A Major Project Report

On

**IMPLEMENTATION OF IMPLEMENTATION OF SYSTOLIC ARCHITECTURE IN MAC UNIT-VLSI**

Submitted in partial fulfilment of the requirements for the award of the

Degree of **Bachelor of Technology**

In

**Department of Electronics and Communication Engineering**

|  |  |
| --- | --- |
|  | By |
| **K. Kalyan Shanker** | **17241A04Q8** |
| **Ch. Shiva Kumar** | **18245A0405** |
| **D. Naresh** | **18245A0406** |
| **G. Arvind Raj** | **18245A0420** |
| **J. Raju** | **18245A0425** |
| **K. Krishna Vyas** | **18245A0426** |
| **R. Kranthi Kumar** | **18245A0450** |
| **V. Akhil Reddy** | **18245A0458** |
| **Y. Hemanth** | **18245A0459** |

Under the Esteemed guidance of

**Dr. V. Sreehari**



**Department of Electronics and Communication Engineering**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING**

**AND TECHNOLOGY**

**(Approved by AICTE, Autonomous under JNTUH, Hyderabad)**

**Bachupally, Kukatpally, Hyderabad-500090**

i

**Department of Electronics and Communication Engineering**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING**

**AND TECHNOLOGY**

**(Approved by AICTE, Autonomous under JNTUH, Hyderabad)**



**CERTIFICATE**

This is to certify that the major project entitled “**FPGA IMPLEMENTATION OF ARITHMETIC OPERATIONS”** is submitted by **G. Arvind Raj (18245A0420) ,V.Akhil Reddy(18245A0458) , Y. Hemanth(18245A0459) , R. Kranthi Kumar(18245A0450) , K. Krishna Vyas(18245A0426) , D. Naresh(18245A0406) , J. Raju(18245A0425) and Ch. Shiva Kumar(18245A0405)**in partial fulfilment ofthe award of degree in BACHELOR OF TECHNOLOGY in Electronics and Communication Engineering of the Jawaharlal Nehru Institute of Technological University, Hyderabad during academic year 2020-2021, is a bonafied record of work carried out under our guidance and supervision.

**(Guide)(External Examiner)**

**(Head of the Department)**

**Dr. V. Sreehari**

**Dr. N. Swetha**

**(Professor)**

ii

**ACKNOWLEDGEMENT**

We would like to express our gratitude to the outstanding group of people you have helped us in completing our project successfully. We would like to convey our sincere thanks to them for their unabated support rendered at all times.

Firstly, we would also like to thank our college administration mainly Dr. J. Praveen (PRINCIPAL, GRIET) and Dr. J. N. Murthy (DIRECTOR, GRIET) for providing us with such world class infrastructures.

Then, we like to thank Prof. Dr.N. Swetha (Professor and HOD), undue support and guidance given to us during the course of our project.

We would like to thank our project guide DR. V. Sreehari for guiding us through the right path to the completion of our project.

We would like thank all the faculty and our friends for the help and constructive criticism during our project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve higher goals. We have no words to express our gratitude and we are very thankful to our parents who have shown us this world and for every support they gave us.

Finally, we would like to again the outstanding group of people, the supremely positive individuals who encouraged, pushed and supported us throughout the long period of this project.

iii

**DECLARATION**

We hereby declare that the industrial major project entitled **“FPGA IMPLEMENTATION OF ARITHMETIC OPERATIONS”** is the work doneduring the period from **8thmarch 2021 to 07thJune 2021**and is submitted in the partial fulfilment of the requirements for the award of degree of Bachelor of Technology in Electronics and Communication Engineering from Gokaraju Rangaraju Institute of Engineering and Technology (Autonomous under Jawaharlal Nehru Technology University, Hyderabad).The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

**K. Kalyan Shanker (17241A04Q8)**

**Ch. Shiva Kumar (18245A0405)**

**D. Naresh (18245A0406)**

**G. Arvind Raj (18245A0420)**

**J. Raju (18245A0425)**

**K. Krishna Vyas (18245A0426)**

**R. Kranthi Kumar (18245A0450)**

**V. Akhil Reddy (18245A0458)**

**Y. Hemanth (18245A0459)**

iv

**ABSTRACT**

Computer architecture is the pivotal design in terms of system performance. The central processing unit in a system performs complex operations. Processing time is the ultimate barrier during computations. The main motto is to improve the speed of operation by implementing Systolic Architecture. Systolic architecture is the arrangement of n number of processing elements in an array fashion.

The processing element consists of a multiplier and addition unit which identifies the key mathematical operations involved in application. The specialized hardware of systolic array arrangement performs simple calculations on large numbers in a parallel fashion. Complex addition units are reconfigured and replaced with Binary tree adders. Thus, proposed model is to implement trade-off between delay, energy and area. A matrix multiplication and addition operation is performed as a computation in this project. This architecture is implemented on FPGA device which can be reprogrammed for functional validation and Synthesis. This project is aimed to increase processing speed with limited memory and bandwidth.

v

**LIST OF FIGURES**

Figure 2.1 : Half Adder

Figure 2.2 : Full Adder

Figure 2.3 : Ripple Carry Adder

Figure 2.4 : Braun Multiplier

Figure 2.5 : Array Multiplier

Figure 2.6 : Carry Save Multiplier

Figure 3.1 : Generalised N-Operand BTA Structure

Figure 3.2(a): 4 Bit Ripple Carry Adder Binary Tree Adder

Figure 3.2(b): 8 Bit Ripple Carry Adder Binary Tree Adder

Figure 3.3 : 4 Bit Wallace Multiplier

Figure 4.1(a) : 4 Bit Han-Carlson Parallel Prefix Adder

Figure 4.1(b) : 8 Bit Han-Carlson Parallel Prefix Adder

Figure 4.2(a) : 4 Bit Han-Carlson Sum Propagate Adder

Figure 4.2(b) : 8 Bit Han-Carlson Sum Propagate Adder

Figure 5.1 : Architecture of 4 Bit Proposed Adder 1

Figure 5.2 : Architecture of 8 Bit Proposed Adder 1

Figure 5.3 : 4 Bit Proposed Adder 2

Figure 5.4 : 8 Bit Proposed Adder 2

Figure 6.1 : 4 Bit Proposed Multiplier (Wallace-Han-Carlson Multiplier)

Figure 7.1 : 2X2 Proposed Multiplier

Figure 8.1 : 2X2 Matrix Addition

Figure 8.2 : 3X3 Matrix Addition

vi

**LIST OF TABLES**

Table 3.1: Delay Analysis of RCA BTA

Table 5.1: Delay Analysis of Proposed adder 1

Table 5.2: Delay Analysis of Proposed adder 2

Table 6.1: Delay Analysis of Wallace Multiplier and Wallace-Han-Carlson Multiplier

Table 7.1: Delay Analysis of Matrix Multiplier and Matrix Multiplier using HCA

Table 7.2: Delay Analysis of Matrix Addition with RCA BTA and Proposed adder 2

Table 8.1: Delay comparison Table of Different Adders

vii

**CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | **Page No** |
| **CHAPTER 1** |  |  |  |  |
| 1. | INTRODUCTION | | |  |
| 1.1 | | Arithmetic Operation | | 10 |
|  | 1.1.1 | | Binary addition | 10 |
|  | 1.1.2 | | Binary multiplication | 10 |
|  | 1.1.3 | | Binary Subtraction | 10 |
|  | 1.1.4 | | Binary Division | 10 |
| **CHAPTER 2** |  |  |  |  |
| 2. | Classification of Adders and Multipliers | | | 11 |
|  | 2.1 | Binary adders | | 11 |
|  | 2.1.1 | | Half adder | 11 |
|  | 2.1.2 | | Full adder | 12 |
|  | 2.1.3 | | Ripple carry adder | 12 |
|  | 2.1.4 | | Carry Save adder | 13 |
|  | 2.2 | Multipliers | | 14 |
|  | 2.2.1 | | Braun Multiplier | 15 |
|  | 2.2.2 | | Array Multiplier | 16 |
|  | 2.2.3 | | Carry Save Multiplier | 18 |
| **CHAPTER 3** |  |  |  |  |
| 3. Existing adders and Multipliers | | | | 19 |
|  | 3.1 Multi Operand Adder | | | 19 |
|  | 3.1.1 | | Ripple Carry Adder Binary tree adder | 20 |
|  | 3.1.2 | | Delay analysis of Binary Tree Adder | 20 |
|  | 3.2 Wallace Tree Multiplier | | | 21 |

**CHAPTER 4**

|  |  |
| --- | --- |
| 4. Han-Carlson Adder | 24 |

viii

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER 5** |  |  |  |
| 5. Proposed Adder | | | 29 |
|  | 5.1 Proposed Adder 1 | | 29 |
|  | 5.1.1 | Hardware requirements | 29 |
|  | 5.1.2 | Architecture of Proposed adder1 | 29 |
|  | 5.1.3 | Delay Analysis of Proposed adder1 | 30 |
|  | 5.2 Proposed Adder 2 | | 31 |
|  | 5.2.1 | Hardware requirements | 31 |
|  | 5.2.2 | Architecture of Proposed adder2 | 31 |
|  | 5.2.3 | Delay Analysis of Proposed adder2 | 33 |
| **CHAPTER 6** |  |  |  |
| 6. Proposed Multiplier | | | 34 |
|  | 6.1 Wallace – Han-Carlson Multiplier | | 34 |
| **CHAPTER 7** |  |  |  |
| 7. Matrix Multiplication | | | 36 |
| **CHAPTER 8** |  |  |  |
| 8. | Matrix Addition | | 40 |
| **CHAPTER 9** | |  |  |
| 9. | SOFTWARE DESCRIPTION | | 45 |
|  | 9.1 Xilinx Tool | | 45 |
|  | 9.2 Xilinx User Interface | | 45 |
|  | 9.3 Simulation | | 46 |

|  |  |
| --- | --- |
| **CHAPTER 10** |  |
| 10 CONCLUSION | 47 |
| **RESULT** | 48 |

**REFERENCES**

ix

**CHAPTER-1**

**INTRODUCTION**

**1.1 ARITHMETIC OPERATIONS**

Arithmetic operations are basic mathematics that includes the study of numbers that undergoes operations like Addition, Subtraction, Multiplication and, Division. The mathematical operators (+,-,x and, /) are used in our daily life to calculate profits, incomes, annual salaries. The arithmetic operations can be performed in digital electronics on binary numbers which are based on logical operations includes addition, subtraction, multiplication and, division.

It consists of some basic logic gates are known as AND, OR, NOT. These logic gates are used in a combinational logic circuit.

**1.1.1 Binary Addition:**

It involves a basic arithmetic operation that performs binary addition of two numbers which can implement an OR gate. The final product in this operation is considered as the sum. By using the addition operation we can develop half adders and full adders.

**1.1.2 Binary Multiplication:**

It uses a multiplication operation between two binary numbers (n\*n) and the output of this operation includes more bits. This method can be used to implement AND gate.

**1.1.3 Binary Subtraction:**

This method performs subtraction between two binary numbers. This can be developed by two methods. The binary subtract or circuits have a similar implementation to binary addition as half subtractor and full subtractor.

1. Cascaded Full Subtractor
2. 2’s complement

**1.1.4 Binary Division:**

It is one of the main operations in binary arithmetic operations achieved using binary numbers and binary division is close to decimal division and binary subtraction is mostly used in the binary division operation. This method can be explained using the long division method. There are some rules to divide binary numbers those are 0’s and 1’s. A binary division 0 divided by 1 results in 0 and 1 divided by 1 results in 1.

10

**CHAPTER-2**

**Classification of Adders and Multipliers**

Adder is digital circuit Which performs addition operation. Adders are important building blocks of many systems like computers, micro processors, DSP etc, adders are included in Arithmetic Logic Unit (ALU). Intermediate [addresses](https://en.wikipedia.org/wiki/Address_(computing)) in micro processors and micro controllers are calculated using adders.

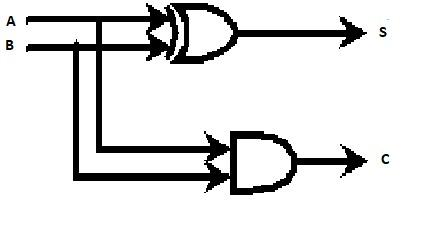
Adders are designed for many representations like Excess 3, BCD, binary but out of all those adders binary adders are most commonly used adders, Binary adders are the one which can perform binary addition.

**2.1 Binary adders:**

**2.1.1 Half adder:**

Half adder is a binary circuit which can perform single bit binary addition, it can add upto two operands that is two binary inputs, Implementation of the half adder is shown in the below figure, it has two inputs A,B and two outputs Sum(S), Carry(C),

Xor gate can calculate Sum(S), And gate can calculate Carry (C). Boolean logic for sum is A’B+AB’ and for carry is AB



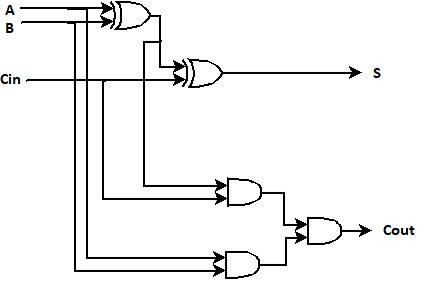
**Fig 2.1: Half adder**

11

**2.1.2 Full adder:**

Full adder is a binary circuit which can perform single bit binary addition, it can add upto three operands that is three binary inputs, Implementation of the Full adder is shown in the below figure, it has three inputs A,B,Cin and two outputs Sum(S), Carry(C).

A full adder can be implemented in many different ways such as with a convention [transistor](https://en.wikipedia.org/wiki/Transistor)-level circuit or composed of other gates. One example functioning is with S = A ⊕ B ⊕ C in and C out = (A ⋅ B) + (C in ⋅ (A ⊕ B)).

