{N}p} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{(N/2)}np} \quad (5.5)$$

By substitution of Equation 5.5 in Equation 5.2, it can be expressed as:

$$X_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{(N/2)}np} + e^{-j\frac{2\pi}{N}p} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{(N/2)}np} \quad (5.6)$$

The N point data sequence is split into two $N/2$ point data sequence A_p and B_p , corresponding to even-numbered and odd-numbered samples.

$$= A_p + W^p B_p \quad (5.7)$$

$$\text{Where } A_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{(N/2)}np} \quad (5.8)$$

$$B_p = \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{(N/2)}np} \quad (5.9)$$

$$W^p = e^{-j\frac{2\pi}{N}p} \quad (5.10)$$

Both A_p, B_p are sequence of DFT with length of $N/2$. As we know in frequency domain DFT is periodic but with period $N/2$ there is further simplification. Now take same Equation 5.2 and at frequency $p+N/2$ evaluate it.

$$X_{p+N/2} = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{(N/2)}n(p+N/2)} + e^{-j\frac{2\pi}{N}(p+N/2)} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{(N/2)}n(p+N/2)} \quad (5.11)$$

Now, by simplify terms as follows:

$$e^{-j\frac{2\pi}{(N/2)}n(p+N/2)} = e^{-j\frac{2\pi}{N}np} \quad (5.12)$$

$$\text{And } e^{-j\frac{2\pi}{N}(p+N/2)} = e^{-j\frac{2\pi}{N}p} \quad (5.13)$$

Hence, after simplification put Equation 5.12 and 5.13 in Equation 5.11, which is expressed as:

$$X_{p+N/2} = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} e^{-j\frac{2\pi}{N/2}np} - e^{-j\frac{2\pi}{N}p} \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} e^{-j\frac{2\pi}{N}np} \quad (5.14)$$

$$= A_p - W^p B_p \quad (5.15)$$

Now compare the equation for $X_{p+N/2}$ with that for X_p :

$$X_p = A_p + W^p B_p \quad \text{and} \quad X_{p+N/2} = A_p - W^p B_p$$

This defines the FFT butterfly structure:

