ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

Design and Analysis of Various 32bit Multipliers in an Approach towards a Fast Multiplier

Savita Nair[1], R.H.Khade[2], Ajit Saraf[3]

PG Student, Dept. of Electronics, PIIT, Panvel, Maharashtra, India[1]

Assistant Professor, Dept. of Electronics, PIIT, Panvel, Maharashtra, India[2]

Assistant Professor, Dept. of Electronics, PIIT, Panvel, Maharashtra, India[3]

ABSTRACT: This paper is based on the designing and simulation of various multipliers. The design is structured for 32 × 32 bit multiplication. The multipliers designed are an Array Multiplier, a Modified Booth Multiplier, a Wallace tree Multiplier and a Modified Booth Wallace tree Multiplier. The results of these multipliers are compared and analysed to find out the fastest multiplier of all the four multipliers. The design entry is done in VHDL, synthesized using Xilinx ISE 14.4 and simulated using ModelSim SE 6.2c design suite of Mentor Graphics.

KEYWORDS: Array Multiplier, Modified Booth Multiplier, Wallace tree Multiplier, Modified Booth –Wallace tree Multiplier

I.INTRODUCTION

Multiplication is a fundamental operation in most digital signal processing algorithms to perform functions like convolution, filtering and so on.Statics shows that more than 70% of the instructions perform addition and multiplication in most of the microprocessor and DSP algorithms [1]. That is, these operations consume most of the execution time.A In a system comprising of multiplier, the system performance usually determined by the performance of the multiplier as it being the slowest element of all.Hence, optimizing the speed of the multiplier is a major design issue.

Multiplication process can be divided into three steps, namely, generating the partial products, reducing the partial product and the last addition to get the final product. The speed of multiplication can be improved by reduction in the generated number of partial products and/or by increasing the speed at which these partial products are accumulated.

The objective of a good multiplier is to provide a compact utilization, high speed and low power consumption unit.The multiplier is selected based on the required nature of application where it is to be implemented. In this paper first we discussed on various adders such as Ripple Carry adder (RCA), Carry Look-Ahead adder (CLA) and Carry Save adder (CSA) as adders also play a crucial role in designing multipliers. Different methodology of generating the partial products and accumulating them is also explained in detail. The paper deals with comparison of different types of multipliers like Array multiplier, Modified Booth multiplier, Wallace tree multiplier and Modified Booth-Wallace tree multiplier and their comparison based on maximum combinational path delay so as to find the fast multiplier of the four.

II.ADDERS

Multiplication employs addition in its operations, and their hardware is similar if not identical to addition hardware. Hence an adder or multiple adders will be in the critical path of the design, so the performance of a design will be often be limited by the performance of its adders. When looking at other attributes of a chip, such as

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

area or power, it is found that the hardware for addition will be a large contributor to these areas. It is therefore beneficial to choose the correct adder to implement in a design.

According to the presented results in [8], the adder topology which has the best compromise between area, delay and power dissipation are carry look-ahead and hence they are suitable for high performance and low-power circuits. The fastest adders are carry select and carry save adders with the penalty of area. The simplest adder topologies that are suitable for low power applications are ripple carry adder with least gate count and maximum delay. Thus, based on the nature of the application, an adder needs to be selected.In this section, various types of adders are explained.

2.1 Ripple Carry Adder(RCA)

A Ripple Carry Adder is a logical circuit using multiple full adders (FA) to add *N*-bit numbers.Each FA inputs a carry Cin which is the Coutof the previous adder.This kind of adder is a Ripple Carry Adder (RCA), since each carry bit "ripples" to the next full adder.The first FA can be replaced by a HA. The layout of a RCA is simple, which allows fast design time. However, the ripple carry adder is relatively slow as it has to wait for the carry bit that comes from the previous full adder. A RCA for adding two 4-bit numbers is shown in fig. 1 below:

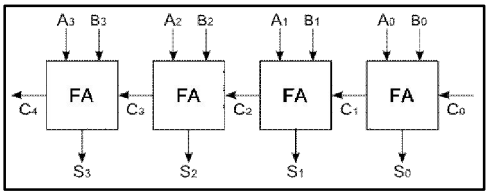


Figure 1. 4-bit Ripple Carry Adder

The sum and carry output of each full adder is given by the following expression:

$$\textbf{Sum } (S_i) = (A_i \text{ xor } B_i) \text{ xor } C_i$$

$$\textbf{Carry } (C_{i+1}) = (A_i \text{ and } B_i) \text{ or } (C_i \text{ and } (A_i \text{ xor } B_i))$$

whereC_{i+1} is the carry out of each of the full adder and is the carry input for the next full adder.