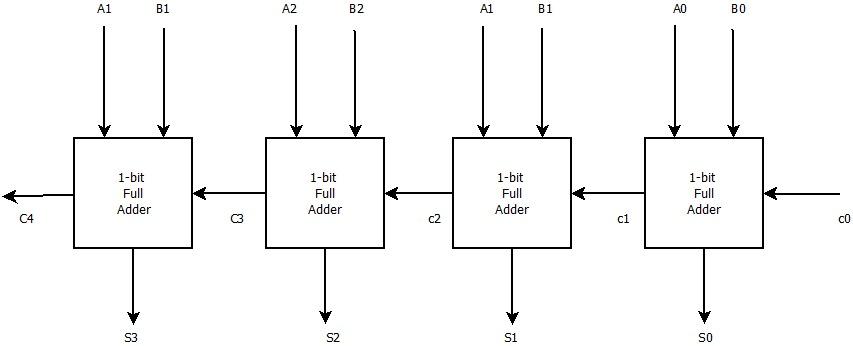


**Fig 2.2: Full adder**

**2.1.3 Ripple carry adder:**

Ripple carry adders are the adders which can perform multi bit addition, Implementation of four bit ripple carry adder is shown in the below diagram, it supports two four bit operands that is it can perform addition of two four bit numbers. In RCA carry output of one full adder is propagated to input of another full adder as shown in the below figure. As carry is propagated from one full adder to another full adder dependency is increased, because of dependency it is consuming more delay.

12



**Fig 2.3: Ripple carry adder**

**2.1.4 Carry save adder:**

If we want calculate sum of three or more numbers, it could be advantageous if carry result is not propagated. As an alternative three input adders are used each adder generate two results that are sum and carry. The sum and carry may be fed into two inputs of the subsequent 3 number adder need not wait for propagation of carry signal. After all stages of addition, however a conventional adder must be used to join the final sum and carry results

13

**2.2 Multipliers:**

A multiplier is an electronic circuit used in many digital applications and it plays a major role in digital signal processing and digital electronics. In high-performance systems like microprocessor uses arithmetic operations in which most of the instructions can be performed and in DSP algorithms by using multiplication and addition operations. It is known as binary multipliers and also a combinational logic circuit in digital electronics.

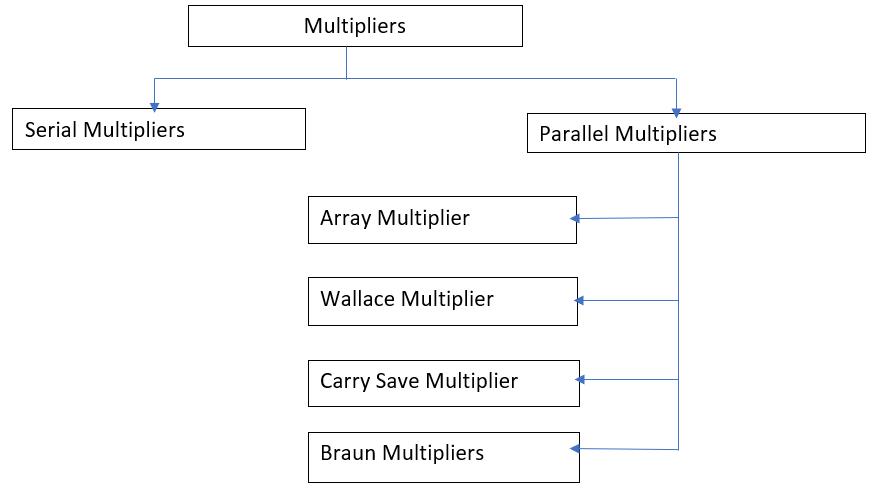
Multiplication in the binary system uses binary numbers such as 0’s and 1’s and multiplication can be performed using product addition and shifting using parallel multipliers. Then the partial products are added to produce the final value. The multiplication process uses n-shifts and adds to multiply n-bits binary numbers. In digital applications, the multiplier design plays an important role as we reduce the number of operations the power consumption becomes less. Based on low power efficient circuit designing the speed of multipliers increases. There are many aspects for multipliers to perform a better operation which depends on speed, power consumption, area and, accuracy.

**Classification:**

The multipliers are classified into two types.

1. Serial multipliers
2. Parallel multipliers

14

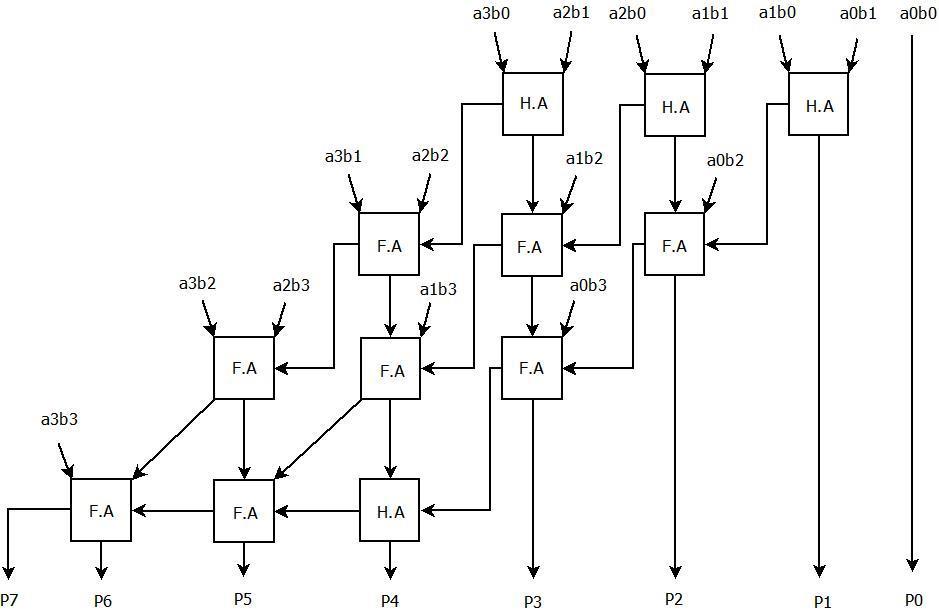


**2.2.1 Braun Multipliers:**

It is a parallel multiplier also known as carry-save array multiplier. The structure of this multiplier consists of an array of AND gates and adders. Braun multipliers can implement by n\*n bit. The product can be generated in parallel with the And gates. Each partial product can be added with the sum of partial product which has previously produced using the row of adders. The carry out is shifted one bit to the left or right side and then it is added to the sum which is generated by the first adder and the partial product. The shifting is carried out with help of a carry-save adder and the ripple carry adder is used as the final stage of output. Brown multiplier has good speed, power and, area in unsigned operations which are less than 16 bits and it has a simple structure when compared with other multipliers.

This multiplier has some disadvantages like flitching problems due to ripple carry adder in the last stage and it also has some delay problems because of full adder in the last stage. These can be minimized by using a parallel prefix adder known as KOGGE STONE ADDER. This adder is a type of Carry Look ahead adder. The structure of Braun multiplier with Kogge stone adder provides fast multiplication.

15



**Fig 2.4 Braun Multiplier**

**2.2.2 Array Multiplier**

Array Multiplier is simple to design which uses adders as basic building blocks and it is based on repeated addition and shifting procedure. Each product is generated by the multiplication of the multiplicand with multiplier digit. The partial product is shifted according to the bit sequences and then added. The summation can be performed with a normal carry propagation adder and we require N-1 adders where n is no of multiplier bits.

To perform array multiplication, we should take two 4 bits numbers like a and b (a1, a2, a3, a4 and b1, b2, b3, b4). The below operation shows that the multiplication between two different numbers which consists of 4 bits each. Then we get the first partial products. The second partial products are generated by multiplying with the second bits from the LSB. As we repeat the same procedure with different bits, we get 4 partial products of the total. To get final product values we should add the columns

16

and the carry is shifted to another adder and performs sum to that column till the last and the carry generated at the MSB is considered as p7.

a3 a2 a1 a0 multiplicand

b3 b2 b1 b0 multiplier



a3b0 a2b0 a1b0 a0b0

a3b1 a2b1 a1b1 a0b1

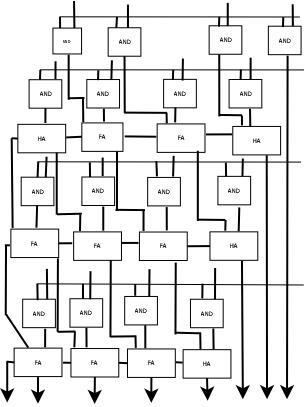
a3b2 a2b2 a1b2 a0b2 partial products

a3b3 a2b3 a1b3 a0b3



p7 p6 p5 p4 p3 p2 p1 p0 final products

**Architecture of Array Multiplier:**



**Fig 2.5 Array Multiplier**

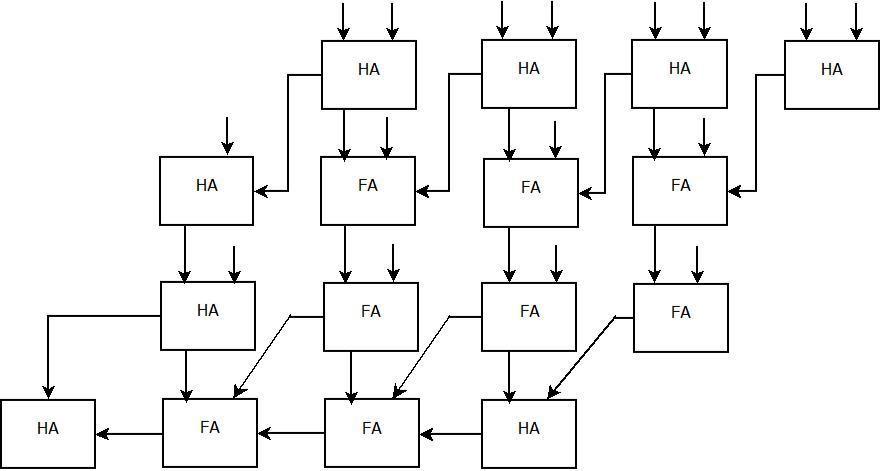
17

In the above diagram, the AND gate performs AND operation of two bits a, b and produces one output. The full adder receives 3 inputs whereas half adder takes 2 inputs to perform their operations and both produce 2 outputs.

**2.2.3 Carry save multiplier:**

The carry-save multiplier has a feature of shifting carry to the next adder is known as the propagation of carry. The implementation carry save multiplier is same as the array multiplier. In this addition process, an adder uses 3 or more n bit numbers. The carry-save multiple adders are fast and easy to understand and it is mainly used to reduce delays with the carry-save addition process. The carry is propagated in a certain path, the output carry of one adder is given as input to the other full adder or half adder and that is the only difference between carry save multiplier and array multiplier. The Structure of the Carry save multiplier consists of half adders and full adders.

**Architecture of Carry save multiplier:**



**Fig 2.6 Carry Save Multiplier**

18

**CHAPTER-3**

**EXISTING ADDERS AND MULTIPLIER**

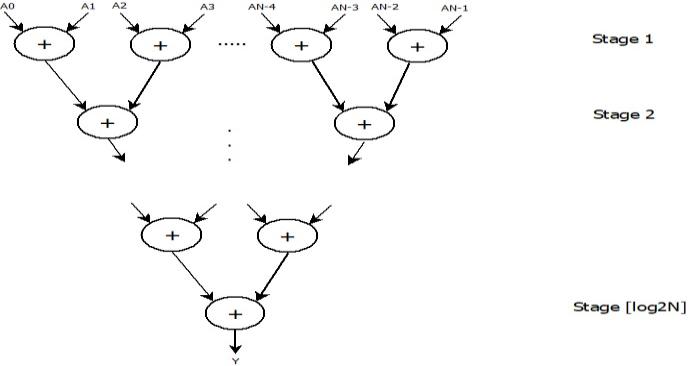
**3.1 Multi operand adders**

The well know adders for partial products of multipliers is Multi-Operand Adders. Binary tree adder (BTA) is simple form of MOAs. BTA employs binary tree structure for multiple operand addition. The generalized block of N-operand BTA is as shown in below Fig 3.1.N operands addition can be calculated using N-1 adder units in ⌈log2N⌉ stages, ceiling operator: - [ . ]

The adder unit parameters are directly proportional to the area, delay and energy efficiency of BTA. BTA can be developed using many adders. Few of them are parallel prefix adder (PPA), ripple carry adder (RCA), carry-select adder (CSLA), where each one of them has its own tradeoffs among area, delay and energy consumption parameters. In comparison with other adders RCA has energy and higher delay with less area. Whereas other are vice versa. In large complex additions area and energy are the two main factors. The BTA employed in these systems should be hardware and energy efficient.

The Applications Areas of MOAs:

* CNN architecture,
* Digital filters,
* Transforms.