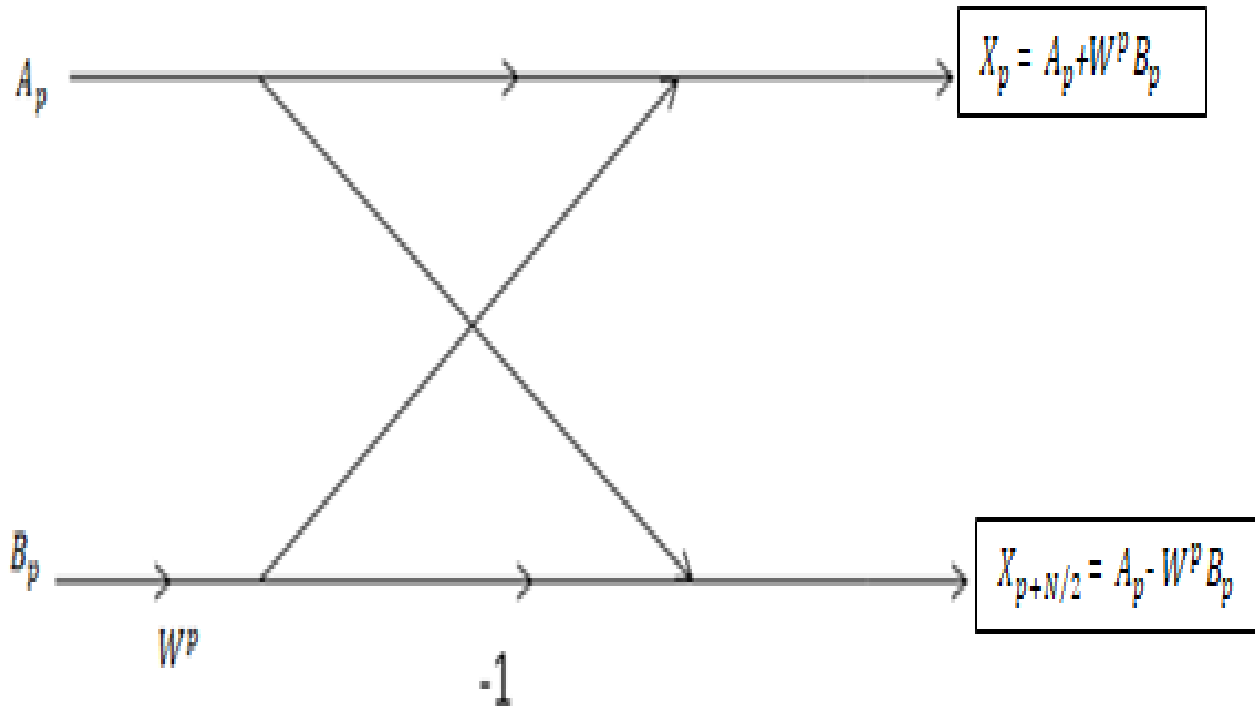


Figure 5.9: Butterfly Structure

So, the complex number of adders and multiplier required is $N/2$ for both A_p and B_p . The total for all $p = 0, 1, \dots, N/2 - 1$ is then $2(N/2)^2$ multiplies and additions for the calculation of all the A_p and B_p . Thus the total number of adders and multipliers required is $N^2/2$ for word length N . As compared to direct DFT the computation is approximately halved. There are different types of FFT such as DIT (decimation in time) and DIF (decimation in frequency don't have that much advantage). Other types are "radix-2" FFT and "radix-4" FFT, in radix-2 there is 2 input-output butterflies and word length N has power of 2 and in radix-4 there is 4 input-output butterflies. In FFT a completely different type of algorithm is used i.e. Winograd Fourier Transform Algorithm (WFTA) in which lengths of FFT is equal to the product of mutually prime factor. WFTA has same length as FFT but it uses more adders and less multipliers. In signal processing application FFT algorithm is most widely used algorithm and the butterfly structure shown in Figure 5.9 plays an important role for processing FFT algorithm. So, it is important to implement an efficient butterfly structure.

5.3.1 Butterfly Structure

It is important that there is an efficient algorithm for processing any application. In FFT algorithm the main building block is butterfly and most of the power in FFT processor is consumed by the butterfly block. So, there is a need to implement butterfly structure firstly because this is the main requirement of our processing. The 4 bit and 8 bit butterfly structure are easy to calculate theoretically but when the word length increases to 16 bit, 32 bit it would be difficult for the researchers to calculate the value theoretically. So, the main focus of our project is to implement 16 bit and 32 bit butterfly structure in an efficient way but in this project we have also implemented 4 bit and 8 bit butterfly structure. For the implementation of butterfly structure adders and multipliers plays an important role. It is necessary to design an efficient adders and multiplier so the butterfly structure implemented in an efficient way. In Section 5.1 and 5.2 we have discussed the high speed adder and multiplier. By using these adders and multipliers, the modified architecture of butterfly is proposed. In the butterfly firstly it is important to design an efficient multiplier by using a high speed adder which we already implemented in Section 5.1. In this report butterfly consist of Wallace multiplier, which is implemented using KSA and CLA and different adders implemented. As shown in Table 5.3 and 5.4 the Wallace multiplier using KSA is best suited for signal application because it has less delay. In butterfly we have calculated delay and power for different word length results are shown in Table 5.5. Flow diagram of butterfly shows in Figure 5.10.

From the synthesis report, the performance parameters like area and delay are obtained and from power analyzer power is calculated which is shown in Table 5.5. Delay, area and power calculation for different bits such 4 bit, 8 bit, 16 bit and 32 bit in butterfly structure are calculated.

Table 5.5: Area, Delay and Power calculation of Butterfly Structure

Sr. No.	Design	No. of Slice LUT	No. Of bonded IOB		Delay (ns)		Power Total (W)		Power Delay Product
			I-Buf	O- Buf	Logic Delay	Router Delay	Power IOs	Power Leakage	

1.	4-bit	122	24	26	5.011	6.672	0.050	0.082	1.5421
2.	8-bit	623	48	46	6.233	12.453	0.050	0.082	2.4665
3.	16-bit	2638	96	86	7.049	16.522	0.050	0.082	3.1113
4.	32-bit	10861	192	166	8.062	21.426	0.050	0.115	4.8360

