IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 13, Issue 1, Ver. II (Jan.- Feb. 2018), PP 29-38
www.iosrjournals.org

Timing Analysis of Generic Multipliers

Anju Rajput
[1]*(ECE Department, Arya Institute of Engineering & Technology, India)*
Corresponding Author: Anju Rajput

Abstract: *Multiplications are very expensive process and slow the overall operation. The performance of many computational problems is often dominated by the speed at which a multiplication operation can be executed. A multiplier is one of the key hardware blocks in most digital signal processing (DSP) systems. In DSP applications, multiplier plays a vital role include digital filtering, digital communication and spectral analysis.(Ayman.A et al (2001)).The aim of this paper is to present analysis of generic multipliers on the basis of timing performance. The generic architecture of all the four multipliers i.e. array, column bypass, wallace tree and booth multiplier has been analysed. Analysis of these multipliers tells that Wallace tree occupy more area but also faster than the other. The simulation and synthesis have been carried out on ISE design suite-14.9. By using this analysis different multipliers can be used for different applications in which timing performance is constraint.*

Keywords: *Array Multiplier, Booth Multiplier, Column Multiplier, Wallace Tree Multiplier*

Date of Submission: 15-02-2018 Date of acceptance: 01-03-2018

I. Introduction

For the fabrication of DSP system and high performance systems, low power consumption and small area are most important design criteria. For any multiplier optimizing the speed and area of multiplier is an important design issue. Multipliers are the basic element in the Microprocessors and DSPs. Multipliers are the major source of power dissipation. Power consumption of multipliers can be reduce at various levels of design hierarchy.Using different algorithm, power consumption can be reduced. Multiplication is a process of adding an integer to itself a specified number of times. Multiplicand is added to itself a number of times as specified by Multiplier to form the result that we called product. . Area and speed are important design criteria. By analysing these constraints multiplier which suits best can be used.[13]The purpose of this paper is to present analysis of four different multipliers i.e. array, column bypass, wallace tree and booth multipliers, on the basis of area and timing performance. Analysis of these multipliers tells that Wallace tree occupy more area but also faster than the other. The growing market for fast floating-point co-processors, digital signal processing chips, and graphics processors has created a demand for high-speed, area-efficient multipliers. Computational performance of a DSP system is limited by its multiplication performance[10]. Multipliers are the main power eating elements of DSP and communication systems. Therefore high speed & low power multiplier is much desired[7]. The three important considerations for VLSI design are Power, area and delay. There are many proposed logics of low power dissipation and high speed, each logic style has its own advantages in terms of speed and power.

II. Multipliers

1). Array Multiplier: It is based on shift and add algorithm. It is used for small circuits. It requires less hardware but takes more time. Each digit of multiplier is multiplied with the multiplicand. We write the result which is called the partial products. Then we take the second digit of the multiplier and multiply with the same multiplicand, write it down from the first partial product, but we do the shifting. We will continue to do that until we exhaust all the digit of the multiplier. Finally we add the whole thing, to get the fellow product. This algorithm is known as shift /add algorithm, because we have to add it after shifting.

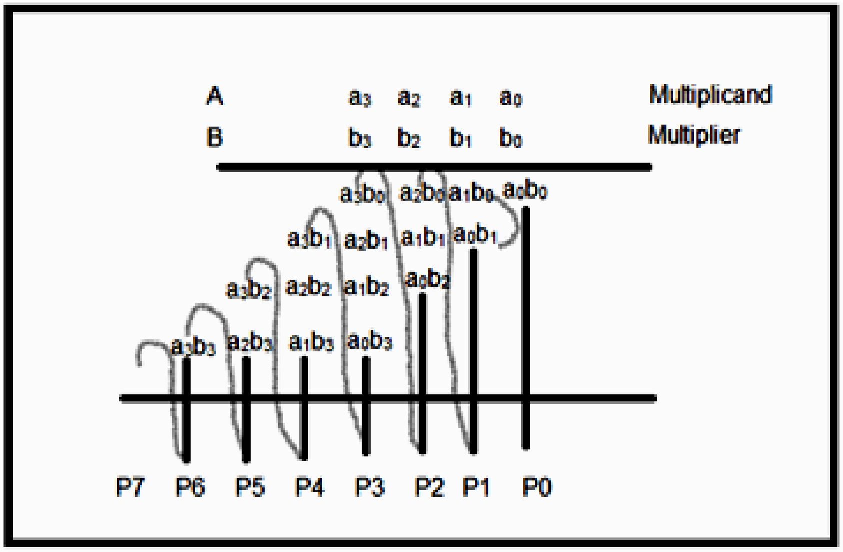

A
a_3 a_2 a_1 a_0
Multiplicand
B
b_3 b_2 b_1 b_0
Multiplier
a_3b_0 a_2b_0 a_1b_0 a_0b_0
a_3b_1 a_2b_1 a_1b_1 a_0b_1
a_3b_2 a_2b_2 a_1b_2 a_0b_2
a_3b_3 a_2b_3 a_1b_3 a_0b_3
P7 P6 P5 P4 P3 P2 P1 P0

Fig.1 *Basic idea of Array Multiplier [1]*

All the partial products are added to get the fellow product, but before adding we have to do shifting of the partial product to one bit position in left as shown in **fig.1** [1]. The idea of basic array multiplier is shown below which will give the partial products on each multiplication and results will be shown as P_0 P_1 P_2 P_3 P_4 P_5 P_6& P_7.