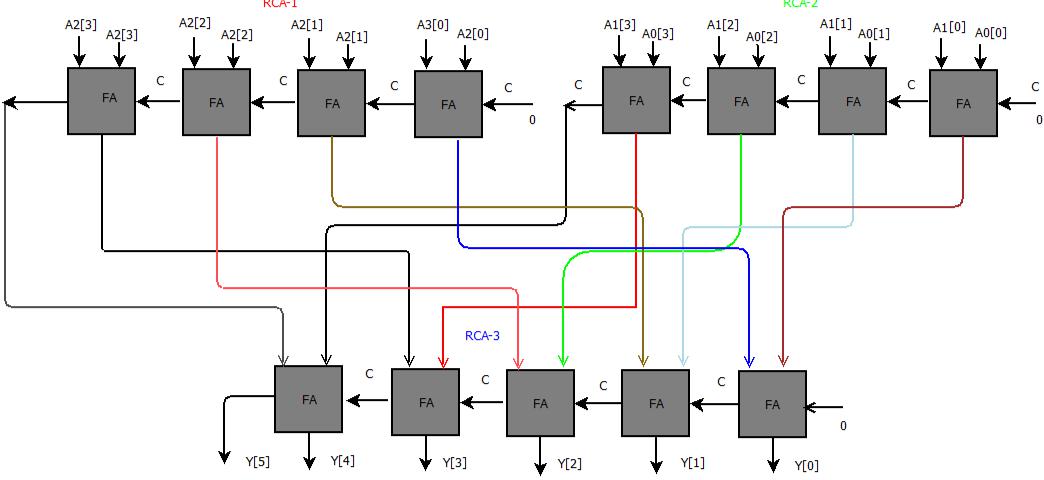


**Fig. 3.1 Generalized N-operand BTA structure**

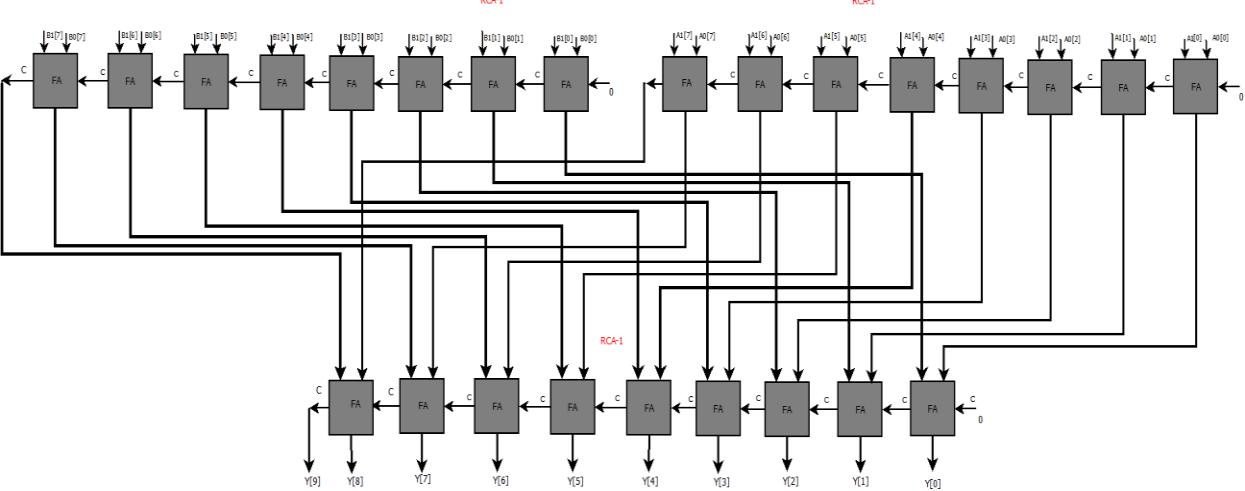
19

**3.1.1 Ripple Carry Adder –Binary Tree adder:**

The four-operand Ripple Carry Adder –Binary Tree adder structure shown in Fig produces sum (*Y*) of four operands (*A*0, *A*1, *A*2 and *A*3) using three RCAs.



**Fig. 3.2(a) 4 bit Ripple Carry Adder –Binary Tree adder**



**Fig. 3.2(b) 8 bit Ripple Carry Adder –Binary Tree adder**

**3.1.2 Delay analysis of Binary Tree Adder:**

The four-operand RCA-BTA structure shown in Fig. It seems that each addition stage introduces one RCA delay. While from the delay analysis:

1-level of Addition of N-bit--------------------------> N-bit RCA time delay;

2-level of Addition ------------------------------------>2 Full Adders time delay.

20

Adder

RCA-BTA 4 bit

RCA-BTA 8 bit

Delay

3.815ns

6.799ns

**Table 3.1 Delay analysis of Binary Tree Adder**

**3.2 Wallace Tree Multiplier:**

Wallace tree multiplier is a multiplier which is used to multiply the two numbers in structural format where this method is used in parallel multiplier architecture.

Normally we use fully designed high speed and low power multipliers for multiplication purposes. We know that during the multiplication process, the partial products will be generated and there are many methods to decrease the partial products and increase the speed, out of them one of the methods is using Wallace tree multiplier. It is the best and useful hardware structure/function available for us which can be used in digital circuits for multiplication of two integers.

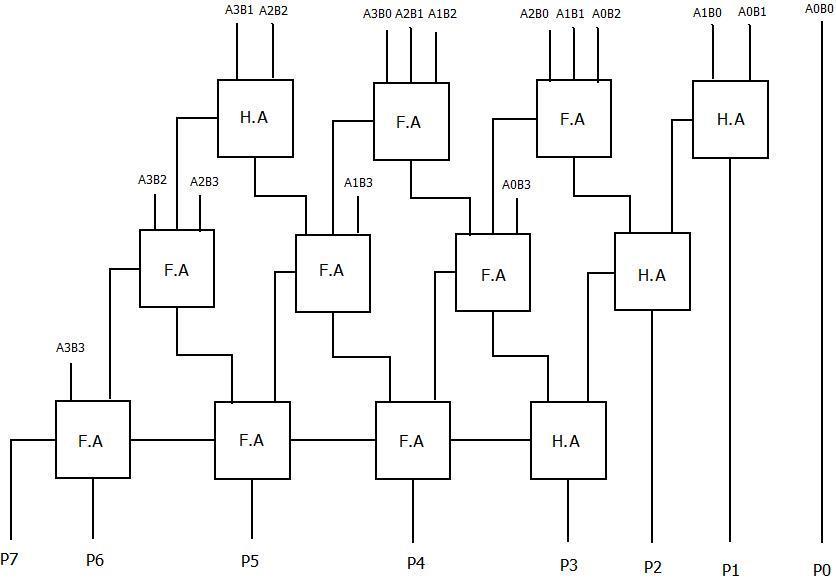
In this Wallace tree multiplication method, multiplication of two numbers is done by decreasing the number of partial products into 2-row matrix by a full adder, half adder and a ripple carry adder / carry save adder and again these two rows are added by using a fast carry propagate adder to give the final output product. Here in this method, we have used a half-adder for adding two bits and for adding 3 bits, we have used a full-adder circuit.

A Wallace tree multiplier is a modified and updated version of a tree-based multiplier architecture. In the below example of 4-bit multiplication using the Wallace tree method, you can see that we used half-adders and full-adders for decreasing the partial products.

21

Generally, we use the “add and shift” algorithm for the multiplication process. But in parallel multipliers the number of partial products which are added in the main parameter determines the performance of the multiplier.

In the example shown below, we used a ripple carry adder in the last stage, is a logic circuit, where the carry output of every full-adder is the carry input of the succeeding next most significant full adder. It is called a ripple carry adder because each bit gets rippled into next stage.

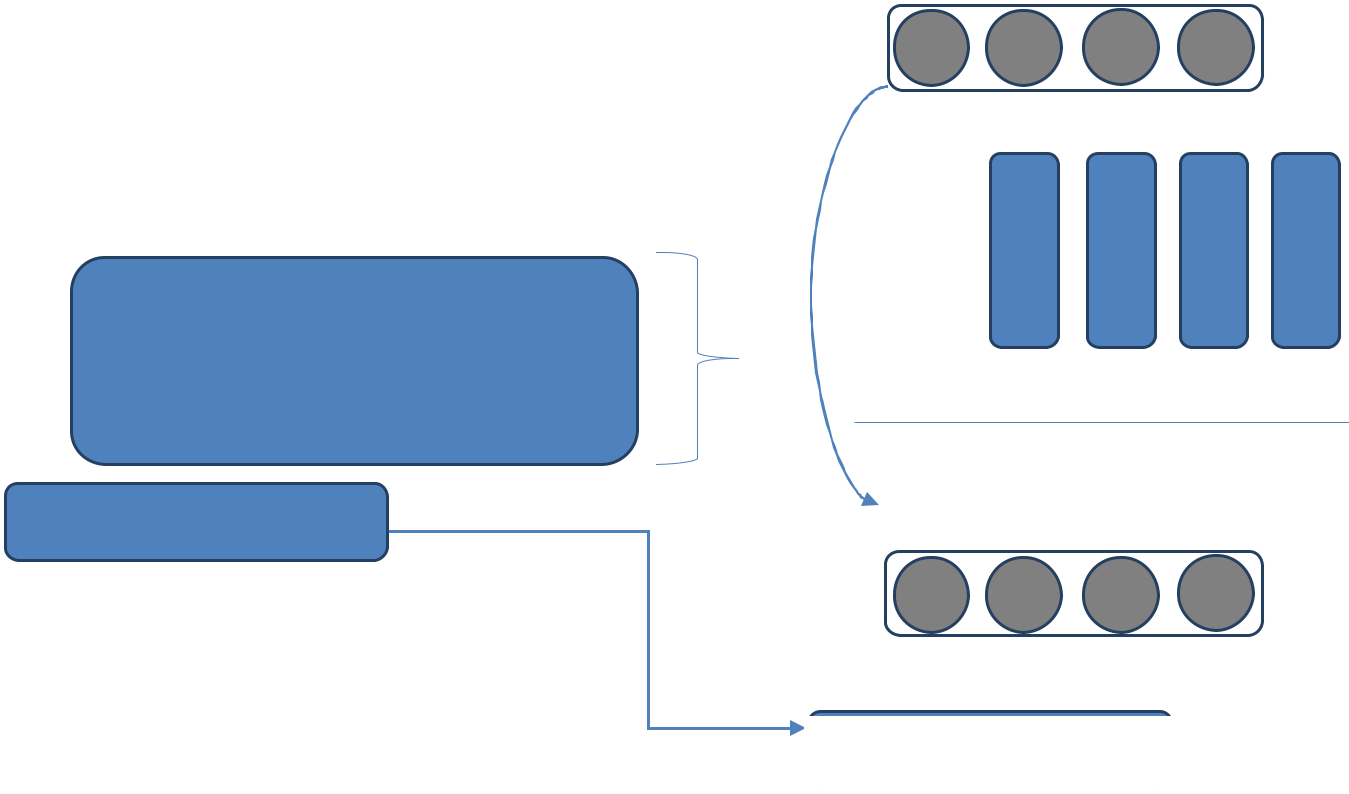


**Fig 3.3: 4-Bit Wallace Multiplier**

The Wallace tree multiplier follows three steps, where in the first stage/step, after multiplying the bits of multiplicand and multiplier, the bit products are formed. In the 2nd stage/step, to lower the number of rows, the bit product matrix is reduced by using the half-adders and full-adders. This process will continue till the last addition remains and finally using ripple carry adder the final addition is done to obtain the result.

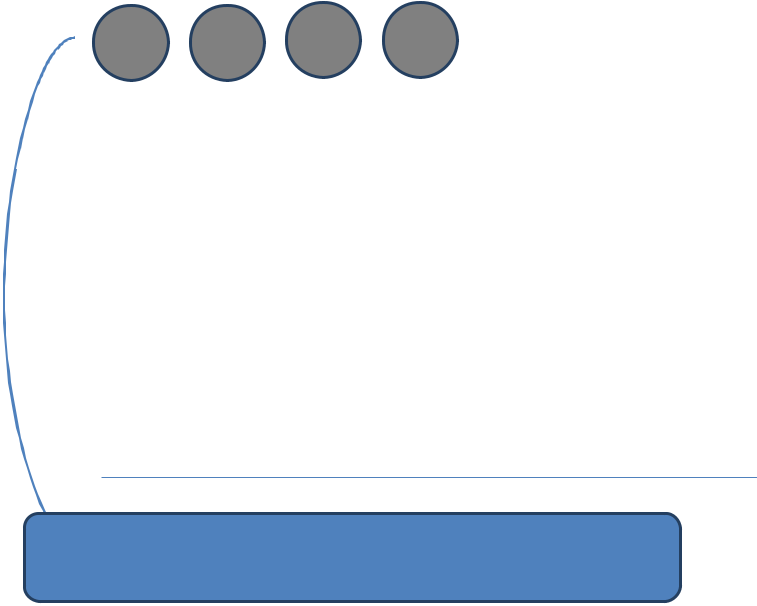
22

**Example of a 4bit Wallace tree multiplier**.



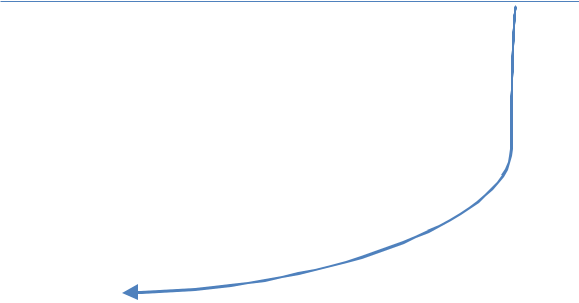
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  | 1 | 1 | 1 | 1 |
|  |  | (x) | 1 | 1 | 1 | 1 |
|  |  |  | ------------------------ | | |  |
|  |  |  | 1 | 1 | 1 | 1 |
|  |  | 1 | 1 | 1 | 1 |  |
|  | 1 | 1 | 1 | 1 |  |  |
| 1 | 1 | 1 | 1 |  |  |  |

------------------------------------------------



Stage 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Carry | 1 | 1 | 1 | 1 |  |
|  |  |  | 1 | 1 | 1 |
|  |  | 1 | 1 | 1 | 1 |
|  | 1 | 1 | 1 | 1 |  |
|  | 1 | 0 | 1 | 1 | 0 |
|  | 1 | 1 | 1 | 1 |  |
| 1 | 1 | 1 | 1 |  |  |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | Stage 2 Carry | | |
| ` | 1 | 0 | 1 | 1 | 0 | 1 |

1 1 1 1

1 1 1 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | Stage 2 |
| 1 | 1 | 1 | 1 |  |  |  |  |



1 1 1 0 0 0 0 1

23

**CHAPTER-4**

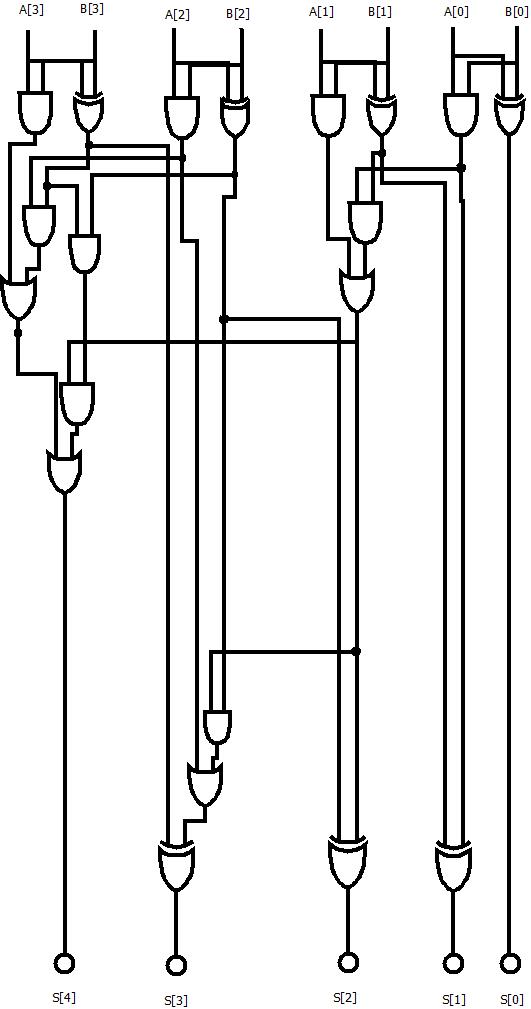
**Han–Carlson Adder**

Adders are very important in ALU. These adders are combined to be used in building of an integrated chip. Generally IC chip are used in the implementation of adders to reduce the area by minimizing the adder in the manufacturing process. If we reduce the number of gates the area occupied is very les also power consumption is also get reduced and speed of proceeding unit increased by decreasing the delay. Here we use parallel prefix adder which increase speed of addition.