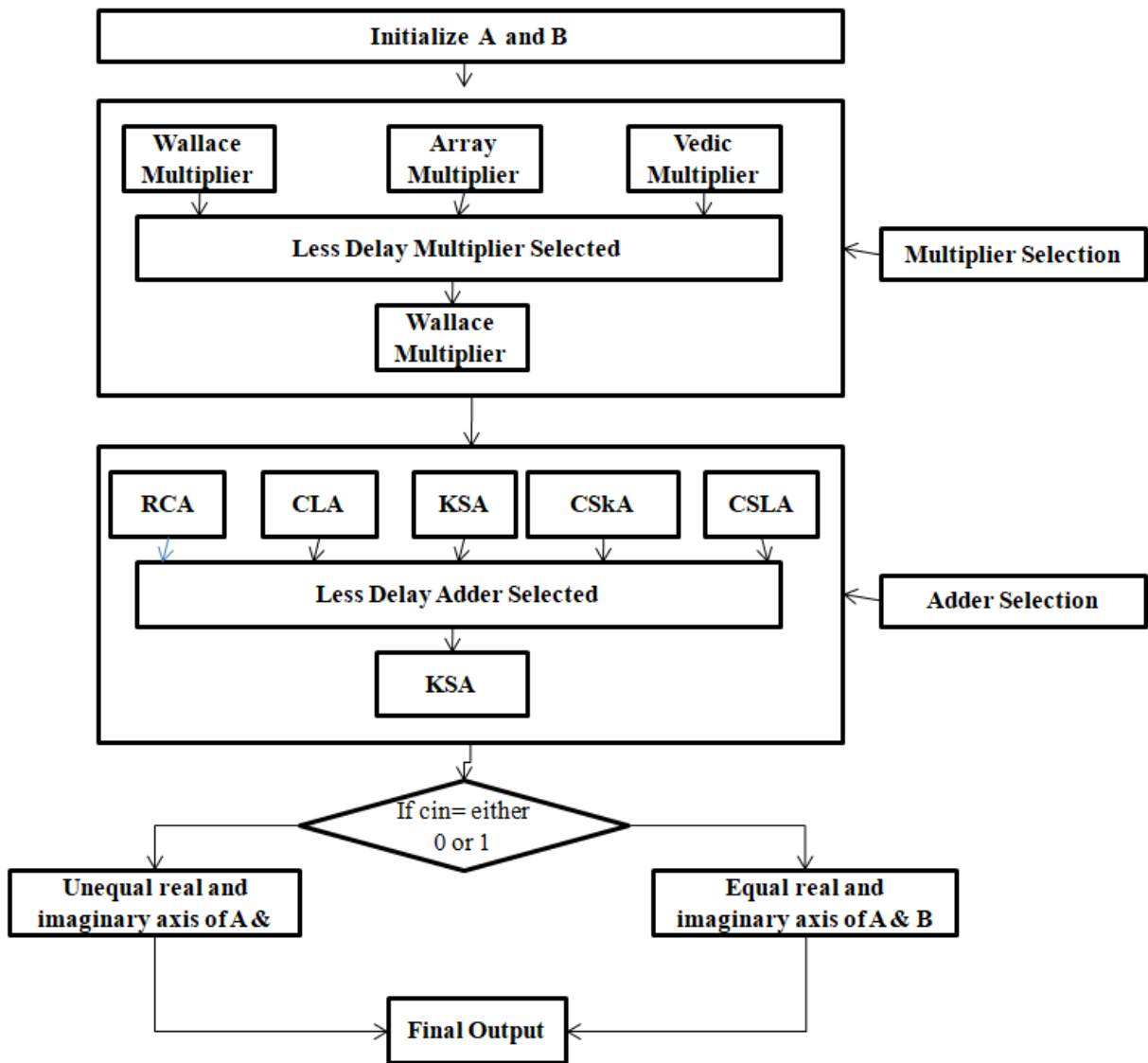


Figure 5.10: Flow Diagram of Butterfly

Conclusion

The performance of any circuit in VLSI design limits by the constituent factors like power, delay and area. In this chapter Wallace multiplier are implemented using KSA and CLA. It is concluded that implementation of Wallace multiplier using KSA have less delay and power as compared to CLA, so it is best suited for implementation of modified Butterfly Structure. The design is tested and verified by Verilog HDL coding and simulation is carried out by Xilinx ISE 14.1 design suite and power is calculated in Xpower Analyzer Future work may be dedicated to decrease in power consumption of butterfly.