2). Column Bypass Multiplier: Column Bypassing in case of multiplier means turning off some columns in the multiplier array in case when certain multiplicand bits are zero. In this technique, during working, the operations in a column can be disabled if the corresponding bit in the multiplicand is 0, to save the power. The modified cell of adder circuit is shown in fig.2 below.

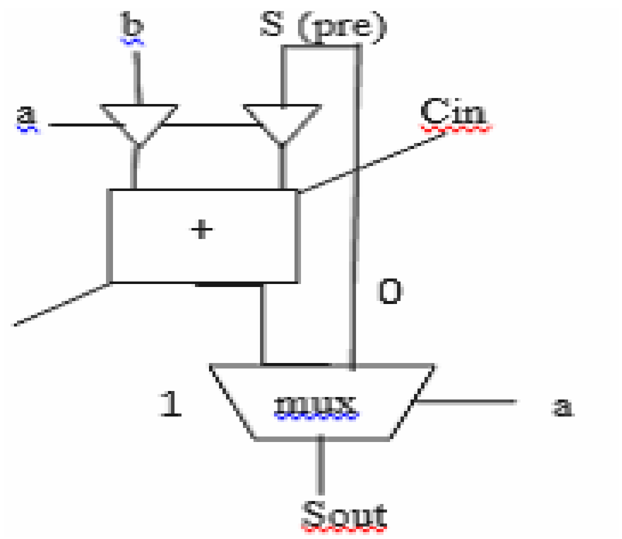
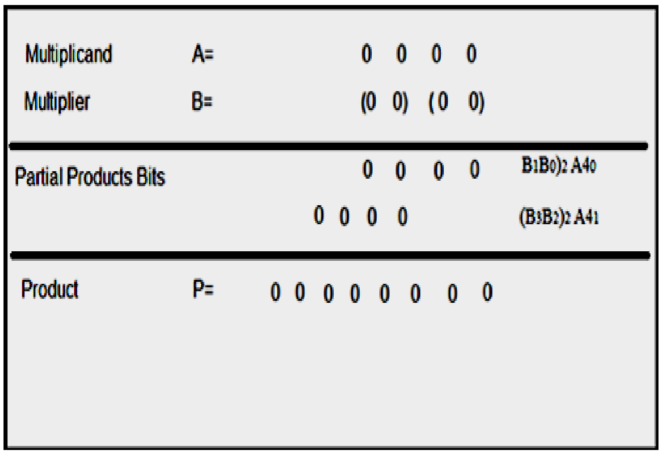


Fig.2 *Modified cell of Adder circuit*

In **fig.2**, there are two tristate buffers, a and b are the inputs, S(pre) is the previous sum and Cin is the previous carry. When a=0, multiplexers' select line is 0, then b will not get propagated and S (pre) will be the output. When a=1, select line is 1 then a, b and S(pre) will get added by the adder and Sout is the output[2].

3). Booth Multiplier: To enhance the addition among the partial products, requirement of fast adder architectures are required. The Modified- Booth algorithm is mostly used for high speed multiplier circuits. By decreasing the number of generated partial products, we can improve the multiplier performance. For the signednumbers multiplication booth is a powerful algorithm.[8].



```
Multiplicand          A=              0  0  0  0
Multiplier            B=             (0  0)  (0  0)

Partial Products Bits                 0  0  0  0     B1B0)2 A40
                                  0  0  0  0         (B3B2)2 A41

Product               P=       0 0 0 0 0 0  0  0
```

Fig.3 Booth Multiplication

It treats both signed and unsigned numbers. Booth algorithm is a technique which will reduce number of multiplicand multiples. The booth multiplication is shown in **fig 3** [2] in which A_x is multiplicand and B_x is multiplier, P_x is the product of booth multiplication.

4). Wallace Tree Multiplier: For fast multiplication of two numbers, Wallace tree multiplier is used. It is faster than simple array multiplier [11]. It has logarithmic height. But Wallace tree has irregular wiring. Due to this reason, it is often avoided by designer .Wallace tree use log-depth tree network for reduction [6].

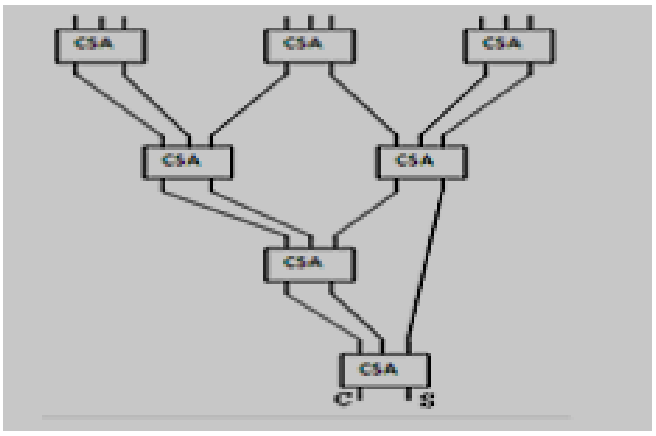


Fig.4 Wallace Tree

This multiplier is not used for low power applications, because excess wiring consumes extra power. But it is faster as it uses carry save adder. It is high speed multiplier. Basic idea of Wallace tree is shown below in **fig.4** [7] in which adders are shown which will help in implementation of Wallace tree.