Han–Carlson adder is the next version of Kogge-stone. If we consider radix-2 Han-Carlson prefix adder are chosen for the low power analysis prefix adder computation takes in two steps: One to get carry and with that next to compute the sum in the next stage prefix adder.

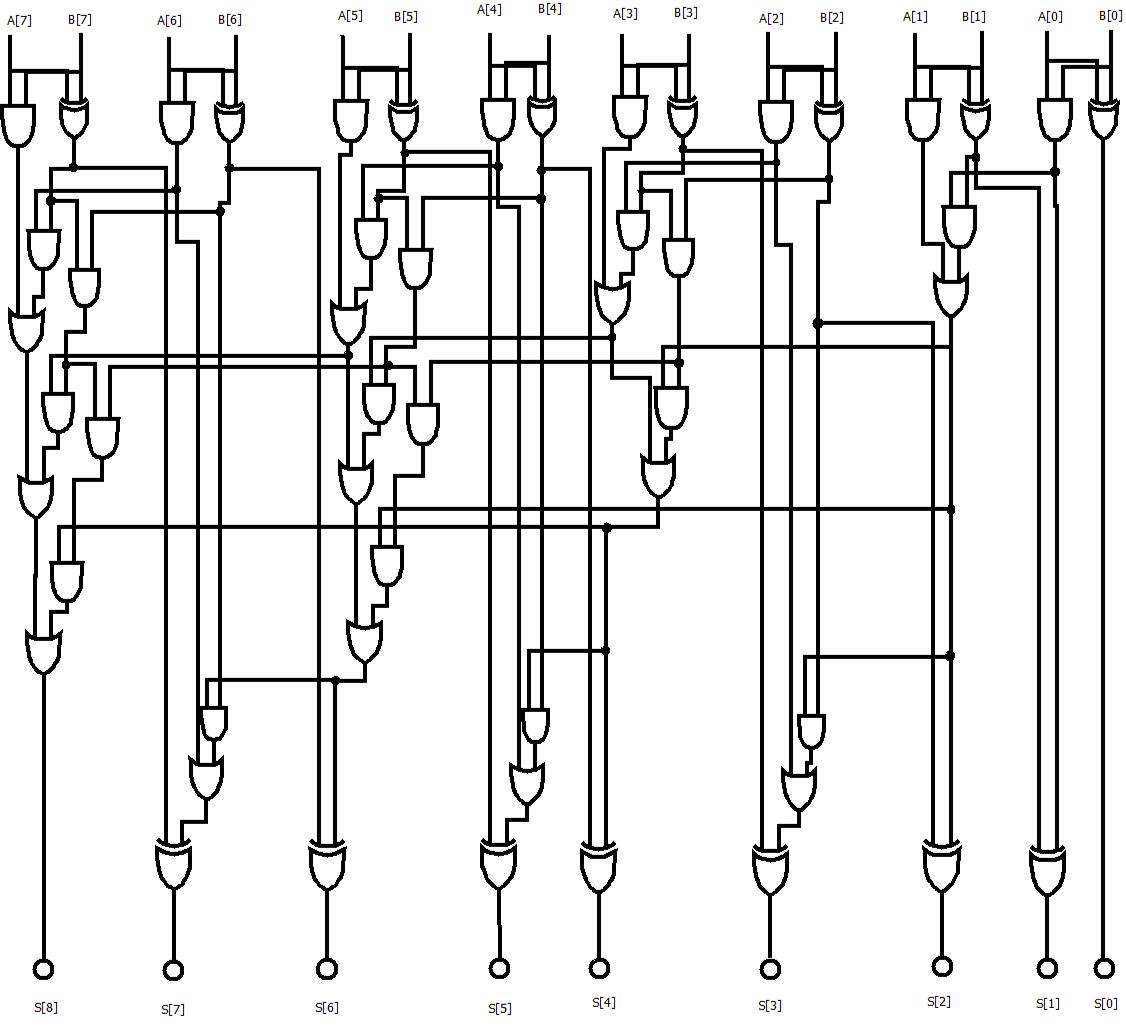
Here we also have hybrid version of Han-Carlson which is combination of Han-Carlson and Kogge-stone. In this delay is very less compared to Han-Carlson. It has log2n stages and Brent-kung construction has (2log2n-1) stages. In the transition level the number of prefix operation are less in Han-Carlson than Kogge-stone .Thus the Han-Carlson reduces the area used by one delay in addition circuitry also reaches the same speed with less power consumption. Han-Carlson has good trade-off between fab-out, number of black cells and number of levels. Because of these reasons we have removed other adder and included Han-Carlson.

24



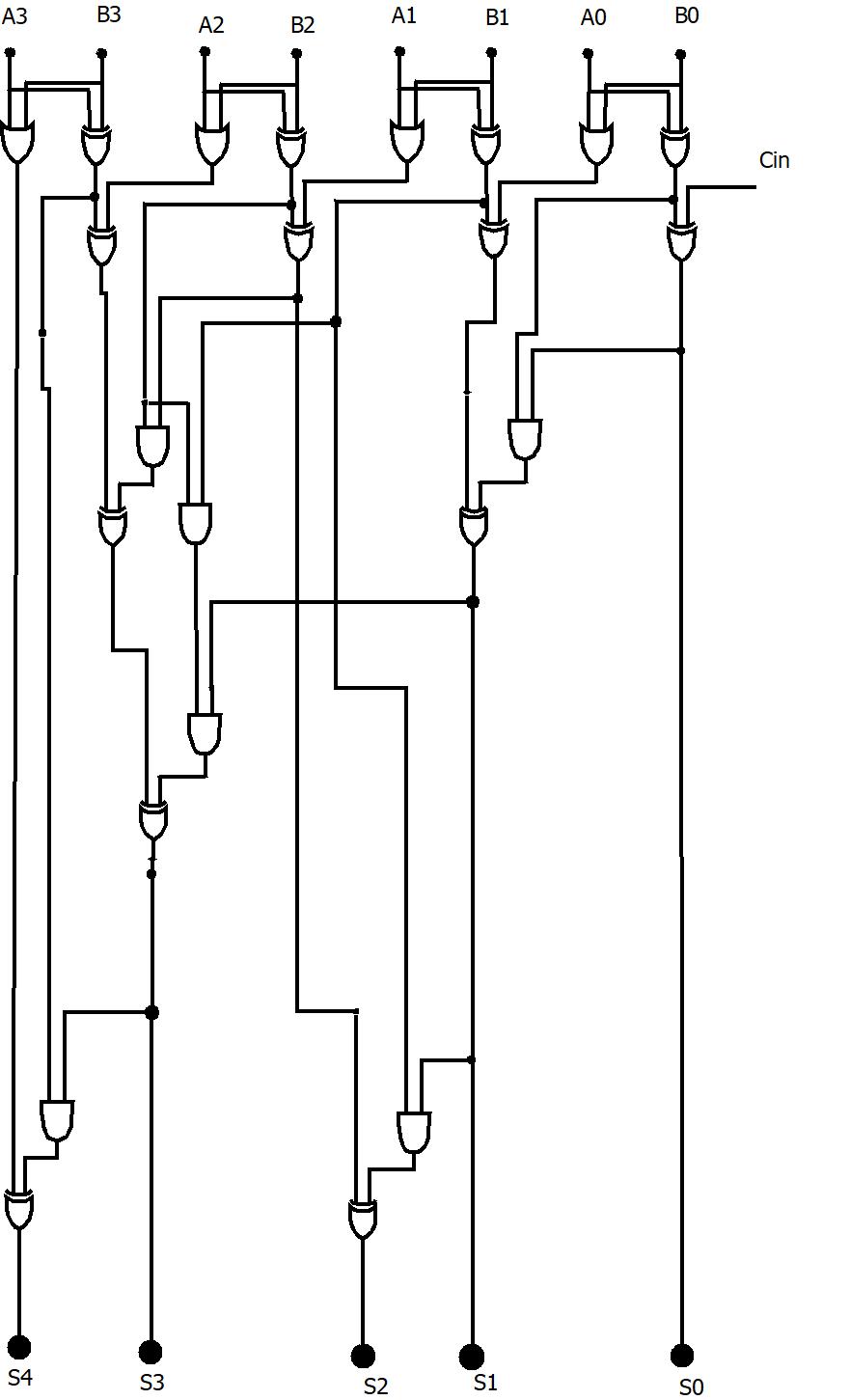
**Fig 4.1(a) 4 bit Han Carlson Parallel Prefix Adder**

25



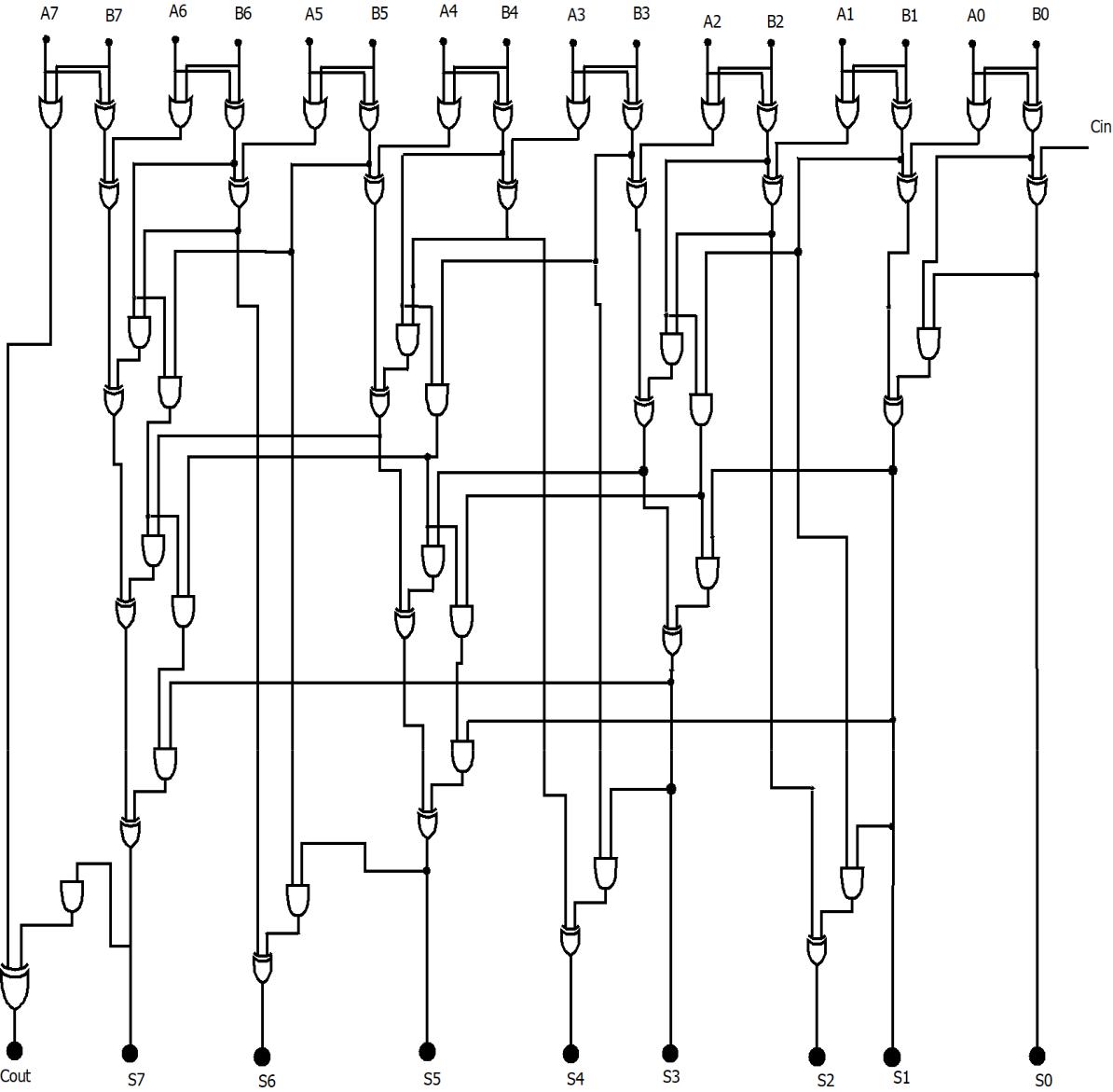
**Fig 4.1(b) 8 bit Han Carlson Parallel Prefix Adder**