2.2 Carry Look-Ahead Adder(CLA)

The designing of Carry- Lookahead Adder (CLA) is done to overcomethe delay problem caused due to the rippling behaviour of RCA. The speed of CLA is improved by reducing the time taken for calculating the carry bit. The idea of propagating and generating carry is the logic behindCLA. The structure of CLA for 4-bit adder is shown in fig.2.

$$\mathbf{G_i} = A_i \text{ and } B_i$$

$$\mathbf{P_i} = A_i \text{ xor } B_i$$

$$\mathbf{S_i} = P_i \text{ xor } C_i$$

$$\mathbf{C_{i+1}} = G_i \text{or } (P_i \text{ and } C_i)$$

The S_i and C_{i+1} indicate the sum and carry of the i^{th} full adder respectively.

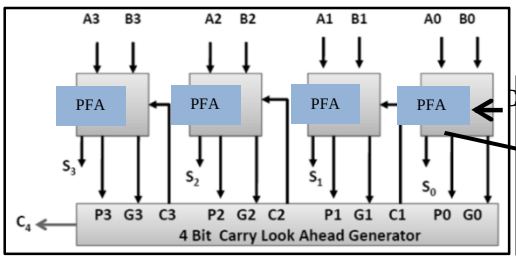


Figure 2. 4-bit Carry Look-Ahead Adder

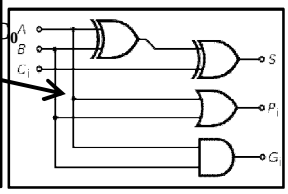


Figure 3. Partial Full Adder

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

The carry-lookahead adder is classified into two blocks: (a) The Partial Full Adder, PFA (shown in fig.3), that generates Si, Pi and Gi. (b) The Carry Look-Ahead Logic that is used to generate the carry-out bits.

2.3 Carry Save Adder (CSA)

A carry-saveaddercomputes the sum of three or more *n*-bit numbers that are in binary. The output of the CSA is same in dimension as the input. Of the two outputs, one is the partial sum bits sequence and the other is carry bits sequence.Final sum is obtained using a carry propagate adder like RCA or CLA or CSelA.

The carry-save unit is made of '*n*' number of full adders, each full addercomputes a carry and the sum from the three inputs. Let the three numbers each of *n* - bit be A, B, and C, the output produced will be a partial sum (PS) and a shift-carry (SC):

$$PS_i = A_i \text{ xor } B_i \text{ xor } C_i$$

$$SC_i = (A_i \text{ and } B_i) \text{ or } (A_i \text{ and } C_i) \text{ or } (B_i \text{ and } C_i)$$

The total sum is calculated by, leftshifting of the carry sequence SC by one place.Appending a 0 in the MSB of thepartial sum PS.Using a RCA or CLA, add these two together and produced result is a *n* + 1-bit value.

III.MULTIPLICATION

Multiplication is a process of adding an integer to itself for a specified number of times. A number (multiplicand) is added to itself a number of times as specified by another number (multiplier) to form a result.Multiplication process have three main steps:

1. Partial product generation.
2. Partial product reduction.
3. Final addition.

For the multiplication of an *n*-bit multiplicand with an *m*-bit multiplier, *m* partial products are generated and product formed is *n* + *m* bits long.

3.1 Partial Product Generation(PPG)

The process of partial product generation can be further classified into two:

i. Simple PPG

In this method, the partial products are generated by multiplying each bit of the multiplier with the multiplicand using logical AND gate.

ii. PPG using Radix-4 Modified Booth Recoding

Partial products are generated with Radix-4 modified Booth recoding. The speed of multiplier can be improved by reducing the number of generated partial products. Using Booth recoding only half the number of partial products is generated when compared with simple PPG which reduces the amount of area occupied by the hardware and the time required for execution.O. L. MacSorley proposed the Modified Booth's Algorithm (MBA) in 1961, as a powerful algorithm for multiplication of signed number, treating both positive and negative numbers uniformly[3] [4].