III. Timing Report of Different Multipliers

1.Timing Report of 4-bit Array Multiplier:

Timing Summary:
Speed Grade: -4
Maximum combinational path delay: 17.681ns
Timing Detail:
All values displayed in nanoseconds (ns)
Timing constraint: Default path analysis
Total number of paths / destination ports: 330 / 8

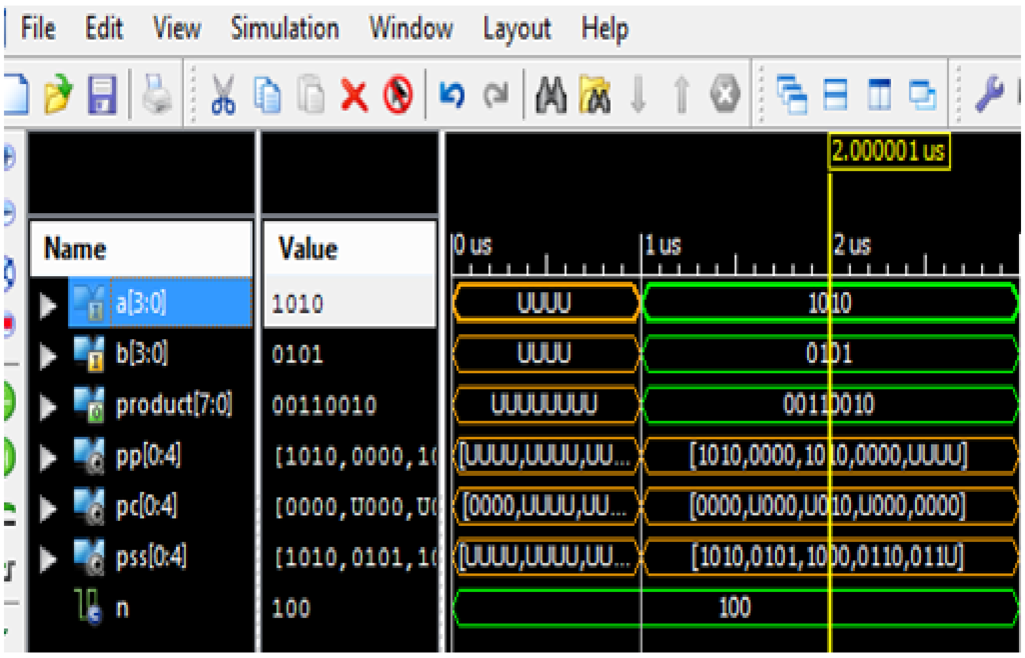


Fig.5 Simulation Result of 4-bit Array Multiplier

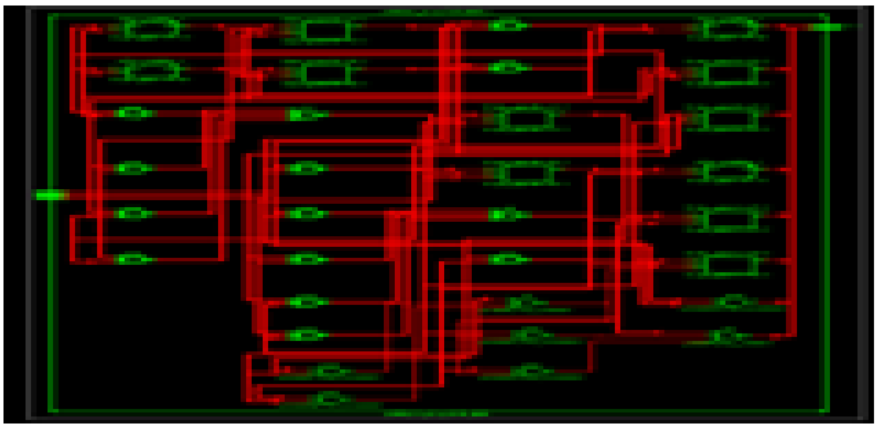


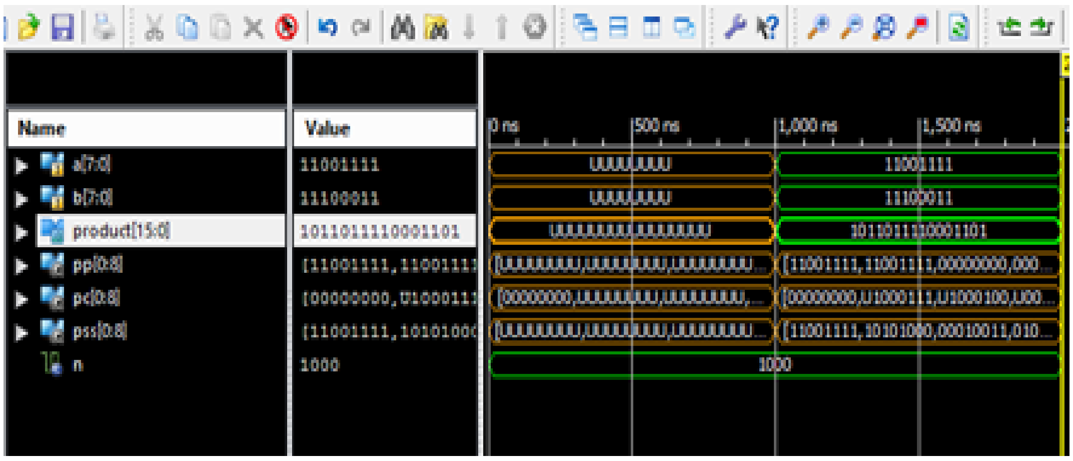
Fig.6 RTL of 4-bit Array Multiplier

```
        Total                    17.681ns (9.771ns logic, 7.910ns route)
                                          (55.3% logic, 44.7% route)
```