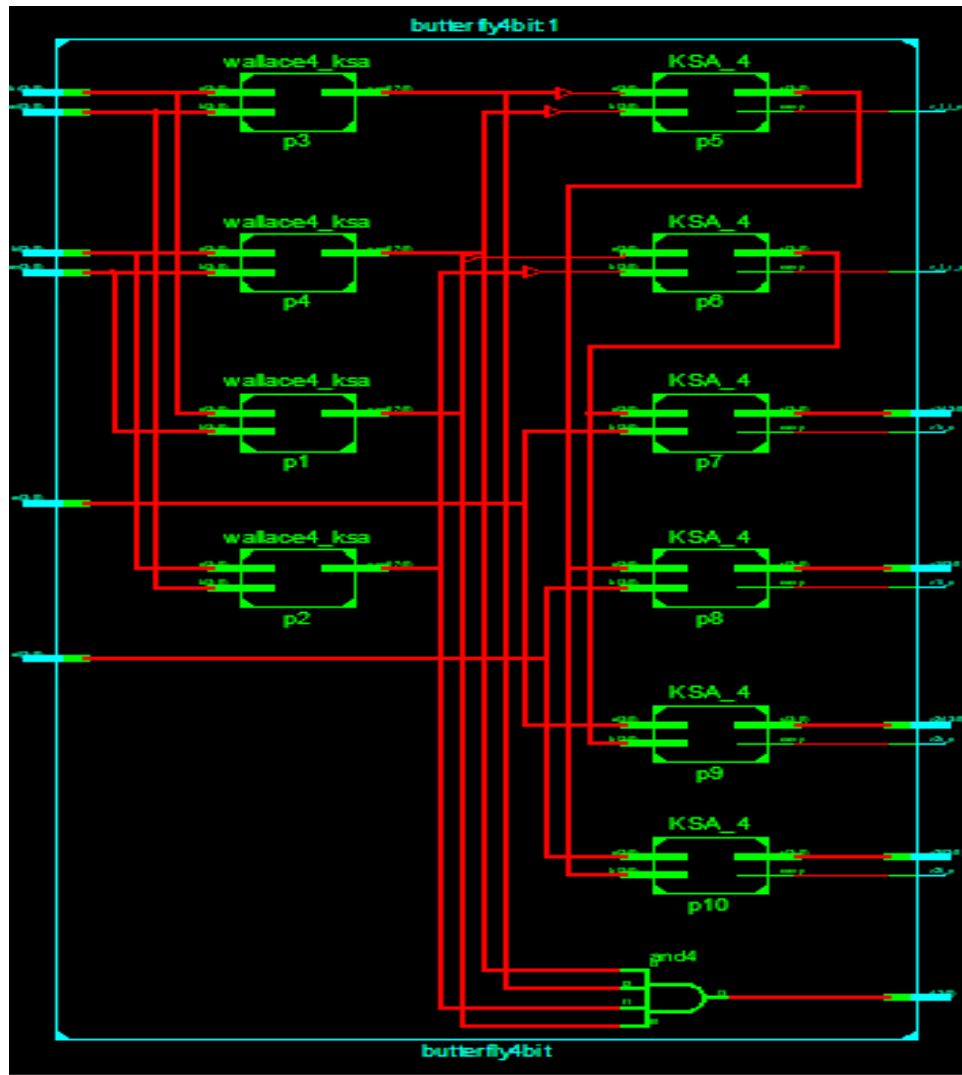


Figure 5.11: RTL Schematic of 4 bit Butterfly

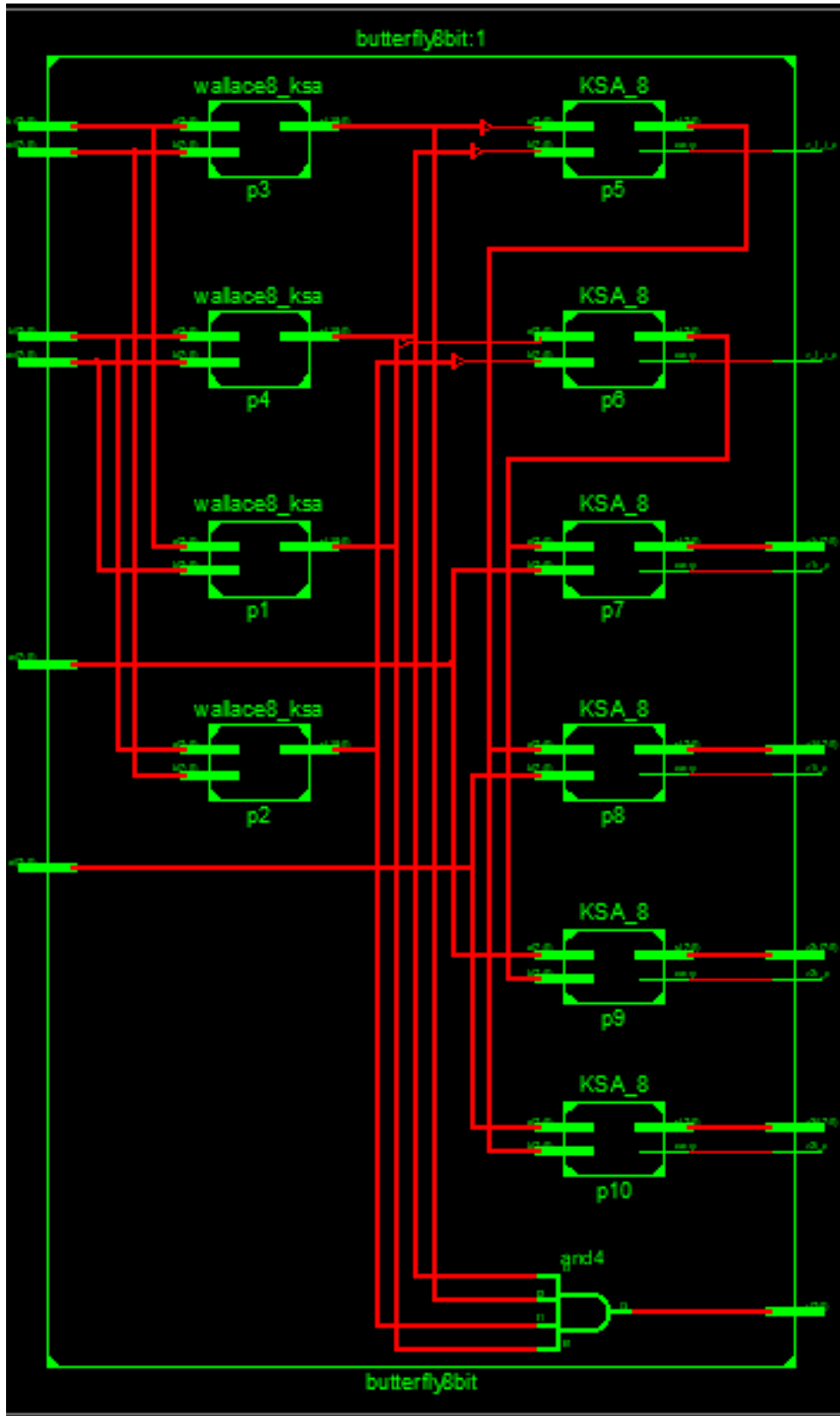


Figure 5.12: RTL Schematic of 8 bit Butterfly

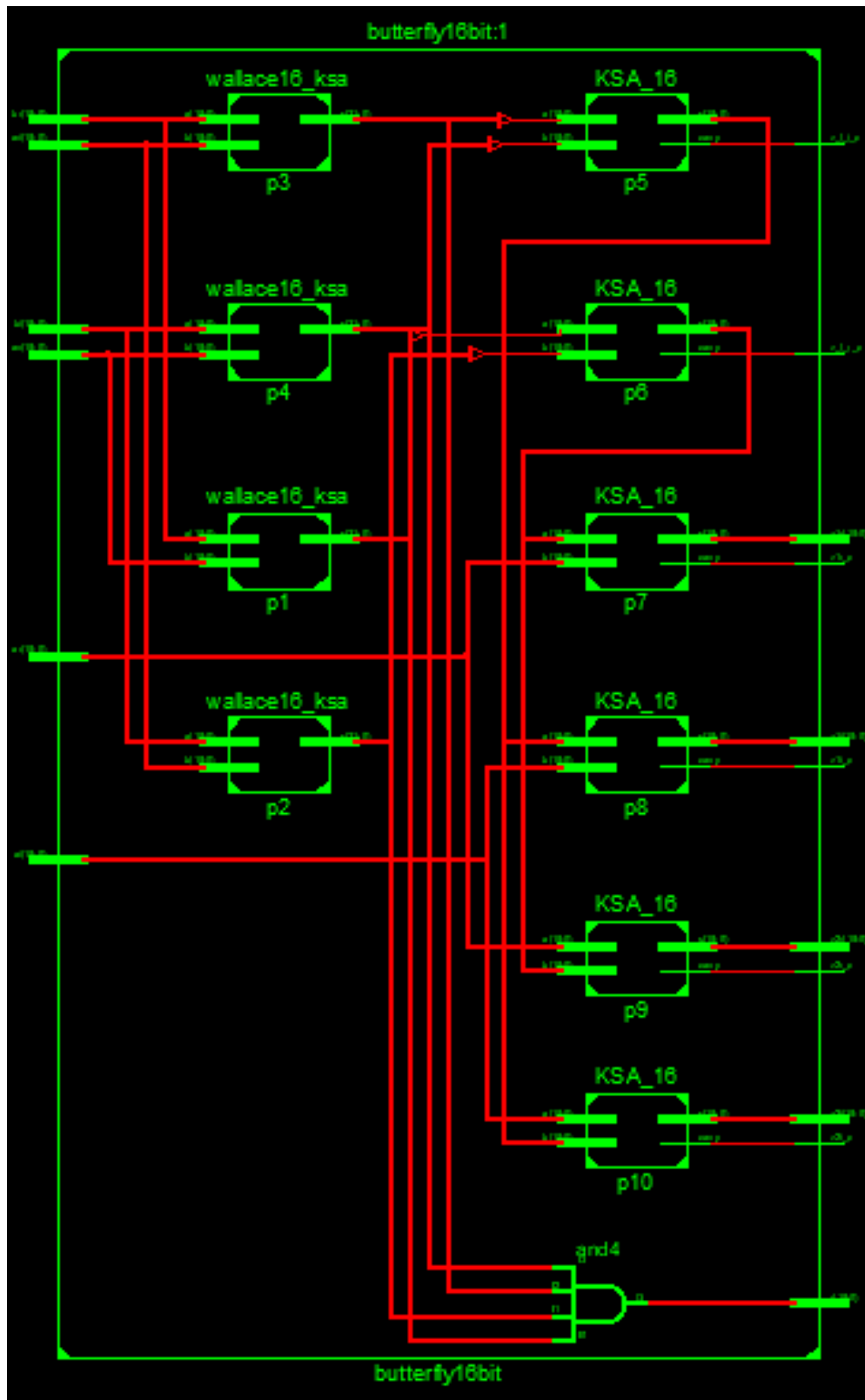


Figure 5.13: RTL Schematic of 16 bit Butterfly

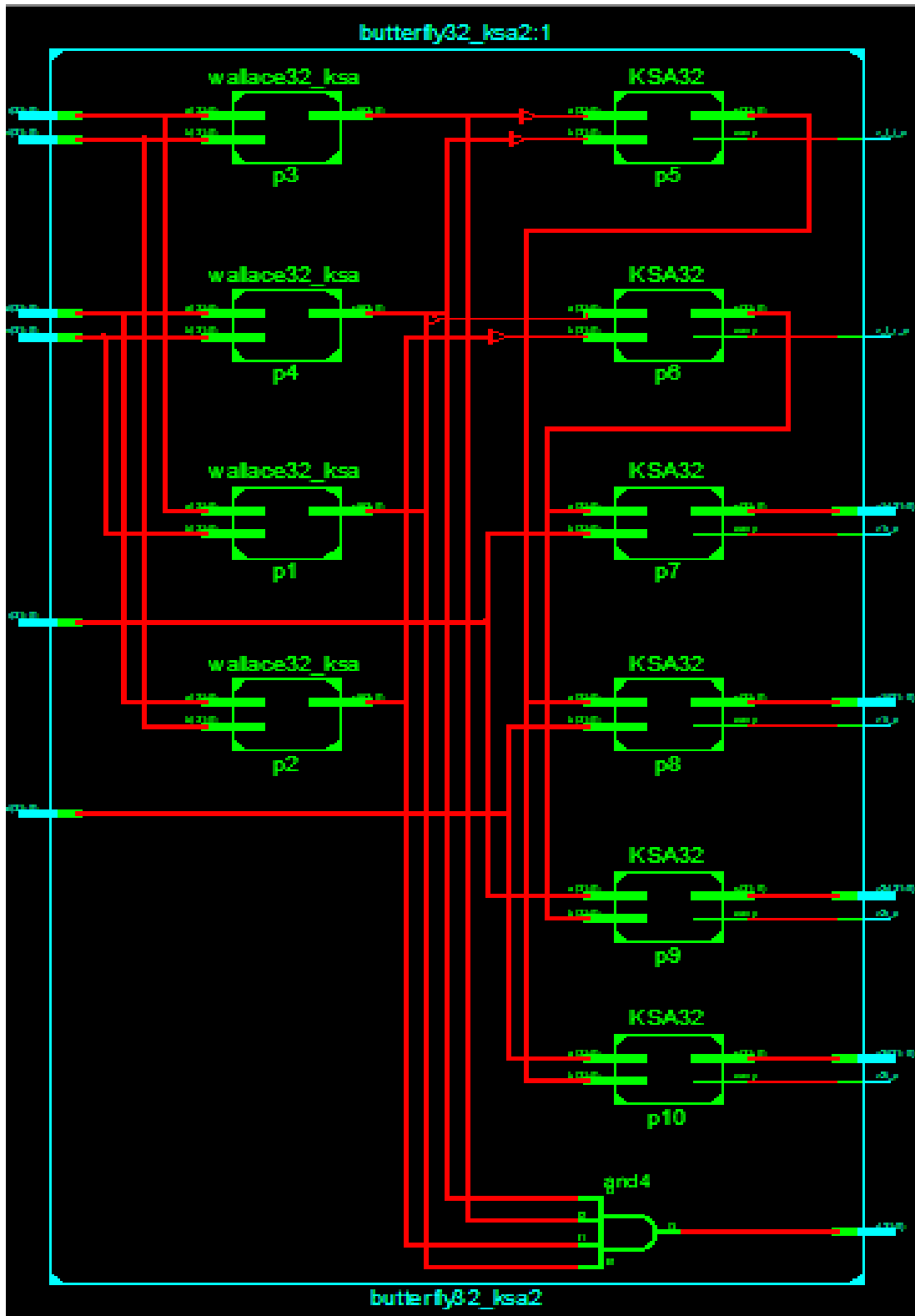


Figure 5.14: RTL Schematic of 32 bit Butterfly

CONCLUSION

The performance of any circuit in VLSI design is limited by the constituent factors like power, delay and area. The designs of various adders and multiplier have been implemented on Xilinx ISE 14.1 design suite and synthesized for Spartan 3E FPGA. Firstly, a modified Carry increment adder is proposed using KSA instead of ripple carry adder. Without affecting the delay performance of the circuit is improved by replacing the 4-bit RCA with a proposed 4-bit KSA. The design is tested and verified by Verilog HDL coding. The delay performance of KSA is better than RCA but as operand size increases (32-bits and above), KSA suffers from complexity due to an increase in the number of logic cells and wiring and proposed CIA_KSA has the disadvantage of more power consumption. Future work may be dedicated to studying the complexity of CIA_KSA when the number of bits is increased. Secondly, a modified Wallace multiplier, Array multiplier and Vedic multiplier are proposed using KSA instead of other adders because KSA has less delay (8.608ns). Out of these multipliers Wallace multiplier has less delay compared to other multiplier but have more power consumption. In Array multiplier and Vedic multiplier decrease in power consumption is proportional to decrease in the speed. In future work we are trying to optimize delay and power, so to design a high speed circuit with low power consumption. Lastly, an efficient algorithm for butterfly processing element implemented. It is implemented by using modified multiplier i.e Wallace multiplier and high speed adder. This butterfly block plays an important role in FFT. FFT block is very important for the computation of other transforms. Application of FFT i.e. in current usage FFT algorithm percentage is large and in Signal Processing application it is widely used. Other examples of FFT are: Coding, Spectrum analysis etc.