The encoding of the multiplier bits obtained is the multiple of the multiplicand. The three bits of the multiplier [Y_{i+1}, Y_i and Y_{i-1}] are encoded into [-2X,-X, 0, +X, +2X]. The shifting of multiplicand to the left and then taking the complement gives (-2X), complement of the multiplicand is (-X), multiplicand itself is (X), left shift of multiplicand by one bit is (2X) or taking zeros. The steps of Radix -4 Booth algorithmis as follows:

a. Append '0' on the right of LSB of the multiplier
b. Group the multiplier bits in blocks of three with a bit overlapping from the LSB
c. If the multiplier has odd number of bits, add an extra bit on the left of MSB
d. Examine the each block of multiplier and generate the partial product using the table below:

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

Table 1Radix-4 Modified Booth Recoding

Y_{i-1}	Y_i	Y_{i+1}	Recoded Operation on Multiplicand X
0	0	0	0X
0	0	1	+X
0	1	0	+X
0	1	1	+2X
1	0	0	-2X
1	0	1	-X
1	1	0	-X
1	1	1	0X

e. The new partial product generated are added to the previous partial product by shifting two bits to the left and the multiplier bits are shifted two bits towards right. Initially the partial product is zero.
f. It is then sign extended.
g. The above operations are repeated n/2 times

3.2 Partial Product Reduction

Multiplier require high amount of power and delay during the partial products addition. At this stage, most of the multipliers are designed with different kind of multi operands adders that are capable to add more than two input operands and results in two outputs, sum and carry. The Wallace tree method is used in high speed designs to add the partial products.Wallace Tree helps in reducing the stages of sequential addition of partial products thereby improving speed.

Wallace tree used here is made up of several compressors that take three or more inputs and produce two outputs, of the same dimension as the inputs.The speed, area and power consumption of the multipliers will be in direct proportional to the efficiency of the compressors.There are various types of compressors, namely 3:2, 4:2, 5:2 and so on. A Wallace tree with 4:2 compressors is considered.

i. 3:2 Compressor

A 3-2 compressor takes 3 inputs X_1, X_2,$X_3$and generates 2 outputs, the sumand the carry. The compressor is governed by the basic equation

$$X_1 + X_2 + X_3 = \text{Sum} + 2 * \text{Carry}$$

The 3-2 compressor can also be employed as a full adder cell when the third input is considered as the Carry input from the previous compressor block or X_3 = Cin. The logical expression for sum and carry are as follows:

$$\textbf{Sum} = (X_1 \text{xor } X_2) \text{ xor } X_3$$
$$\textbf{Carry} = (X_1 \text{ and } X_2) \text{ or } (X_3 \text{and } (X_1 \text{ xor } X_2))$$

ii. 4:2 Compressor

The so-called 4: 2 compressor, the 4 numbers are compressed into two numbers, as shown in the following fig.4is a block diagram of a 4: 2 compressor:The 4:2 compressors have been employed in the high speed multipliers to lower the latency of accumulation stage.

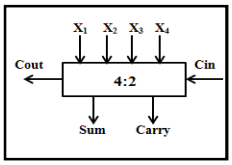


Figure 4. Block diagram of a 4:2 compressor

The 4:2 compressor comprises of X_1, X_2, X_3 and X_4as inputsand as outputs Sum and Carry along with Carry-in (Cin) and a Carry-out (Cout). The output of a previous significant compressor acts as input Cin. Cout depends only on X_1, X_2, X_3 and X_4 and is not dependent on Cinit stops the rippling effect.The output from the compressor Cout, acts as an input to the next compressor that is Cin. A 4:2 compressor the basic equation

$$X_1+X_2+X_3+X_4+C_{in} = \text{Sum} + 2 \times (\text{Carry} + \text{Cout})$$

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

The corresponding logical expression of Sum, Cout and Carry is as follows:

$$S = X1 \oplus X_2 \oplus X_3$$

$$Sum = X1 \oplus X_2 \oplus X_3 \oplus X_4 \oplus C_{in}$$

$$Cout = (X1 \oplus X_2).\ X_3 + X_1.\ X_2$$

$$Carry = (S \oplus X_4).\ C_{in} + S.\ X_4$$

iii. Array Structure using 3:2 Compressors

Array is a straightforward way to accumulate partial products.An array which has 'n' inputs will have 'n-2' levels of carry save adders or 3:2 compressors.It requires 14 levels to add 16 partial products. The final partial sequence of sum and carry can be added using any of the carry propagate adder like CLA or RCA. Fig.5 shows a array structure with 3:2 compressors where 3:2 compressor is a carry-save adder having three multi-bit inputs and two multi-bit outputs and PP indicates the partial products.