2).Timing Report of 8-bit Array Multiplier:

Timing Summary:Speed Grade: -4

```
   Maximum combinational path delay: 32.001ns
Timing Detail:
  Total number of paths / destination ports: 20382 / 16
-------------------------------------------------------------------------
Delay:              32.001ns (Levels of Logic = 16)
  Source:             a<0> (PAD)
  Destination:          PRODUCT<15> (PAD)
                              Gate      Net
    Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
    ----------------------------------------  ------------
 Total    32.001ns (14.179ns logic, 17.822ns route)
```



```
Name            Value
a[7:0]          11001111
b[7:0]          11100011
product[15:0]   1011011110001101
pp[0:8]         [11001111,1100111
pc[0:8]         [00000000,U100011
pss[0:8]        [11001111,1010100
n               1000
```

Fig.7 Simulation of 8-bit Array Multiplier

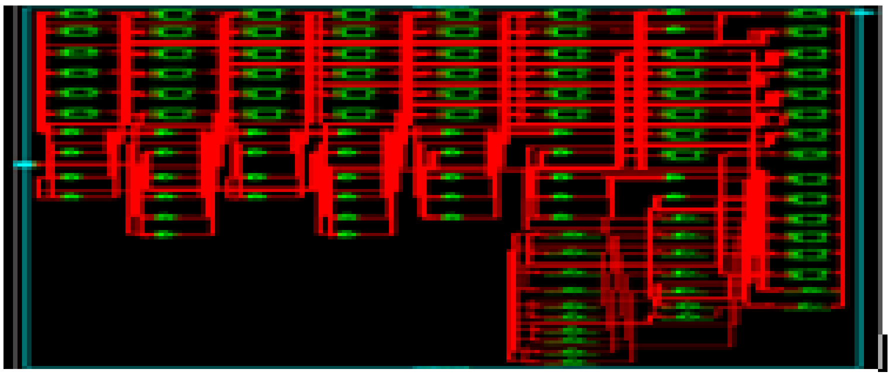


Fig.8 RTL of 8-bit Array Multiplier

3).Timing Report of 4-bit Booth Multiplier:

Timing Summary:

Speed Grade: -4

Minimum input arrival time before clock: 6.270ns

Maximum output required time after clock: 10.493ns

Maximum combinational path delay: No path found

```
------------------------------------------------------------------------
    Total                 10.493ns (8.552ns logic, 1.941ns route)
                               (81.5% logic, 18.5% route)
```