REFERENCES

- [1] Shamim Akhter, Vikas Saini, Jasmine Saini, “Analysis of Vedic Multiplier using Various Adder Topologies”, 4th International Conference on Signal Processing and Integrated Networks (SPIN), 2017.
- [2] K V Gowreesrinivas, P.Samundiswary, “Comparative Study on Performance of Single Precision Floating Point Multiplier using Vedic Multiplier and different types of Adders”, International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2016.
- [3] G. R. Gokhale, S. R. Gokhale, “Design of Area and Delay Efficient Vedic Multiplier Using Carry Select Adder”, International Conference on Information Processing (ICIP), Dec 16-19, 2015.
- [4] S.Murugeswari, S.Kaja Mohideen, “Design of Area Efficient and Low Power Multipliers using Multiplexer based Full Adder”, 2nd International Conference on Current Trends in Engineering and Technology, ICCTET, 2014.
- [5] Anjana.R, Abishna.B, Harshitha.M.S., Abhishek.E, Ravichandra V., Suma M. S., “Implementation of Vedic Multiplier using Kogge-Stone Adder”, International Conference on Embedded Systems - (ICES), 2014.
- [6] S. Rajaram, K. Vanithamani, “Improvement of Wallace multipliers using Parallel prefix adders”, International Conference on Signal Processing, Communication, Computing and Networking Technologies (ICSCCN), 2011.
- [7] R. B. S. Kesava, B. L. Rao, K. B. Sindhuri & N. U. Kumar, “Low Power And Area Efficient Wallace Tree Multiplier Using Carry Select Adder With Binary To Excess-1 Converter”, Conference on Advances in Signal Processing (CASP), Jun 9-11, 2016.
- [8] S.Srikanth, I.ThahiraBanu, G.VishnuPriya & G.Usha, “Low Power Array Multiplier Using Modified Full Adder”, 2nd IEEE International Conference on Engineering and Technology (ICETECH), 17th - 18th March, 2016.
- [9] D. Paradasaradhi, M. Prashanthi & N Vivek, “Modified Wallace Tree Multiplier using Efficient Square Root Carry Select Adder”, International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), 2014.

- [10] P. Gurjar, R. Solanki, P. Kansliwal & M. Vucha ,”VLSI Implementation of Adders for High Speed ALU”, International journal of Computer Applications, Vol. 29-No.10, September 2011.
- [11] Bais K. & Ali Z., “Comparison of various adder designs in terms of delay and area”, International journal of science and research (IJSR), Volume 5, Issue 5, May 2016.
- [12] M. Nandini & A. Jayavani, “High Speed and Power Optimized Parallel Prefix Modulo Adders using Verilog”, International Journal of Advanced Technology and Innovative Research, Vol.07, Issue.01, January 2015, pp. 0216-0133.
- [13] Kulkarni R. R., “Comparison among different adders”, IOSR journal of VLSI and Signal Processing (IOSR-JVSP), Vol. 5, Issue 6, Ver. 1 (Nov-Dec 2015), pp. 01-06.
- [14] A. Mitra, A. Bakshi, B. Sharma & N. Didwania, “Design of a High Speed Adder”, International journal of Scientific and Engineering Research, Vol. 6, Issue 4, April 2015.
- [15] A. Kumar & D. Sharma, “Performance Analysis of Different types of Adders for High Speed 32 bit Multiply and Accumulate Unit”, International journal of Engineering Research and Applications(IJERA),Vol. 3, Issue 4, Jul-Aug 2013, pp.1460-1462.
- [16] M. Saikumar & P. Samumdiswary, “Design and performance analysis of various adders using verilog”, International journal of computer science and mobile computing, Vol. 2, Issue 9, September 2013, pp. 128-138.
- [17] C. Suba, S. Karthick & M. Prakash, “Analysis of different bit Carry look ahead adder with reconfigurability in low power VLSI using Verilog code”, International journal of innovative research in computer and communication engineering, Vol. 2, Issue11, November 2014.
- [18] R. UMA, V. Vijayan, M. Mohanapriya & S. Pau, “Area,Delay and Power Comparison of Adder Topologies”, International Journal of VLSI design & Communication Systems (VLSICS), Vol.3, No.1, February 2012.
- [19] S. R. Sahoo & K. K. Mahapatra, Design of Low Power and High Speed Ripple Carry Adder using Modified Feed through Logic, Proceedings of IEEE International Conference on Communications, Devices and Intelligent Systems, West Bengal, pp.377-380, June 2012.

- [20] V. R. Hakki, "Design and Implementation of 4-bit Carry Skip Adder Using NMOS Pass Transistor Logic", *International Journal of Computer Science and Mobile Computing*, Vol.6 Issue.7, July- 2017, pp. 203-207.
- [21] J. Kaur & P. Kumar, "Analysis of 16 & 32 Bit Kogge Stone Adder Using Xilinx Tool", *Journal of Environmental Science, Computer Science and Engineering & Technology*, Vol. 3 .No. 3, June-August 2014.
- [22] P. C. Kumari & R. Nagendra, "Design of 32 bit Parallel Prefix Adders", *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, Vol. 6, Issue 1, May - Jun 2013, pp. 01-06.
- [23] D. H. K. Hoe, C. Martinez, & S. J. Vundavalli, "Design and Characterization of Parallel Prefix Adders using FPGAs", *43rd Southeastern Symposium on System Theory, IEEE*, pp. 168-172, 2011.
- [24] K. Boddireddy, B. P. Kumar & C. S. Paidimarry, "Design and Implementation of Area and Delay Optimized Carry Tree Adders using FPGA", *International Conference on Computers and Communications Technologies, IEEE*, pp. 1-6, Dec. 2014.
- [25] S. K. Yezerla, & B. R. Naik, "Design and Estimation of delay, power and area for Parallel prefix adders", *IEEE Proceedings of 2014 RA ECS, UIET, Panjab University, Chandigarh*, pp. 1-6, March 2014.
- [26] B. K. Mohanty & S. K. Patel, "Area-Delay-Power Efficient Carry-Select Adder", *IEEE Transaction on circuits and systems-II*, Vol. 61, No. 6, June 2014.
- [27] D. Li, "Minimum Number of Adders for Implementing a Multiplier and Its Application to the Design of Multiplierless Digital Filters", *IEEE Transaction on circuits and systems-II: Analog and Digital signal Processing*, Vol. 42, No. 7, July 1995.
- [28] A. K. Singh & A. Nandi, "Design of Radix 2 Butterfly Structure using Vedic multiplier and CLA on Xilinx", *Proc. IEEE Conference on Emerging Devices and Smart Systems*, 3-4 March 2017.
- [29] M. J. Rashmi, G. S. Biradar & M. Patil, "Efficient VLSI Architecture using DIT-FFT Radix-2 and Split Radix FFT Algorithm", *International Journal for Technological Research in Engineering*, Vol. 1, Issue 10, June 2014.
- [30] A. Mankar, *FPGA Implementation of Fast Fourier Transform Core using NEDA*, Mtech Thesis, National Institute of Technology, Rourkela, 2011-2013.