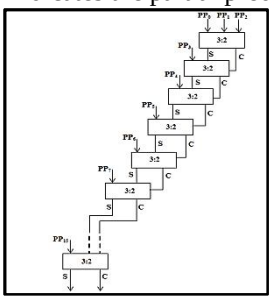


Figure 5.Array structure with 3:2 compressors

iv. Wallace tree Structure using 4:2 Compressors

Wallace tree with 4:2 Compressors is made by a tree like formation of many 4:2 compressor each having four inputs that are multi-bit and produces two outputs which is also multi-bit .The fig. 6 shown below is a Wallace tree structure with 4:2 compressors has 16 partial products given as inputs and a carry and a sum as the output which has the same dimension same as input.

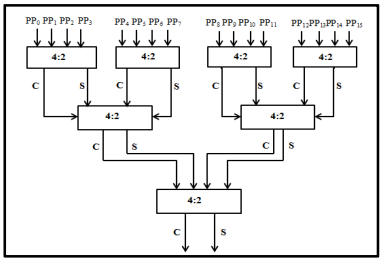


Figure 6. Wallace tree structure with 4:2 compressors

3.3 Final Addition

This stage is also vital for any multiplier because in this stage, addition of operands which has large number of bits is performed and so a fast carry propagate adders like Ripple Carry Adder (RCA) or Carry-look Ahead Adder (CLA) needs to be used as per the requirement.

IV.TYPES OF MULTIPLIERS

Multipliers play avital role in various applications like digital signal processing and many more. As there is technological advancement, the researchers are trying to design multipliers which offer one among the design targets, that is, high speed, less area and low power consumption or a mixture of these features so as to make is suitable to the required nature of the application. An efficient multiplier should have following characteristics:-

- Accuracy: - Should give the correct result.
- Speed: - Perform operation at high speed.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

- Area: - Occupy less number of slices and LUTs.
- Power: - Consume less power.

Multiplication operation involves generation of partial products and their accumulation. In this paper, the aim is to find out a fast multiplier. Reduction in the generated number of partial products (PPs) and/or increase in the speed of adding these PPs can improve the multiplication speed. A reduction in the number of partial products is obtained using Modified Booth Algorithm. The decrease in the number of addition stages can be achieved using Wallace tree. In this section, four multipliers are discussed namely

1. Array Multiplier
2. Modified Booth Multiplier
3. Wallace tree Multiplier
4. Modified Booth-Wallace Tree Multiplier

4.1 Array Multiplier

Partial products are obtained by multiplying each bit of the multiplier with the multiplicand at the output of a partial product generator. These partial products are added using an array structure of 3:2 Compressors.The final adder used is a Ripple Carry Adder to add the sequence of sum and carry obtained as the output of array structure with 3:2 compressors.The block diagram of 32× 32 bit array multiplier is shown below in fig. 7:

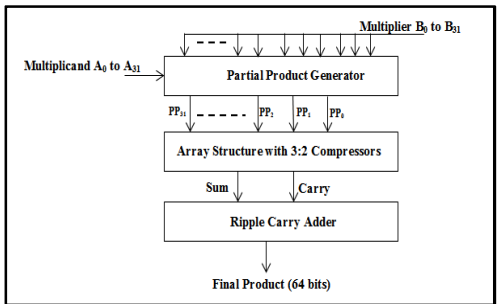


Figure 7. Block diagram of 32× 32 bit Array Multiplier

4.2 Modified Booth Multiplier

The Partial products are generated using the Radix -4 Modified Booth Algorithm in which the multiplier are grouped in blocks of three for recoding and this recoded operation is performed on the multiplicand to obtain partial products and this is done using Modified Booth Recoder(MBR).The no. of partial products is reduced to n/2 where 'n' indicates the available multiplier bits.With inputs of 32 bits, 16 partial products obtained. An array made of 3:2 compressors is used to add these partial products.The final adder used is the ripple carry adder to add the sequence of sum and carry. Thefig. 8 belowdepicts the block diagram a32×32 bit Modified Booth multiplier.

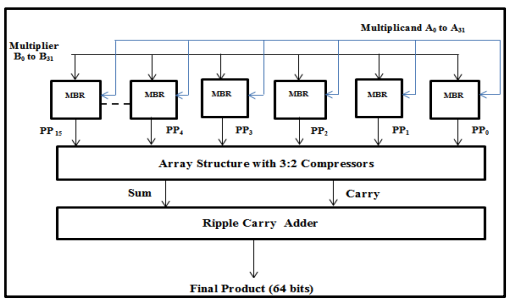


Figure 8.Block diagram of 32× 32 bit Modified Booth Multiplier

4.3 Wallace tree Multiplier

Partial products are obtained by multiplying each bit of the multiplier with the multiplicand in a partial product generator. The multiplier and multiplicand is each of 32 bit, thus generating 32 partial products. These partial products are added using a Wallace tree structure made up of several 4:2 Compressors thus increasing the speed of