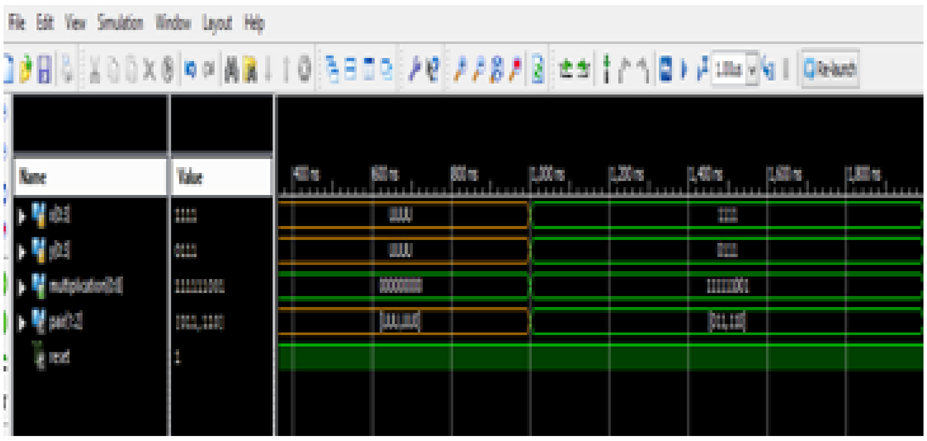


Fig.9 Simulation Result of 4-bit Booth Multiplier

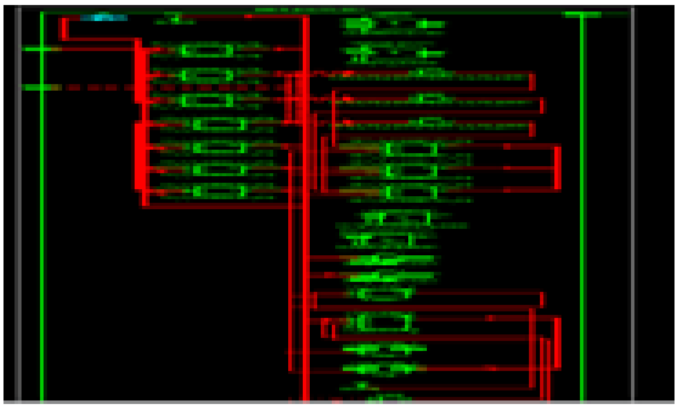
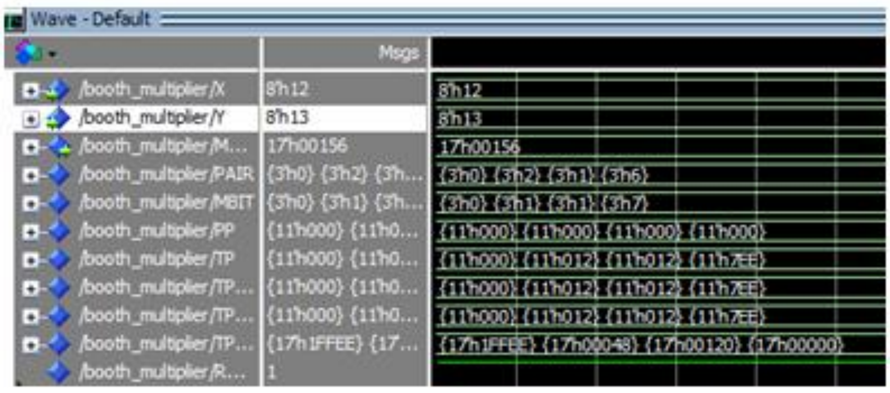


Fig.10 RTL of 4-bit Booth Multiplier

4).Timing Report of 8-bit Booth Multiplier:
Timing Summary:Timing Summary:
Speed Grade: -4
Minimum input arrival time before clock: 8.409ns
Maximum output required time after clock: 15.611ns
Timing Detail:

```
 Total              15.611ns (11.314ns logic, 4.297ns route)
                         (72.5% logic, 27.5% route)
```



```
Wave - Default
                                  Msgs
/booth_multiplier/X      8'h12                    8'h12
/booth_multiplier/Y      8'h13                    8'h13
/booth_multiplier/M...   17'h00156                17'h00156
/booth_multiplier/PAIR   {3'h0} {3'h2} {3'h...    {3'h0} {3'h2} {3'h1} {3'h6}
/booth_multiplier/MBIT   {3'h0} {3'h1} {3'h...    {3'h0} {3'h1} {3'h1} {3'h7}
/booth_multiplier/PP     {11'h000} {11'h0...      {11'h000} {11'h000} {11'h000} {11'h000}
/booth_multiplier/TP     {11'h000} {11'h0...      {11'h000} {11'h012} {11'h012} {11'h7EE}
/booth_multiplier/TP...  {11'h000} {11'h0...      {11'h000} {11'h012} {11'h012} {11'h7EE}
/booth_multiplier/TP...  {11'h000} {11'h0...      {11'h000} {11'h012} {11'h012} {11'h7EE}
/booth_multiplier/TP...  {17'h1FFEE} {17...       {17'h1FFEE} {17'h00048} {17'h00120} {17'h00000}
/booth_multiplier/R...   1
```

Fig.11 Simulation Result of 8-bit Booth Multiplier

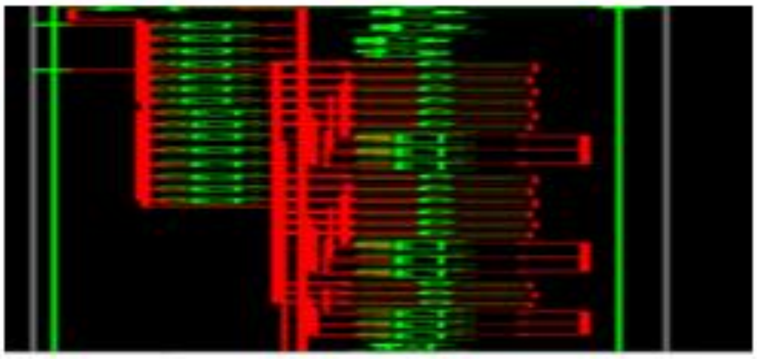
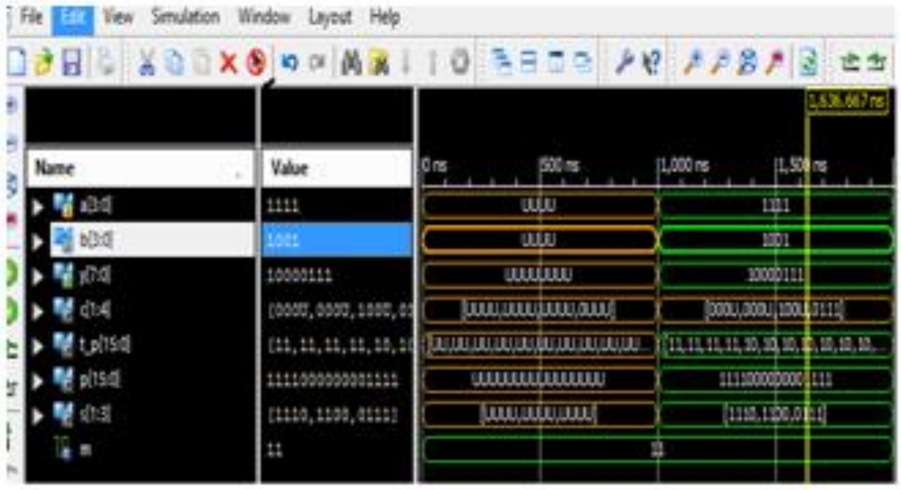