26



**Fig 4.2(a) 4 bit Han Carlson Sum Propagate Adder**

27



**Fig 4.2(b) 8 bit Han Carlson Sum Propagate Adder**

28

**CHAPTER-5**

**Proposed Adder**

**5.1 Proposed Adder1:**

**5.1.1 Hardware required:**

For 4 bit adder

11 Full adders,

2 Half adders,

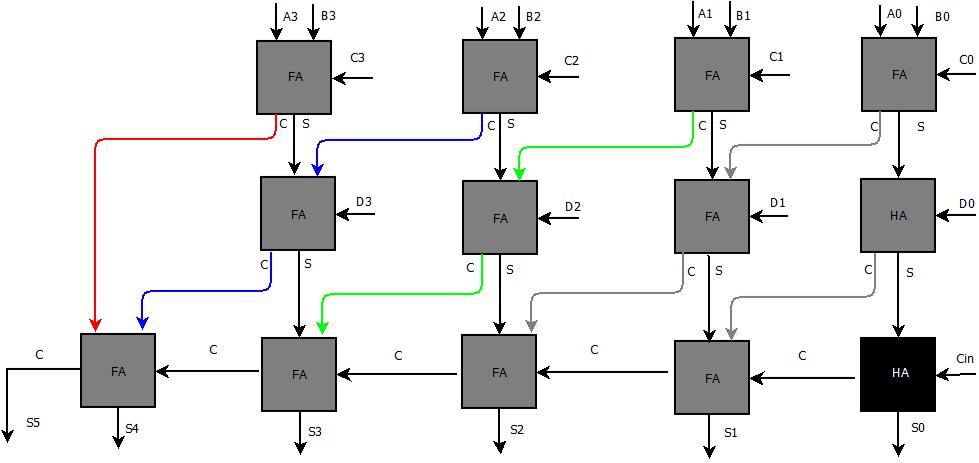
For 8 bit adder,

23 Full adders,

2 Half adders.

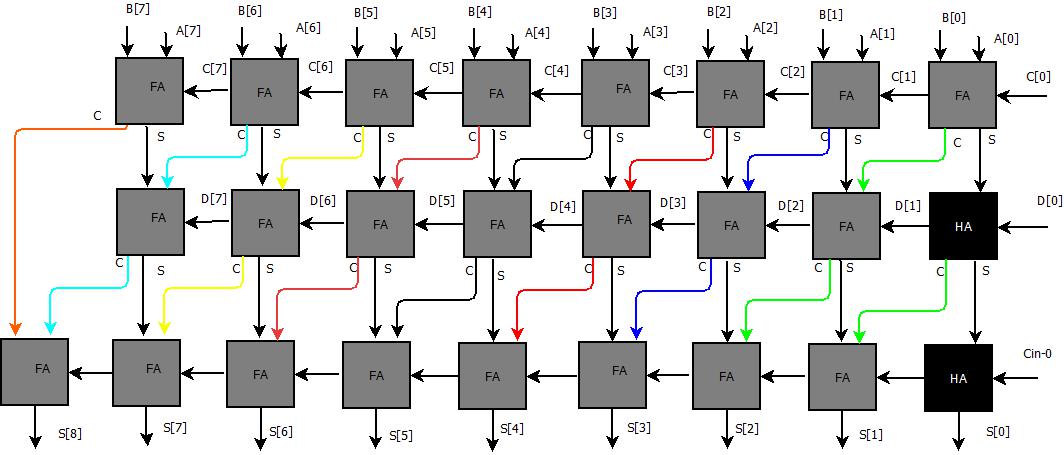
**5.1.2 Architecture of Proposed Adder 1**

Four bit and Eight bit implementations of proposed adder1 are being showed in the following figures



**Fig 5.1 Architecture of 4 bit Proposed Adder 1**

29



**Fig 5.2 Architecture of 8 bit Proposed Adder 1**

Here carries generated in the present stage of adder are routed to next stage full adders present in next place value of the adders as shown in the above figures

**5.1.3 Delay Analysis of Proposed Adder1:**

Unlike in ripple carry adder binary tree adder structure Here carries generated in the present stage of adder are routed to next stage full adders present in next place value of the adders except in final stage, because of which all the full adders present in the first stage are independent to each other, and they are executed simultaneously, full adders present in the second stage are independent to each other but they are dependent on first stage outputs i.e. adders present in second stage are executed only after the execution of full adders present in the first stage so, delay is reduced when compared with ripple carry adder binary tree adder, delay analysis is shown in below table

|  |  |
| --- | --- |
| **Adder** | **Delay** |
| Proposed adder1 4 bit | 3.827 ns |
| Proposed adder2 8 bit | 4.208ns |

**Table 5.1 Delay Analysis of Proposed Adders**

30

**5.2 Proposed Adder 2:**

**5.2.1Hardware required:**

For 4 bit adder

7 Full adders,

1 Half adders,

1 four bit Han Carlson adder

For 8 bit adder,

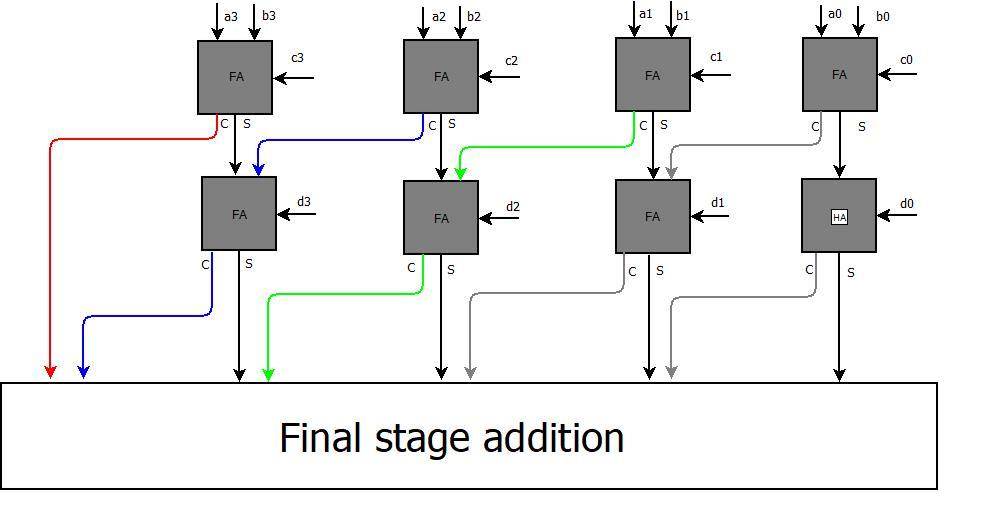
15 Full adders,

1 Half adders

1 eight bit Han Carlson adder

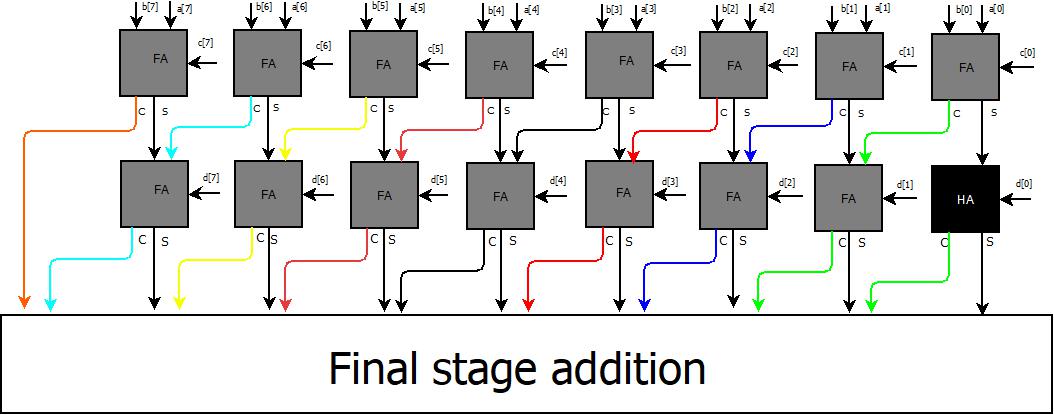
**5.2.2 Architecture of Proposed adder 2:**

Four bit and Eight bit implementations of proposed adder are being showed in the following figures



**Fig 5.3 4 bit Proposed Adder2**

31



**Fig 5.4 8 bit Proposed Adder2**

Here carries generated in the present stage of adder are routed to next stage full adders present in next place value of the adders as shown in the above figures and the final ripple carry stage present in proposed adder is replaced with Final stage addition block as shown in the above figures

Different adders like Han Carlson parallel prefix adder, Han Carlson Sum Propagate adders can be used in final stage addition block. Han Carlson parallel prefix adder is shown in the Fig4.1, Han Carlson Sum Propagate adders is shown in fig 4.2

32

**5.2.3 Delay analysis of Proposed Adder 2:**

Ripple carry stage present in final stage of proposed adder has more dependency as carry is propagated from one full adder to another full adder present in next place value as shown in below fig so, this ripple carry stage is replaced with final stage addition block to increase independency which reduces delay. Delay analysis of proposed adder 2 is shown in the below table

|  |  |
| --- | --- |
| **Adder** | **Delay** |
| 4 bit Proposed adder2 using Han Carlson | 3.937 ns |
| Sum propagate adder |  |
| 8 bit Proposed adder2 using Han Carlson | 4.646 ns |
| Sum propagate adder |  |
| 4 bit Proposed adder2 using Han Carlson | 3.139 ns |
| Prefix adder |  |
| 8 bit Proposed adder2 using Han Carlson | 3.755 ns |
| Prefix adder |  |

**Table 5.2 Delay Analysis of Proposed Adder2**

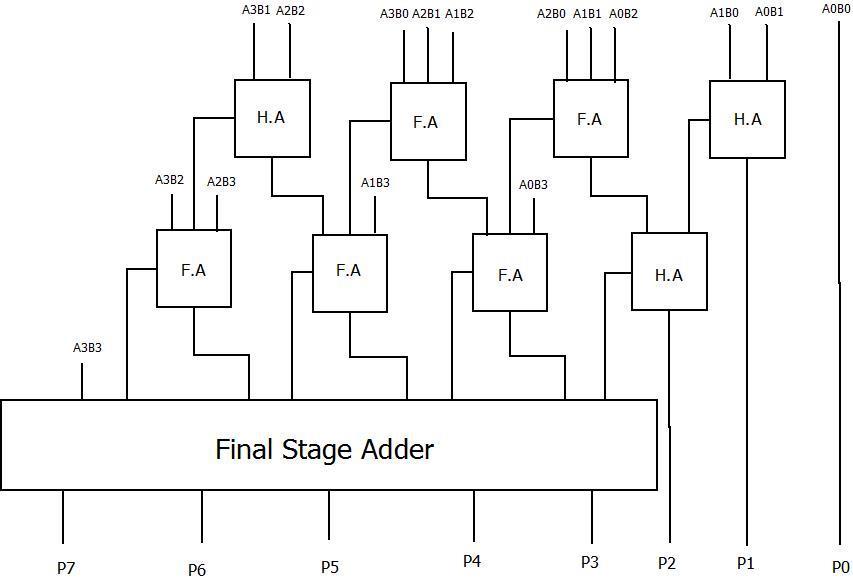
33

**CHAPTER-6**

**Proposed Multiplier**

**6.1 Wallace – Han-Carlson Multiplier:**

The Multiplier we proposed is a combination of Wallace Tree Multiplier and Han-Carlson Adder. The final stage of Wallace Tree Multiplier is addition of partial products generated from the previous stages, this operation is performed by the Han-Carlson Adder, replacing the Ripple Carry Adder or Carry Save Adder.



**Fig 6.1: 4-Bit Proposed Multiplier (Wallace – Han-Carlson Multiplier)**

34

Note: Some of the products are acquired at initial stages of the Multiplier, Those are not needed in the final stage adder, similar to the Wallace Tree Multiplier. The Final Stage Adder is Han-Carlson Adder shown In Fig.

The usage of Han-Carlson Adder Results in reduction of delay as it performs in parallel, for addition of partial products. The Final Products of the Multiplier are P0, P1, P2, P3, P4, P5, P6, and P7.

**Comparison between Wallace Tree Multiplier and Wallace-Han-Carlson Multiplier:**

1. Both the Multipliers perform the multiplications in Three Stages.
2. In Wallace Tree Multiplier, the Partial products are reduced as soon as possible.
3. In Wallace-Han-Carlson Multiplier the generation of partial products is similar to the Wallace Tree Multiplier. But when comes to the addition of the partial products the multiplier with Han-Carlson Adder performs faster.
4. Generally for a given size, the Wallace Tree Multiplier and Multiplier with Han-Carlson Adder, there is difference in the delay.
5. Due to the presence of the Han-Carlson Adder in the Proposed Multiplier yields the Multiplication result faster.
6. Although the Wallace multiplier uses a slightly smaller carry propagating adder, usually this provides no significant speed advantage.

|  |  |  |
| --- | --- | --- |
| **Circuit Type (Length)** | **Final Stage Adder** | **Delay** |
| Wallace Multiplier 4-Bit | Ripple Carry Adder | 5.897.ns |
|  |  |  |
| Wallace Multiplier 4-Bit | Sum Adder(Han-Carlson) | 4.012ns |
|  |  |  |
| Wallace Multiplier 4-Bit | Prefix Adder(Han-Carlson) | 3.125ns |
|  |  |  |
| Wallace Multiplier 8-Bit | Ripple Carry Adder | 12.894ns |
|  |  |  |
| Wallace Multiplier 8-Bit | Sum Adder(Han-Carlson) | 8.217ns |
|  |  |  |
| Wallace Multiplier 8-Bit | Prefix Adder(Han-Carlson) | 7.125ns |
|  |  |  |

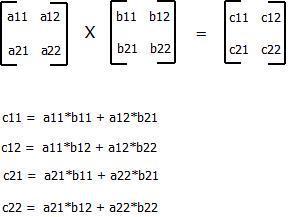
**Table 6.1: Delay Analysis of Wallace Multiplier and Wallace - Han-Carlson Multiplier**

35

**CHAPTER-7**

**Matrix Multiplication**

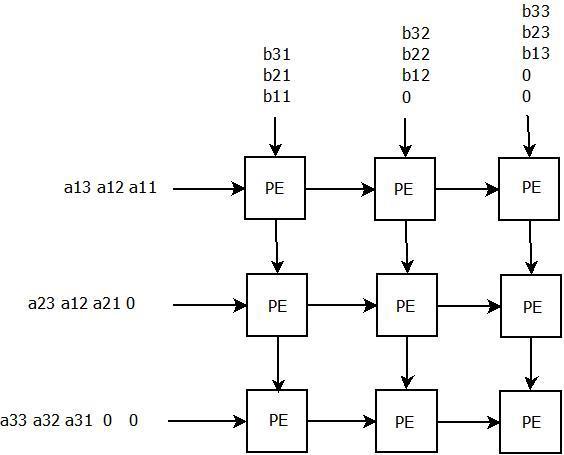
Matrix Multiplication is a binary operation. Matrix Multiplication is very common operation in mathematical analysis and scientific operations. It plays a key role in applications such as Digital Image Processing (DIP), Digital Signal Processing (DSP), Radars, Sonar and Robotics. The Matrix Multiplication C=A\*B of two Matrices A and B is conformable, if the number of columns of A is equal to number of rows of B.