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

accumulation. The sequence of sum and carry bits obtained at the end is given to final carry propagate adder that is Carry look-ahead adder. The block diagram of a 32 bit Wallace tree multiplier is shown in fig. 9 below:

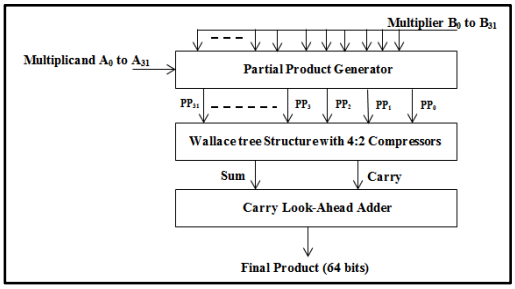


Figure 9.Block diagram of 32× 32 bit Wallace tree Multiplier

4.4 Modified Booth-Wallace Tree Multiplier

The Partial products are generated using the Radix -4 Booth Algorithm.This reduces the partial products generated to half in number, that is, m/2 where 'm' is the available multiplier bits.These partial products are added using a Wallace tree structure made up of several 4:2 Compressors to givea sequence of sum and carry bits that is added using final Carry look-ahead adder.Using Modified Booth algorithm the number of partial products to be added are reduced and using Wallace Tree the number of sequential adding stages are also reduced thus resulting in enhancement of speed. The block diagram of a 32bit Modified Booth-Wallace tree multiplier is shown in fig. 10:

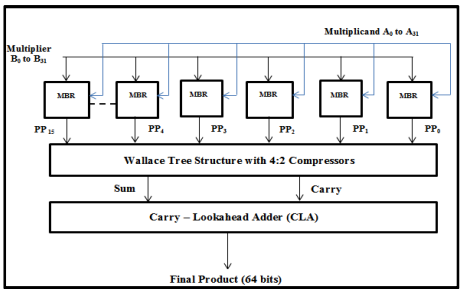


Figure 10.Block diagram of 32× 32 bit Modified Booth-Wallace tree Multiplier

V. RESULT AND DISCUSSION

The multipliers were synthesized in Xilinx ISE 14.4 and simulated using ModelSim 6.2c from Mentor Graphics.The simulation result, device utilization table and the RTL Schematic of all the four multipliers are shown below. Their area and maximum combinational path delay is also compared and analysed.

5.1 Array Multiplier

The fig11.shows the simulation result of the multiplication of two numbers namely in1 and in2 and the product obtained is p, in decimal form, table gives the device utilization summary of the Array multiplier in the form of LUTs and Slices, which specifies the area occupied and the RTL schematic view of multiplier.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