Fig.12 RTL of 8-bit Booth Multiplier

5).Timing Report of 4-bit Column Bypass Multiplier: Timing Summary:
Speed Grade: -4
Maximum combinational path delay: 15.853ns
Timing Detail:



Name	Value
a[3:0]	1111
b[3:0]	1001
y[7:0]	10000111
c[1:4]	[0000,0000,1000,0
t_p[15:0]	[11,11,11,11,10,1
p[15:0]	1111000000001111
s[1:3]	[1110,1100,0111]
m	11

Fig.13 Simulation of 4-bit Column Bypass Multiplier

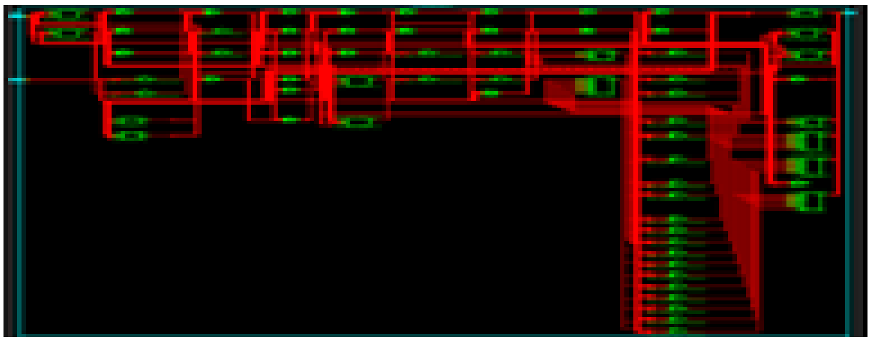


Fig.14 RTL of 4-bit Column Bypass Multiplier

Total 15.853ns (9.220ns logic, 6.633ns route)
(58.2% logic, 41.8% route)

6). Timing Report of 8-bit Column Bypass Multiplier:

Timing Summary:

Speed Grade: -4

Maximum combinational path delay: 29.452ns

Timing Detail:

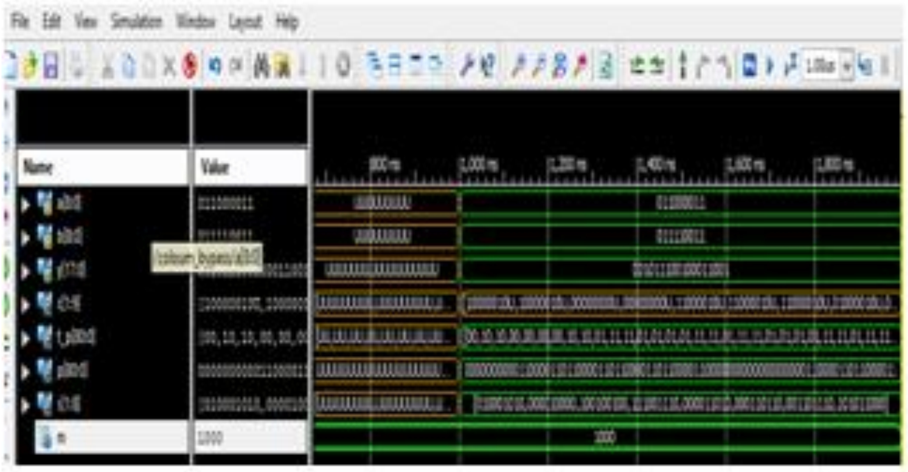


Fig.15 Simulation Result of 8-bit Column Bypass

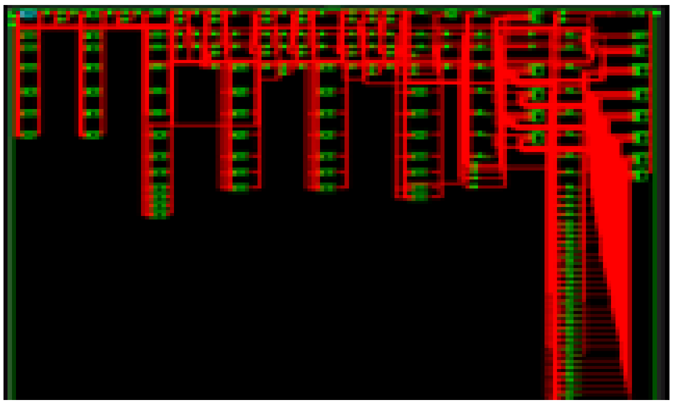


Fig.16 RTL of 8-bit Column Bypass

Total 29.452ns (13.628ns logic, 15.824ns route)
(46.3% logic, 53.7% route)