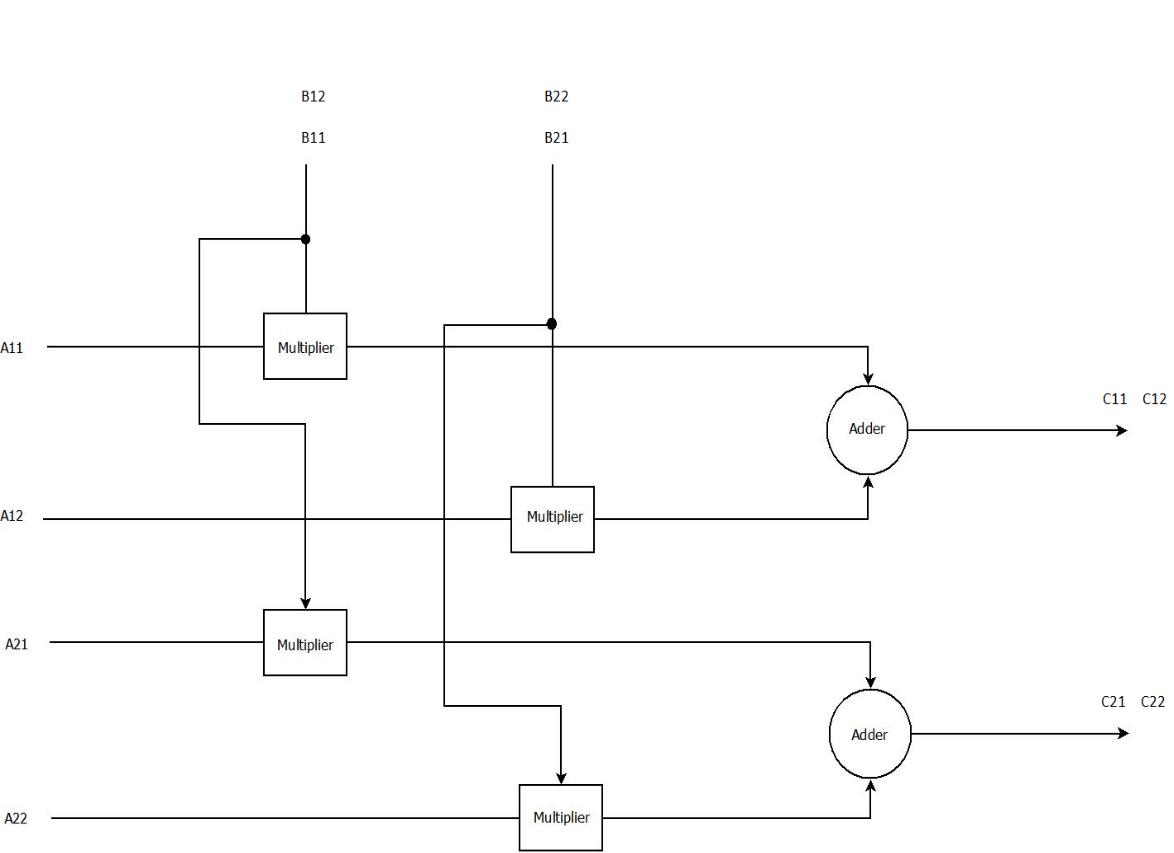
**2\*2 Matrix Multiplication**

The Matrix Multiplication is a time consuming process which results in computational complexity time O (n3) on a sequential processor and O (n3/P) on parallel system with P processors. To enhance the performance of matrix multiplier various design and architectures have been developed to solve the problem more efficiently. The Design we proposed is parallel array design, which maps a nested loop algorithm onto the parallel architecture as shown in Fig ().The Parallel Array design capable of exploiting the inherent parallelism of Matrix Multiplication.

36



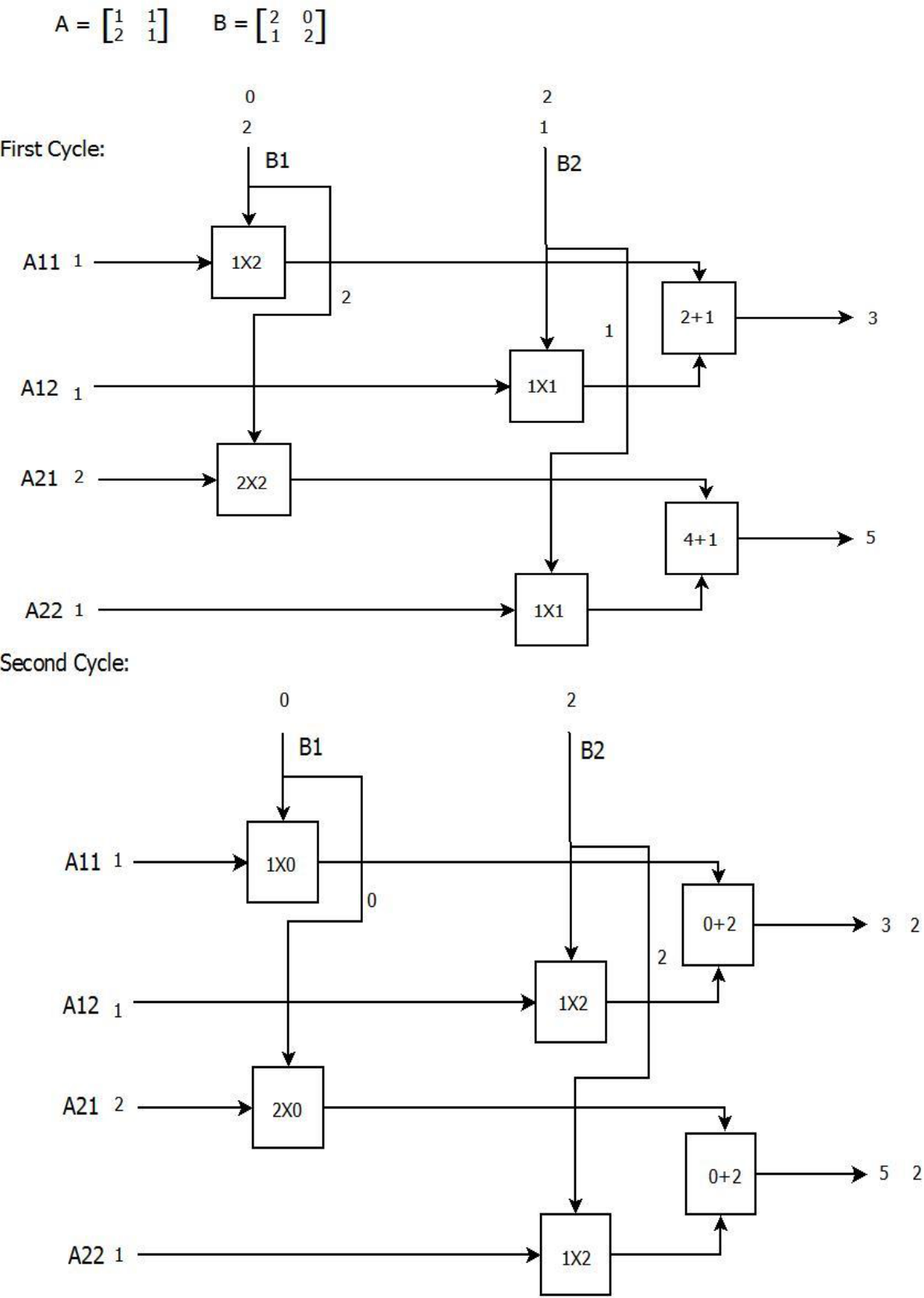
**3X3 Systolic Array**



**Fig 7.1: 2X2 Proposed Multiplier**

37

**Example:**



38

* In First Cycle the first row of matrix A is multiplied with the first column of matrix B. Similarly the second row of matrix A is multiplied with the first column of matrix B.
* In Second Cycle the first row of matrix A is multiplied with the second column of matrix B. Similarly the second row of matrix A is multiplied with the second column of matrix B.
* The output C matrix is obtained in two cycles for 2X2 Matrix.

The Multiplier we used in the matrix multiplication is the proposed Wallace Multiplier using Han-Carlson Adder, Which is also the adder used to add the products of the multipliers.

The Proposed Matrix Multiplication is very much similar to traditional systolic array except there are slight modifications. The Proposed Matrix Multiplication consists of identical Processing Elements (PEs) and the number of PE depends used in the matrix multiplication depends on the size of matrices. Each Processing Element performs the multiply and Accumulate (MAC) operation. The PE operates independently by the connections at the input and output ports.

The Pipelining and Parallel processing both the techniques are used to improve the performance of the Matrix Multiplier. The Pipelining process is used, where there is requirement like high speed applications, massive throughput. The Parallel process is used, where the matrix multiplication operation is divided into several smaller operations which are executed in parallel and the output is obtained by addition of all the products generated from the parallel processing.

|  |  |
| --- | --- |
| **Circuit Type** | **Delay** |
|  |  |
| Matrix Multiplier with RCA | 7.719ns |
|  |  |
| Matrix Multiplier with Prefix Han-Carlson | 4.819ns |
| Prefix Adder |  |
|  |  |

**Table 7.1: Delay Analysis of Matrix Multiplier and Matrix Multiplier using Han-Carlson Multiplier**

39

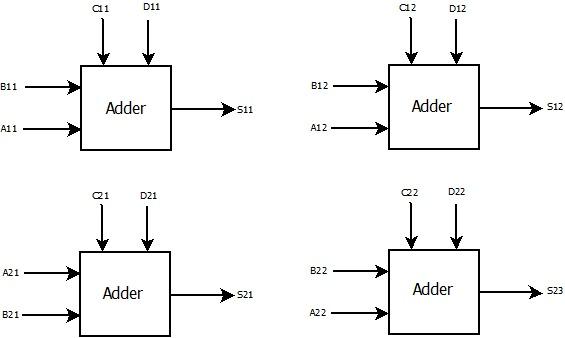
**CHAPTER-8**

**Matrix Addition**

* Matrix addition block diagrams are shown in fig 8.1 and fig 8.2
* Fig 8.1 represents Implementation of 2X2 matrix addition
* Fig 8.2 represents Implementation of 3X3 matrix addition
* The adder we used in the Matrix addition is Proposed adder 2, as it has less propagation delay when compared to all other adders
* As Proposed adder 2 supports four operands we can simultaneously add four matrices
* For N X N matrix addition we require N2 number of adders, for example, if we are performing 2X2 Matrix addition we require 2\*2=4 number of adders, if we

are performing 3X3 matrix addition we require 3\*3 = 9 number of adders

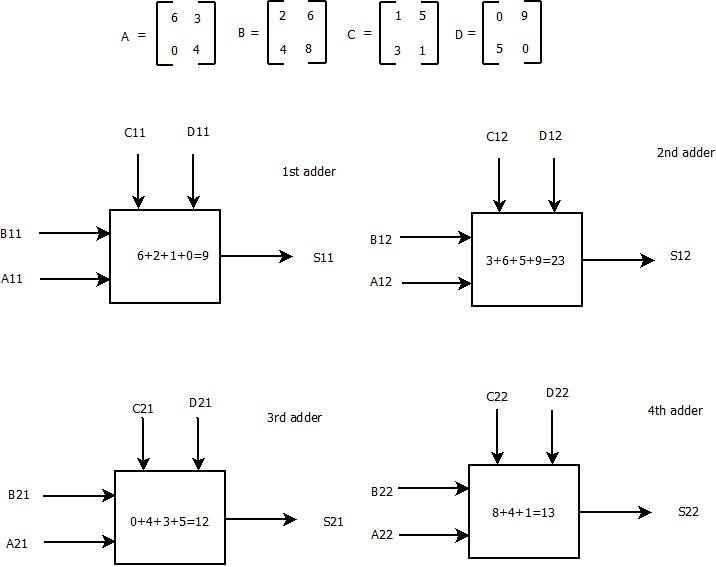
* All the inputs(A,B,C,D) to the adders are given simultaneously
* As all the adders are executed simultaneously output(S) is obtained after the first clock cycle itself



**Fig 8.1 2 X 2 Matrix addition**

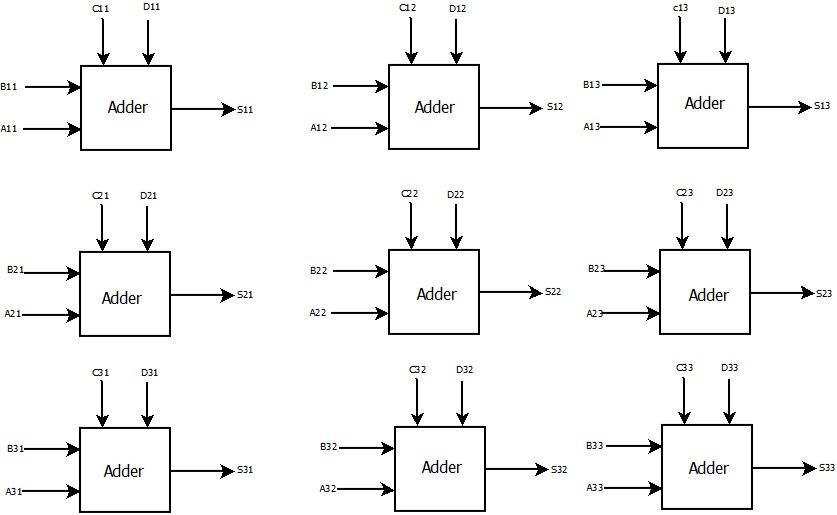
40

**Example:**



* Matrix addition takes place as shown in the above example
* As shown in the above figure
* Inputs (A11, B11, C11, and D11) are given to 1st proposed adder2 to produce sum (S11)
* Inputs (A12, B12, C12, and D12) are given to 2nd proposed adder2 to produce sum (S12)
* Inputs (A21, B21, C21, and D21) are given to 3rd proposed adder2 to produce sum (S21)
* Inputs (A22, B22, C22, and D22) are given to 4th proposed adder2 to produce sum (S22)