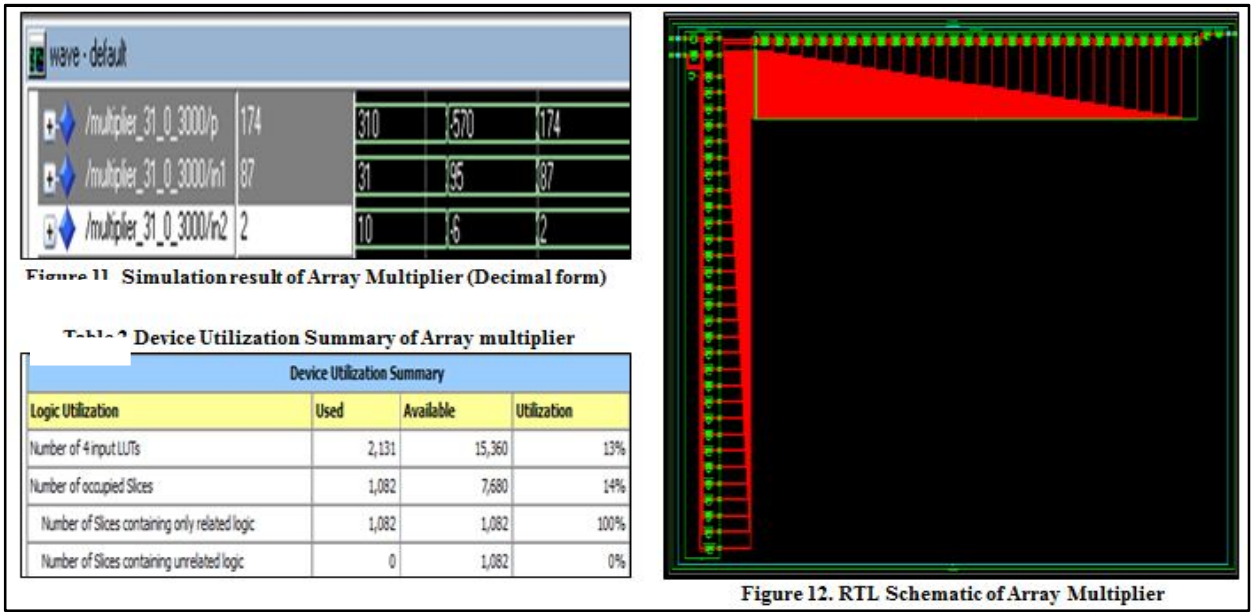


Figure 11 Simulation result of Array Multiplier (Decimal form)

Table 2 Device Utilization Summary of Array multiplier

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	2,131	15,360	13%
Number of occupied Slices	1,082	7,680	14%
Number of Slices containing only related logic	1,082	1,082	100%
Number of Slices containing unrelated logic	0	1,082	0%

Figure 12. RTL Schematic of Array Multiplier

Figure11. Simulation Result, Device Utilization table and RTL Schematic view of Array Multiplier

5.2 Modified Booth multiplier

The simulation result, device utilization summary and RTL schematic of the Modified Booth multiplier are depicted in fig. 12 below. The simulation result is in decimal form obtained from ModelSim simulator.

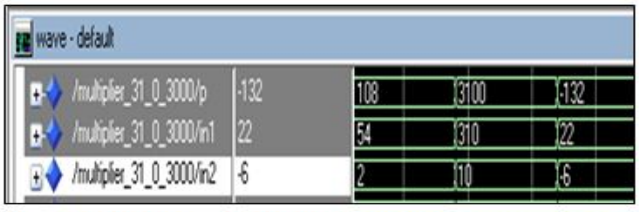


Figure 13. Simulation result of Modified Booth Multiplier (Decimal form)

Device Utilization Summary of Modified Booth multiplier

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	1,858	15,360	12%
Number of occupied Slices	964	7,680	12%
Number of Slices containing only related logic	964	964	100%
Number of Slices containing unrelated logic	0	964	0%

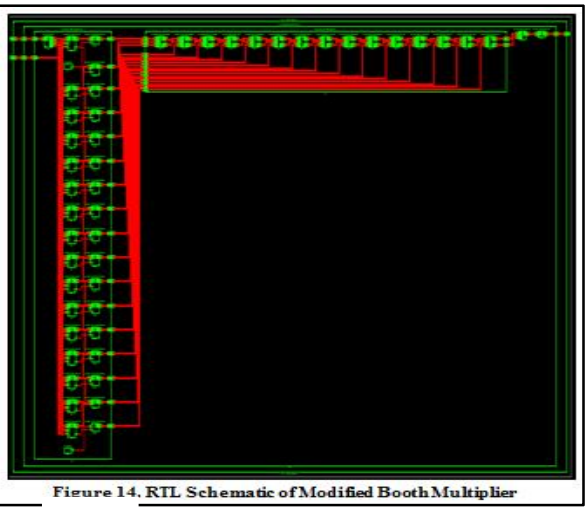


Figure 14. RTL Schematic of Modified Booth Multiplier

Figure12. Simulation Result, Device Utilization table and RTL Schematic view of Modified Booth Multiplier