7). Timing Report of 4-bit Wallace Tree Multiplier:
Timing Summary:
Speed Grade: -4
Minimum period: 4.150ns (Maximum Frequency: 240.964MHz)
Minimum input arrival time before clock: 2.831ns
Maximum output required time after clock: 7.165ns
Total 7.165ns (6.364ns logic, 0.801ns route) (88.8% logic, 11.2% route)

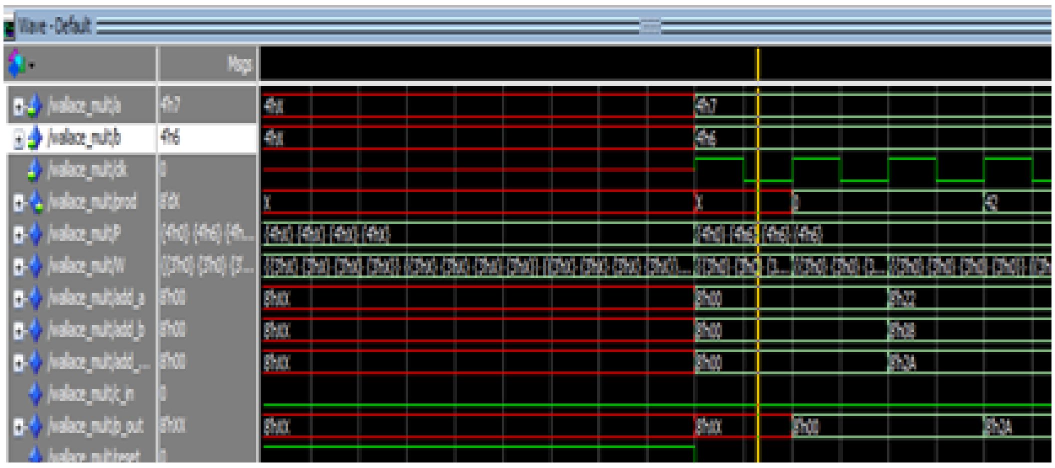


Fig.17 Simulation Result of 4-bit Wallace Tree

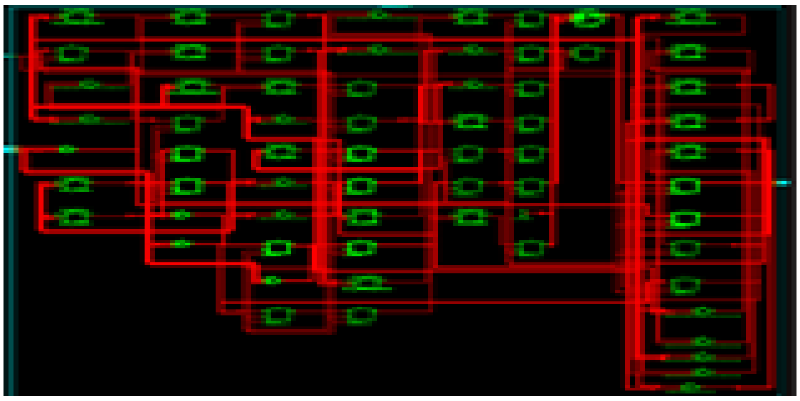


Fig.18 RTL of 4-bit Wallace Tree

8). Timing Report of 8-bit Wallace Tree Multiplier:

Timing Summary:

Speed Grade: -4

Minimum period: 5.947ns (Maximum Frequency: 168.152MHz)