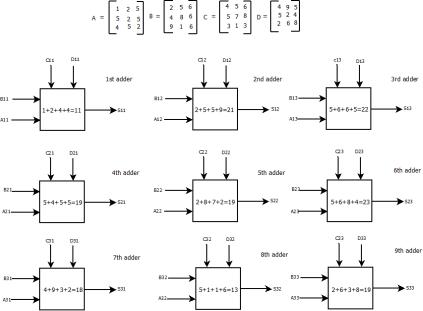
41



**Fig 8.2 3 X 3 Matrix addition**

42

**Example:**



* As shown in the above figure
* Inputs (A11, B11, C11, and D11) are given to 1st proposed adder2 to produce sum (S11)
* Inputs (A12, B12, C12, and D12) are given to 2nd proposed adder2 to produce sum (S12)
* Inputs (A13, B13, C13, and D13) are given to 3rd proposed adder2 to produce sum (S13)
* Inputs (A21, B21, C21, and D21) are given to 4th proposed adder to produce sum (S21)
* Inputs (A22, B22, C22, and D22) are given to 5th proposed adder2 to produce sum (S22)
* Inputs (A23, B23, C23, and D33) are given to 6th proposed adder2 to produce sum (S23)
* Inputs (A31, B31, C31, and D31) are given to 7th proposed adder2 to produce sum (S31)

43

* Inputs (A32, B32, C32, and D32) are given to 8th proposed adder2 to produce sum (S32)
* Inputs (A33, B33, C33, and D33) are given to 9th proposed adde2r to produce sum (S33)

Delay Analysis for the Matrix adder circuit with RCA BTA and Matrix adder circuit with proposed adder 2 is shown in the below table from the table we can observe that Matrix addition circuit constructed using proposed 2 has less delay when compared with the circuit constructed using RCA BTA.

|  |  |
| --- | --- |
| **Circuit Type** | **Delay** |
|  |  |
| Matrix Addition with RCA BTA | 6.799 ns |

Matrix addition with Proposed adder 2

3.755ns

**Table 7.2: Delay Analysis of Matrix Addition with RCA BTA and Proposed adder 2**

44

**CHAPTER-9**

**SOFTWARE DESCRIPTION**

**9.1 Xilinx Tool:**

**Xilinx ISE** (**I**ntegrated **S**ynthesis **E**nvironment) is a software tool produced by [Xilinx](https://en.wikipedia.org/wiki/Xilinx) for synthesis and analysis of [HDL](https://en.wikipedia.org/wiki/Hardware_description_language) designs, enabling the developer to [synthesize](https://en.wikipedia.org/wiki/Logic_synthesis) ("compile") their designs, perform [timing analysis,](https://en.wikipedia.org/wiki/Static_timing_analysis)

examine [RTL](https://en.wikipedia.org/wiki/Register_transfer_level) diagrams, simulate a designed circuit’s reaction to different stimuli, and configure the target device with the [programmer.](https://en.wikipedia.org/wiki/Programmer_(hardware))

Xilinx ISE is a design environment for Field Programmable Gate Array (FPGA) products from Xilinx, and is tightly-coupled to the architecture of such chips, and cannot be used with Field Programmable Gate Array products from other vendors. The Xilinx ISE is primarily used for design and circuit synthesis, while ISIM or the [Model Sims](https://en.wikipedia.org/wiki/ModelSim) logic simulator is used for system-level testing. Other components shipped with the Xilinx ISE include the Embedded Development Kit (EDK), a Software Development Kit (SDK) and Chip Scope Pro.

Since 2012, Xilinx ISE has been discontinued in favour of [Vivado Design Suite,](https://en.wikipedia.org/wiki/Xilinx_Vivado) that serves the same roles as ISE with additional features for [system on a](https://en.wikipedia.org/wiki/System_on_a_chip) [chip](https://en.wikipedia.org/wiki/System_on_a_chip) development. Xilinx released the last version of ISE in October 2013 (version 14.7), and states that "ISE has moved into the sustaining phase of its product life cycle, and there are no more planned ISE releases."

**9.2 Xilinx User Interface**

The primary user interface of the ISE is the Project Navigator, which includes the design hierarchy (Sources), a [source code](https://en.wikipedia.org/wiki/Source_code) editor (Workplace), an output console window (Transcript), and a processes tree (Processes).

The Design hierarchy consists of design files (modules), whose dependencies are interpreted by the ISE and displayed as a [tree structure.](https://en.wikipedia.org/wiki/Tree_structure) For single-chip designs there may be one main module, with other modules included by the main module, similar to

45

the main () subroutine in programming languages. Design constraints are specified in modules, which include pin configuration and mapping.

The Processes hierarchy describes the operations that the ISE will perform on the currently active module. The hierarchy includes compilation functions, their dependency functions, and other utilities. The window also denotes issues or errors that arise with each function.

The Transcript window provides status of currently running operations, and informs engineers about the design issues. Such issues may be filtered to show Warnings, Errors, or both.

**9.3 Simulation:**

System-level testing can be performed with ISIM or the [Model Sim](https://en.wikipedia.org/wiki/ModelSim) logic simulator, and such test programs must be written in HDL languages. Test bench programs may comprise simulated input signal waveforms, or we can force the input to program, or monitors which observe and verify the outputs of the [device under test.](https://en.wikipedia.org/wiki/Device_under_test)

46

**CHAPTER-10**

**Conclusion**

The Multiplier we have proposed can be implemented in various multiplication processes to achieve high performance and efficient operation speed. The Matrix Multiplication we have performed using the Multiplier and Han-Carlson adder have been yielded great results in Xilinx Software.

As we have gone across the designs of the adders, it’s time to choose the high performance adder in terms of delay, from the beginning of the project, our aim is to choose a adder with less delay, by executing all the above designs in Xilinx tool taking down the delay readings from all the designs .

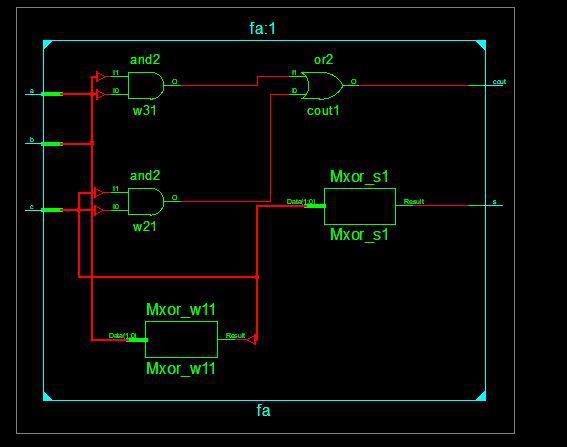
**Below are the readings for the designs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Adder 4 bit** | **Delay** | **Adder 8 bit** | **Delay** |
| RCA BTA | 3.815 ns | RCA BTA | 6.799 ns |
|  |  |  |  |
| Proposed adder1 | 3.827 ns | Proposed adder1 | 4.208 ns |
|  |  |  |  |
| Proposed adder2 | 3.937 ns | Proposed adder2 | 4.646 ns |
| using Han Carlson |  | using Han Carlson |  |
| Sum adder |  | Sum adder |  |
| Proposed adder2 | 3.139 ns | Proposed adder2 | 3.755 ns |
| using Han Carlson |  | using Han Carlson |  |
| Prefix adder |  | Prefix adder |  |

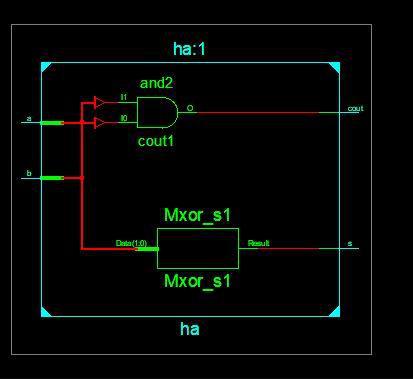
**Table 8.1 Delay Comparison table of different adders**