5.3 Wallace tree multiplier

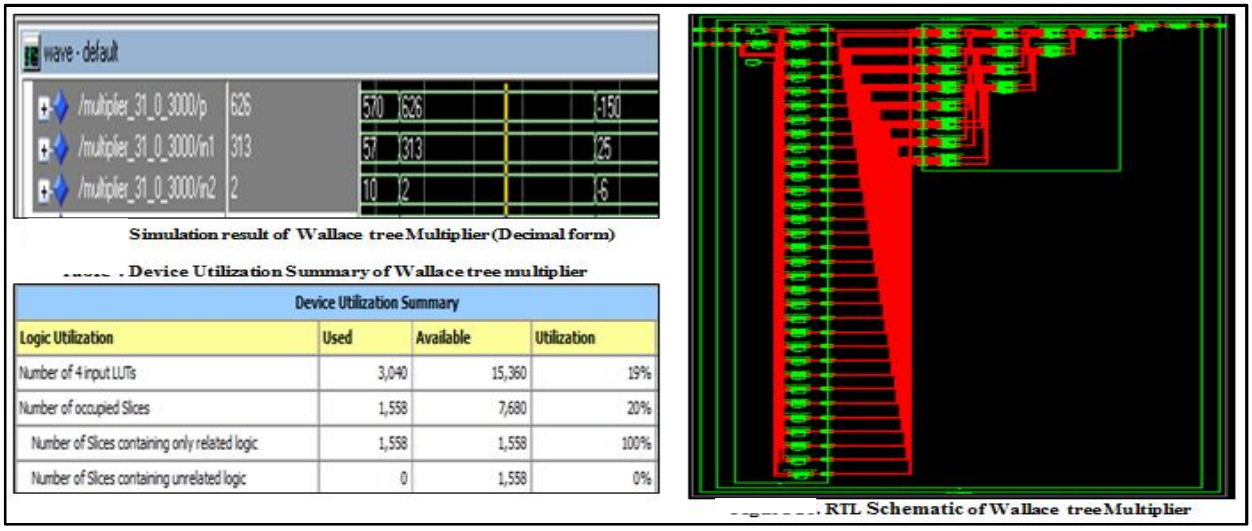
The simulation result, device utilization summary and RTL schematic of the Wallace tree multiplier are depicted in fig. 13 below. The simulation result is in decimal form obtained from ModelSim simulator. The device utilization summary is the indication of the area consumed in form of slices and LUTs.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015



Simulation result of Wallace tree Multiplier (Decimal form)

Device Utilization Summary of Wallace tree multiplier

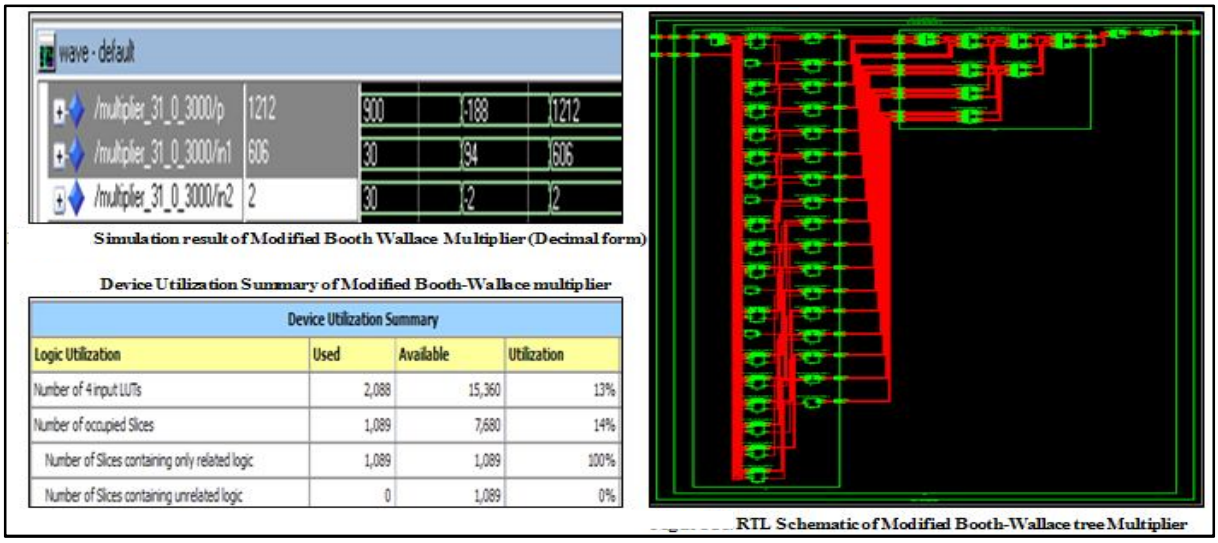
Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	3,040	15,360	19%
Number of occupied Slices	1,558	7,680	20%
Number of Slices containing only related logic	1,558	1,558	100%
Number of Slices containing unrelated logic	0	1,558	0%

RTL Schematic of Wallace tree Multiplier

Figure13. Simulation Result, Device Utilization table and RTL Schematic view of Wallace tree Multiplier

5.4 Modified Booth-Wallace Multiplier

The simulation result, device utilization summary and RTL schematic of the Wallace tree multiplier are depicted in fig. 14 below. The simulation result is in decimal form with in1 and in2 as inputs and p as output, obtained from ModelSim simulator. The device utilization summary is indication of the area consumed in form of slices and LUTs.