Minimum input arrival time before clock: 2.997ns

Maximum output required time after clock: 7.169ns

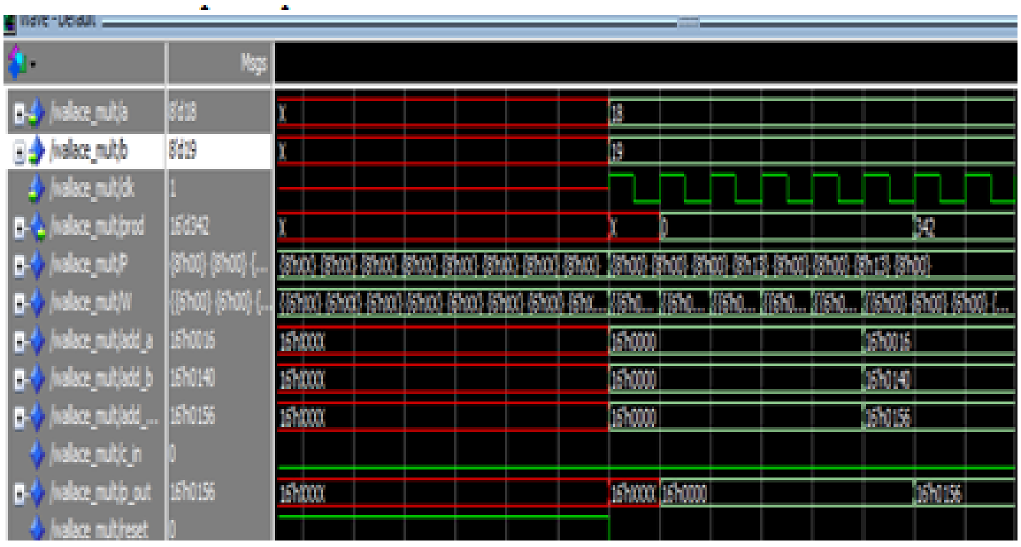


Fig.19 Simulation Result of 8-bit Wallace Tree

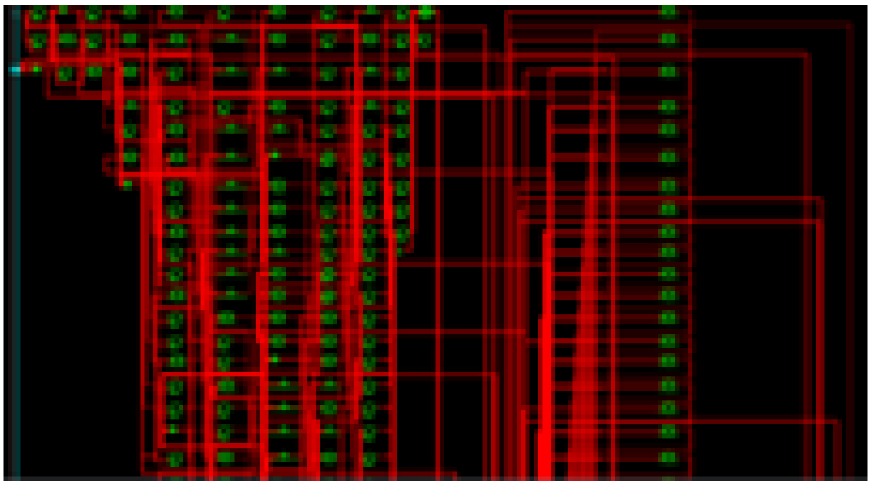


Fig.20 RTL of 8- bit Wallace Tree

Total :7.165ns (6.364ns logic, 0.801ns route)
(88.8% logic, 11.2% route)

IV. Timing Analysis

1). Timing Analysis of 4-bits& 8-bits Multipliers:

Table 1.1 Timing Analysis of 4-bit & 8-bitMultipliers

Mulitpliers	4-Bits Multipliers Timing	8-Bits Mulitpliers Timing
Array Multiplier	17.681ns	32.001ns
Booth Multiplier	10.493ns	15.611ns
Column Bypass Multiplier	15.853ns	29.452ns
Wallace Tree Multiplier	7.165ns	7.169ns

The Table 1.1 shows the comparative analysis of different Mulitpliers.

V. Conclusion

From this analysis table 1.1,it can be easily inferred that, array multiplier has 17.681 ns path delay which is more than column bypass multiplier. Booth multiplier has more output time than the Wallace tree. Wallace tree has minimum input time, so it is fastest.Wallace tree takes 43.4% less time than array multiplier.So we can conclude that. **Wallace < Column Bypass < Booth <Array**

References

[1]. N.Ravi, Dr.T.S.Rao, Dr.T.J.Prasad. "Performance Evaluation of Bypassing Array Multiplier with Optimized Design", International JournalOf Computer Applications (0975-8887), Vol28 No.5, pp. 3 (2011).

[2]. Mahzad Azarmehr., "Multipliers, Algorithm And Hardware Designs", Research Center for Integrated Microsystem, 10 (2008).

[3]. Nishat Bano, "VLSI Design of low power booth multiplier", International Journal of Scientific and Engineering Research Vol 3 issue 2, pp. 1-2(feb-2012) .

[4]. C.Krishnamacharya ,CH. Sravanthi, K. Avinash, K.V. Uma Maheswaar Rao.,"Design of low power 2-D Multiplier using Bypassing

[5]. Technique",International Journal of Innovative Research and Studies ISSN-2319-9725, pp.198 (May 2013).

[6]. Partha Sarathi Mohanty, "Design and Implementation of low power multipliers" WILLOW project, 16, pp.26 (2009).

[7]. Sumit Vaidya and Deepak Dandekar,"Delay-Power Performance Comparison of Multiplier in VLSI Circuit Design", International Journal of Computer Networks and Communication Vol2 No.4, pp.49 (July2010).

[8]. Kuan –Hung Chen and Yuan-SunChu "A low Power Multiplier with the spst" IEEE , VLSI Vol 15 No.7 July 2007.

[9]. H. Lee (A power aware scalable Pipe lines booth multiplier) in proc IEEE int. SOC Conference 2004 PP. 123-12.

[10]. Jorn Stohmann Erich Barke, "A Universal Pezaris Array Multiplier Generator for SRAM-Based FPGAs" IMS- Institute of MicroelectronicsSystem, University of Hanover Callinstr, 34,D30167 Hanover,Germany.Morris Mano, "Computer System Architecture",PP. 346-347, 3rd edition,PHI. 1993.

[11]. Pravinkumar Parate, "ASIC Implementation of 4 Bit Multipliers",,IEEE Computer society. ICETET,2008.25.

[12]. Rajendra Katti, "A Modified Booth Algorithm for High Radix Fixed point Multiplication", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol. 2, pp.: 522-524, Dec. 1994.

Anju Rajput " Timing Analysis of Generic MultipliersIOSR Journal of Electronics and Communication Engineering (IOSR-JECE) 13.1 (2018): 29-38.