- [31] M. Patrikar & V. Tehre, "Design and Power Measurement of 2 and 8 Point FFT using Radix-2 Algorithm for FPGA Implementation", IOSR Journal of VLSI and Signal Processing, Vol. 7, Issue 1, Jan-Feb 2017, PP 44-48.
- [32] J. G. Proakis & D. G. Manolakis, "Digital Signal Processing Principle, Algorithms, and Applications", 4th Edition, 2007, Prentice Hall India Publications.
- [33] A. Thomas, A. Jacob, S. Shibu & S. Sudhakaran, " Comparison of Vedic Multiplier with Conventional Array and Wallace Tree Multiplier", International Journal of VLSI System Design and Communication Systems, Vol. 04, Issue. 04, April 2016.
- [34] A. Maiti , K. Chakraborty, R. Sultana & S. Maity, "Design and implementation of 4-bit Vedic Multiplier", International Journal of Emerging Trends in Science and Technology, Vol. 03, Issue 05, Pages 3865-3868, May 2016.
- [35] Moore's law. [online] Available at: https://en.wikipedia.org/wiki/Moore%27s_law [Accessed 2 May 2018].
- [36]SemiEngineering.com [online] Available at: http://semiengineering.com/kc/knowledge_center/power-trends/107 [Accessed 2 May 2018].

PUBLICATION

[1] G. Thakur, H. Sohal & S. Jain, “An Efficient Design of 8-bit High Speed Parallel Prefix Adder”, Research Journal of Science and Technology, Vol. 10, Issue 2, May 2018.

**JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
LEARNING RESOURCE CENTER**

PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: Garima Thakur Department: Electronics and Communication

Enrolment No. 162005 Registration No. _____

Phone No. 8054364055 Email ID. garimathakur1994@gmail.com

Name of the Supervisor: Dr. Shanti Jain, Dr. Harsh Sahal

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): High Speed Radix-2 Butterfly Structure using Novel Wallace Multiplier

Kindly allow me to avail Turnitin software report for the document mentioned above.

Garima
(Signature)

FOR ACCOUNTS DEPARTMENT:

Amount deposited: Rs. 500/- Dated: 8/5/18 Receipt No. 201805/204
(Enclosed payment slip) Five hundred only

[Signature]
(Account Officer)

FOR LRC USE:

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Report delivered on	Similarity Index in %	Submission Details	
			Word Counts	Character Counts
<u>08/05/2018</u>	<u>08/05/2018</u>	<u>8%</u>	<u>15,405</u>	<u>75,756</u>
			<u>86</u>	<u>1.72M</u>
			Page counts	File Size

Checked by [Signature]
Name & Signature

[Signature]
Librarian

LIBRARIAN
LEARNING RESOURCE CENTER
Jaypee University of Information Technology
Waknaghat, Distt, Solan (Himachal Pradesh)
Pin Code: 173223

HIGH SPEED RADIX-2 BUTTERFLY STRUCTURE USING NOVEL WALLACE MULTIPLIER

ORIGINALITY REPORT

8%

SIMILARITY INDEX

5%

INTERNET SOURCES

5%

PUBLICATIONS

5%

STUDENT PAPERS

PRIMARY SOURCES

- 1 Submitted to Jaypee University of Information Technology
Student Paper 1%
- 2 S. Vassiliadis. "Binary multiplication based on single electron tunneling", Proceedings 15th IEEE International Conference on Application-Specific Systems Architectures and Processors 2004, 2004
Publication 1%
- 3 www.slideshare.net
Internet Source 1%
- 4 ethesis.nitrkl.ac.in
Internet Source 1%
- 5 Gokhale, G. R., and S. R. Gokhale. "Design of area and delay efficient Vedic multiplier using Carry Select Adder", 2015 International Conference on Information Processing (ICIP), 2015.
Publication 1%

J. Chandra
LIBRARIAN

LEARNING RESOURCE CENTER
Jaypee University of Information Technology
Wazirpur, Dist. Patna