Simulation result of Modified Booth Wallace Multiplier (Decimal form)

Device Utilization Summary of Modified Booth-Wallace multiplier

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	2,088	15,360	13%
Number of occupied Slices	1,089	7,680	14%
Number of Slices containing only related logic	1,089	1,089	100%
Number of Slices containing unrelated logic	0	1,089	0%

RTL Schematic of Modified Booth-Wallace tree Multiplier

Figure14. Simulation Result, Device Utilization table and RTL Schematic view of Modified Booth-Wallace tree Multiplier

VI.CONCLUSION

The maximum combinational path delay for a 32×32-bit Modified Booth-Wallace tree multiplier is 104.10ns which is comparatively less than that of Array multiplier, Modified Booth Multiplier, Wallace tree multiplier, that is, 111.016ns, 107.688ns and 104.752ns respectively as shown in fig.19and area comparison in the form of LUTs utilised is also shown in fig. 20 below.

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 7, July 2015

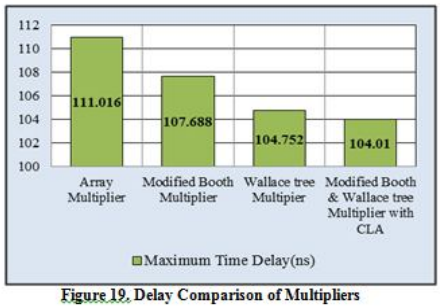


Figure 19. Delay Comparison of Multipliers

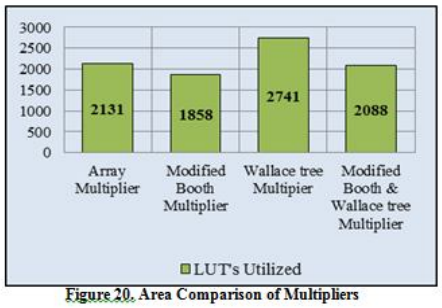


Figure 20. Area Comparison of Multipliers

The delay of Wallace tree and Modified Booth-Wallace tree multiplier is almost same but Wallace tree occupies a much larger area. Booth Multiplier occupies the least area but has high delay when compared to ModifiedBooth-Wallace tree multiplier. Array multiplier offers the highest delay even though it having occupied less area compared to Wallace tree multiplier. So a compromise between delay and area is achieved using Modified Booth-Wallace tree Multiplier which has least delay and occupies comparatively less area thus making it a fast multiplier.

ACKNOWLEDGEMENT

I am thankful to Prof. R.H Khade and Prof. Ajit Saraf of Pillai Institute of Information Technology for their valuable guidance. A great thanks to my friends and family members for their support.

REFERENCES

[1] Soniya, Suresh Kumar, "A Review of Different Type of Multipliers andMultiplier Accumulator Unit,"International Journal of Emerging Trends & Technology inComputer Science (IJETTCS),Volume 2, Issue 4, July – August 2013.
[2] Prasanna Raj P, Rao, Ravi, "VLSI Design and Analysis of Multipliers for Low Power", Intelligent Information Hiding and Multimedia Signal Processing, Fifth International Conference, pp.: 1354-1357, Sept. 2009.
[3] Andrew .D. Booth, "A signed binary multiplication technique," Quarterly Journal for Mechanics and Applied Mathematics, vol. 2, pp. 236-240, 1951.
[4] O.L. MacSorley, "High-speed arithmetic on binary computers," IRE Transaction of Electronic Computers, vol. 49, pp. 67-91, 1961.
[5]Louis P. Rubinfield, "A Proof of the Modified Booth's Algorithm for Multiplication",Computers, IEEE Transactions, vol.24, pp.: 1014-1015, Oct. 1975.
[6]Aparna, Nisha Thomas, "Design and Implementation of High Performance Multiplier using HDL", IEEE Transactions,ISBN:978-1-4673-0270-8, pp.: 1-5, 2012.
[7] C. S. Wallace, "A Suggestion for a Fast Multiplier", IEEE Transactions of Electronic Computers, vol.13, Page(s): 14-17, Feb. 1964.
[8] R.UMA, Vidya Vijayan, M. Mohanapriya, Sharon Paul, "Area, Delay and Power Comparison of Adder Topologies", International Journal of VLSI design & Communication Systems (VLSICS) Vol.3, February 2012.
[9] Prashant Gurjar, Rashmi Solanki, Pooja Kansliwal, Mahendra Vucha, "VLSI Implementation of Adders for High Speed ALU", International Journal of Computer Applications (0975 – 8887)Volume 29– No.10, September 2011.