47

**Results:**

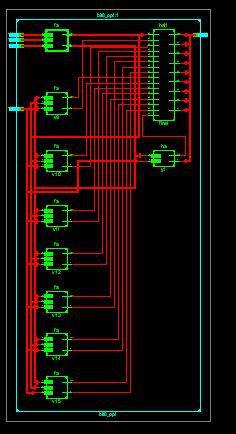


**Schematic of Full adder**

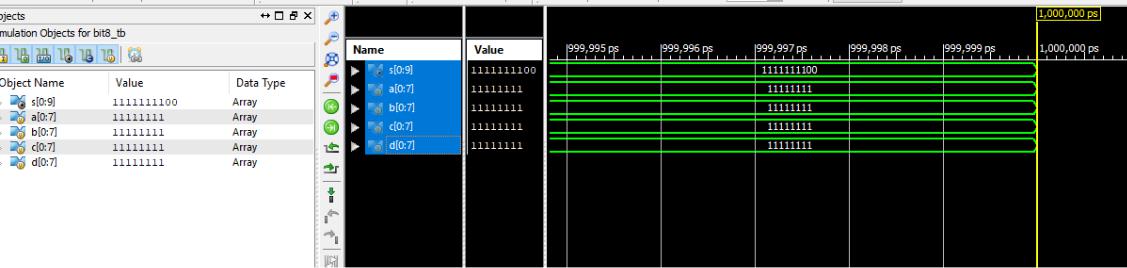


**Schematic of Half adder**

48

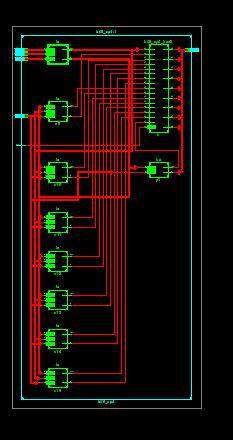


**Schematic of Proposed adder2 using Han Carlson Prefix adder**



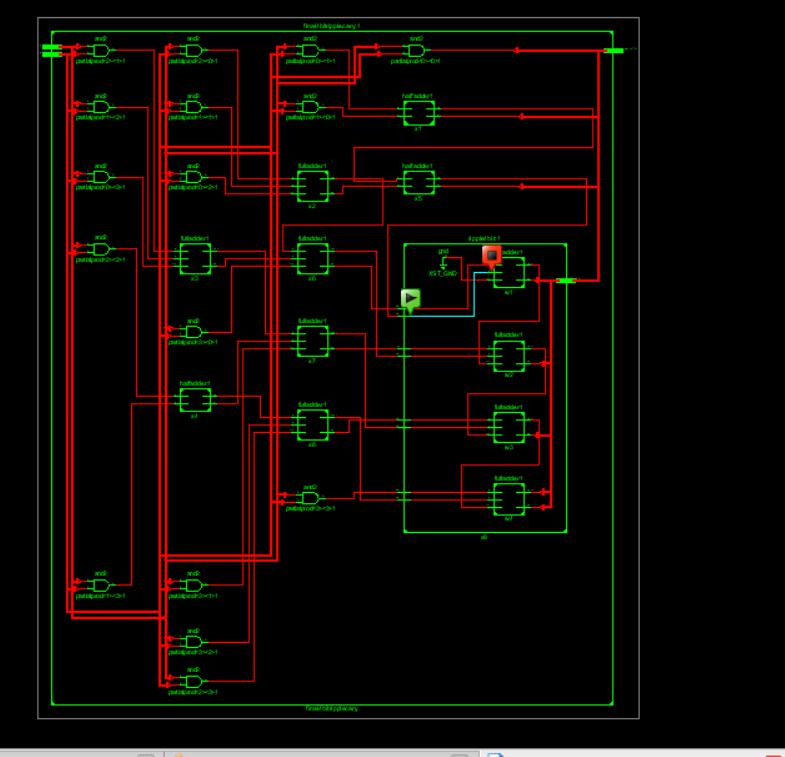
**Result of proposed adder2**

49

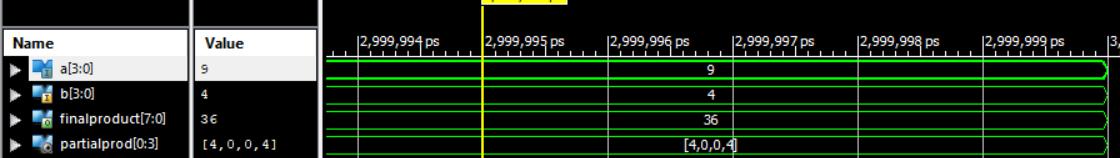


**Schematic of Proposed adder2 using Han Carlson Sum adder**

50

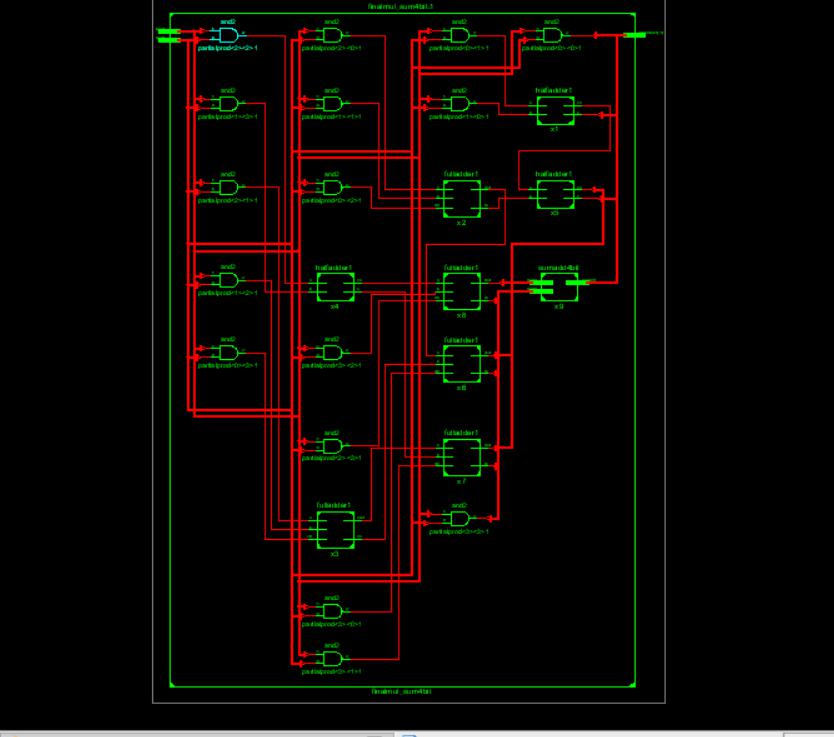


**Schematic of 4-Bit Wallace Tree Multiplier**



**Result of 4-Bit Wallace Multiplier**

51

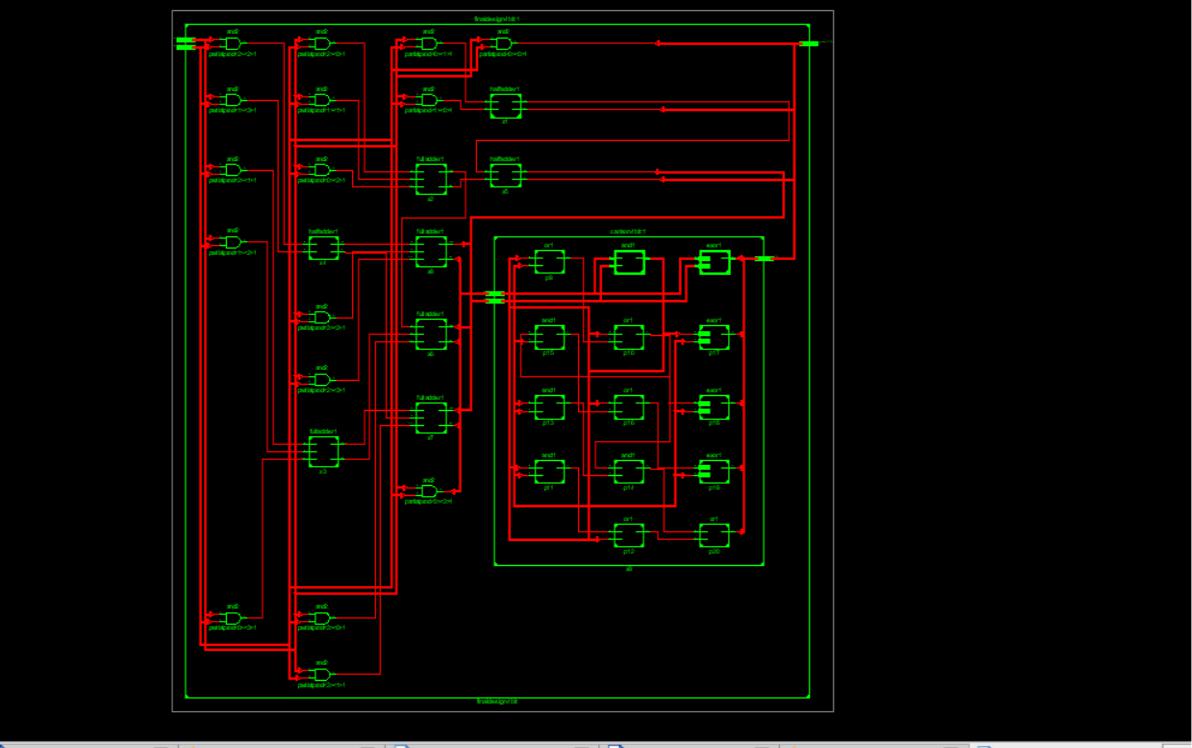


**Schematic of 4-Bit Wallace-Han-Carlson (Sum Adder) Multiplier**

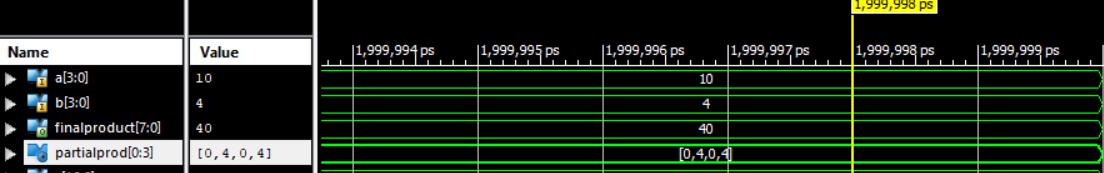


**Result of 4-Bit Wallace-Han-Carlson (Sum Adder) Multiplier**

52

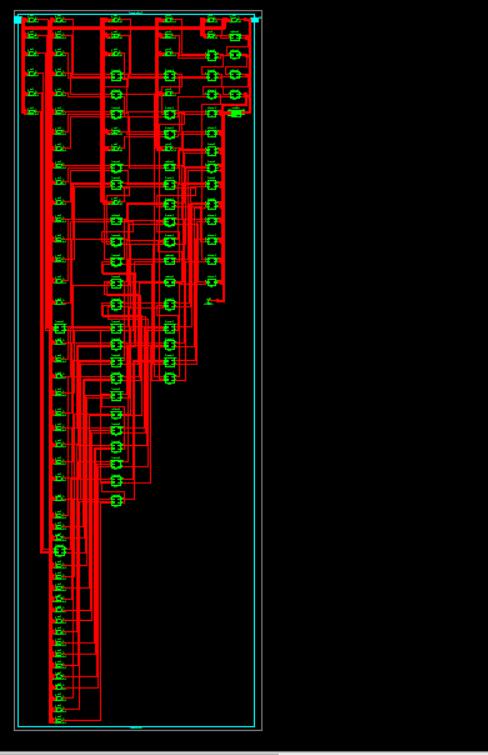


**Schematic of 4-Bit Wallace-Han-Carlson (Prefix Adder) Multiplier**

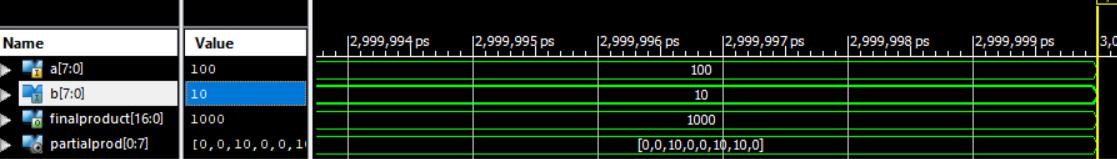


**Result of 4-Bit Wallace-Han-Carlson (Prefix Adder) Multiplier**

53



**Schematic of 8-Bit Wallace Multiplier**

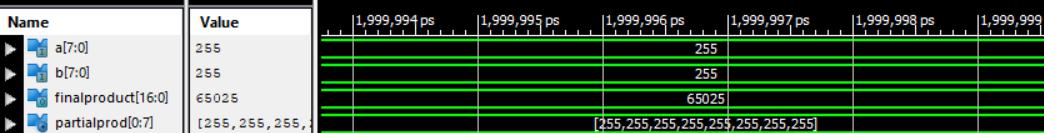


**Result of 8-Bit Wallace Multiplier**

54



**Schematic of 8-Bit Wallace-Han-Carlson (Sum Adder) Multiplier**

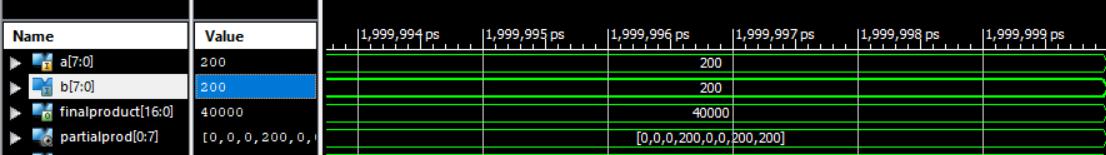


**Result of 8-Bit Wallace-Han-Carlson (Sum Adder) Multiplier**

55

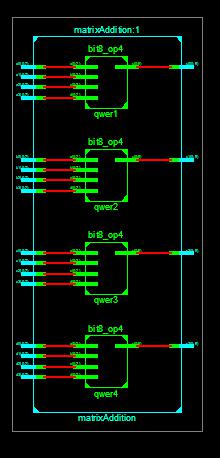


**Schematic of 4-Bit Wallace-Han-Carlson (Prefix Adder) Multiplier**



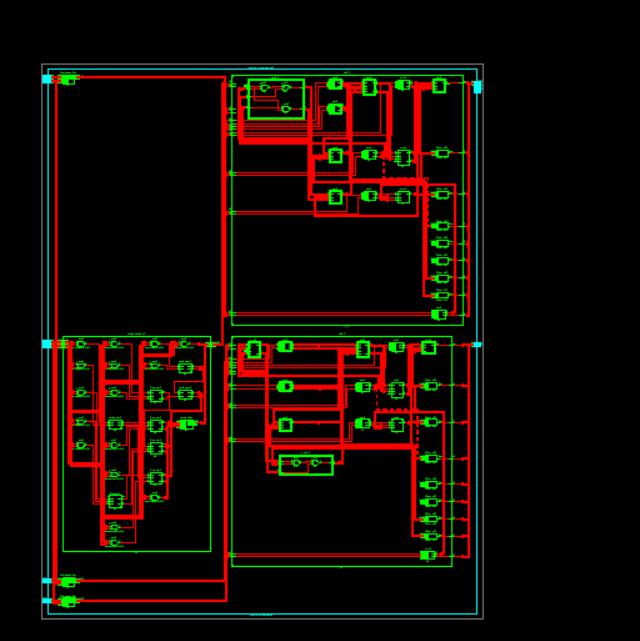
**Result of 8-Bit Wallace-Han-Carlson (Prefix Adder) Multiplier**

56

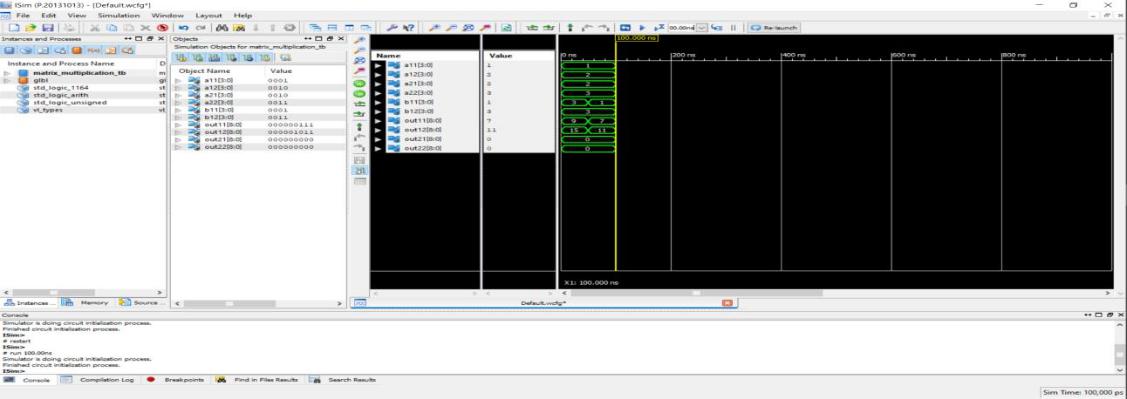


**Schematic of Matrix 2x2 addition**

57

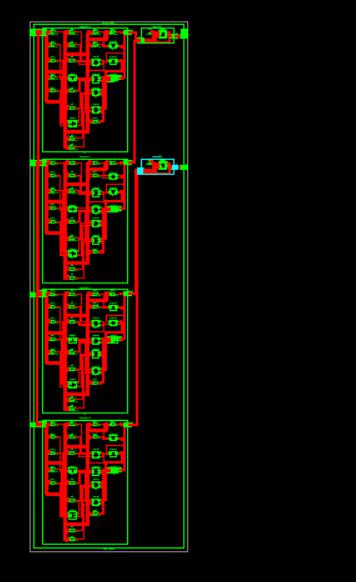


**Schematic of Matrix 2x2 Multiplication**

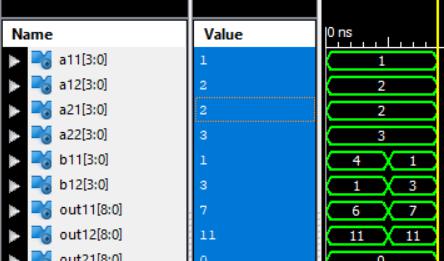


**Result of Matrix 2x2 Multiplication**

58



**Schematic of Proposed Matrix 2x2 Multiplication**



**Result of Proposed Matrix 2x2 Multiplication**

59

**REFERENCES**

1. Sujit Kumar Patel1, Subodh Kumar Singhal2, “ *Area–delay and energy efficient multi-operand binary tree adder*”.
2. Amit Kumar Panda , Rakesh Palisetty ,and Kailash Chandra Ray

“*High-Speed Area-Efficient VLSI Architecture of Three-Operand Binary Adder*”

1. Giorgos Dimitrakopoulos, Kleanthis Papachatzopoulos, and Vassilis

Paliouras, “ *Sum Propagate Adders*”

1. Shanmugakumar M, Vegesna S. M. Srinivasavarma, Noor Mahammad

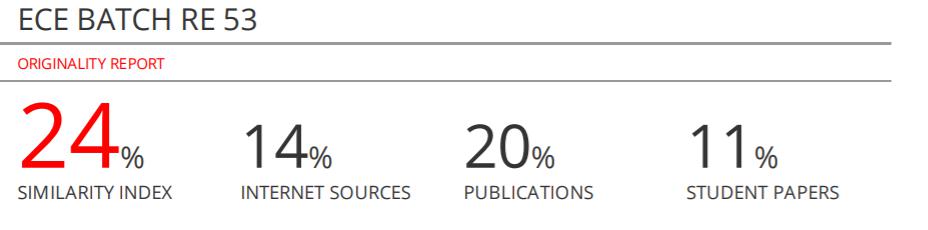
Sk. “*Energy Efficient Hardware Architecture for Matrix*

*Multiplication”*

1. Sakshi Sharma1, Pallavi Thakur 2 “*Design of High Speed Power Efficient Wallace Tree Adders*”
2. Vikramkumar Pudi, K. Sridharan, and Fabrizio Lombardi, “ *Majority Logic Formulations for Parallel Adder Designs at Reduced Delay and Circuit*

*Complexity”*

60



61