DESIGN AND PERFORMANCE OF HIGH SPEED MULTIPLIER USING VERILOG HDL

Kaja Sai Bhargava[1], Manisha Jain[2], Poorva Aggarwal[3], Kritika Sharma[4] and Saji M. Antony[5]

[1-5]*ECE Department, Bharati Vidhyapeeth's College of Engineering, New Delhi, India*

***Abstract:** Multiplier are basic building blocks in digital design and also play a vital role in VLSI design. This paper focuses on various multipliers which are Array Multiplier, Vedic Multiplier, Wallace Tree multiplier and Dadda multiplier which were synthesized and simulated using Xilinx ISE 13.2 for spartan 3E family with a speed grade of -5. When compared with the basic multiplier Dadda Multiplier shows a significant improvement in speed.*

Keywords- Array Multiplier (AM), Wallace Tree Multiplier (WT), Vedic Multiplier (VM), Dadda Multiplier (DM) and MAC unit

1. INTRODUCTION

Multipliers play an important role in day to day life. It is most often used in high performance devices such as microprocessor, DSP etc. All the multiplier architecture are designed using basic building blocks such as Half Adder, Full Adder.

So in this paper we design Array Multiplier (AM), Wallace Tree Multiplier (WM), Dadda Multiplier (DM) and Vedic Multiplier (VM) for different number of bits and their delay and various parameters used were compared (number of LUTs used, number of IOBs used and number of slices used). These were built using Verilog HDL and synthesized using Xilinx ISE 13.2

This paper is organized as follows: section I deals with the introduction, section II deals with related work, section III deals with the simulation results and section IV deals with the conclusion.

2. RELATED WORK

Multiplier unit is the key block of digital signal processors as well as general purpose processors that substantially decide the speed of processor. Design of high speed multiplier is need of the day [1]. Any processor's performance is dependent on three important factors namely speed, area and power. A better trade-off between these factors makes the processor, an effective one. Multipliers are the commonly used architectures inside the processor. If the performance of these multipliers are improved then powerful processors can be created in future [2]. In this paper, different types of multipliers array, Wallace tree, Dadda and Vedic multiplier for 4 bit, 8 bit, 16 bit are implemented and they are compared in terms of delay, power, number of IOBs used and number of slices used.

MAC unit is designed using Vedic multiplier with improvement in delay. The design of Multiplier Accumulator Unit (MAC) using Vedic Multiplier is the Ancient Indian Vedic Mathematics technique that has been modified as per technology for improving the performance of mathematical computations [3].

From the performance analysis on delay, power dissipation, transistor count etc. it is observed that full adder with XOR and MUX gives best performance. Using this full adder ripple carry adder is designed and this ripple carry adder is used to design the multiplier to get minimum delay.

3. PROBLEM FORMULATION

The main aim of this paper is to increase the speed of the multiplier. The proposed design integrates high speed Dadda multiplier and mux based adders to minimize the delay. Mux based adders reduce the number of Boolean equation which reduce the delay.

3.1 ARRAY MULTIPLIER

Array multiplier multiplies 2 binary signals using half adders and full adders. It is just like basic multiplication we do on paper. Before the input is fed to the adder it is passed through the AND gates for multiplication. Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift micro-operations. [4] Architecture of 4 bit array multiplier is shown in Fig 1.1.

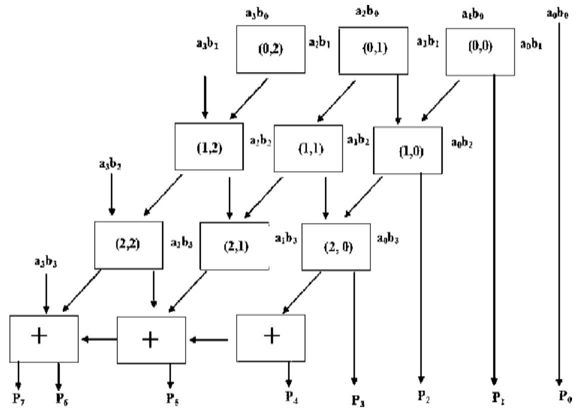


Figure 1.1 4bit Array multiplier [4]

Fig 1.2 shows 8 bit Array multiplier using half adders and full adders.

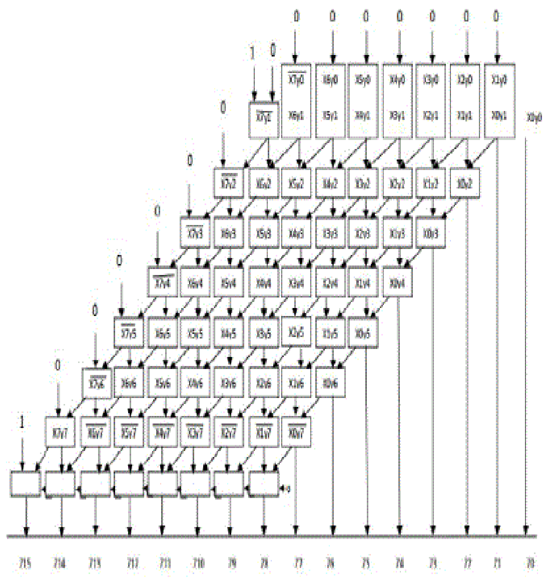


Figure 1.2 8bit Array Multiplier [4]

3.2 VEDIC MULTIPLIER

In Vedic multiplier product generation and addition are done concurrently which makes the multiplication faster as a result delay will be reduce [4] Fig 2.1 shows 2 bit Vedic multiplier using half adders.

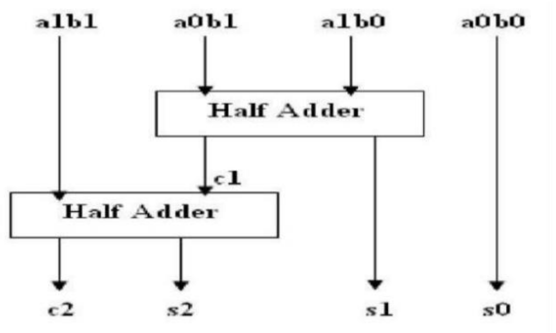


Figure 2.1 2bit Vedic Multiplier [4]

First the LSB are multiplied to get LSB of final product. The LSB of the multiplicand is multiplied with multiplier and MSB with LSB crosswise. Then both of product are passed through the half adder and we get 2bit from LSB of product through the sum of adder. Now carry and product of MSB of both number are passed through the half adder to get the remaining product terms (MSB from carry and 3bit from sum).

The 4 bit Vedic multiplier [4] is designed using 4 2bit Vedic multipliers and adders shown in Fig 2.2.

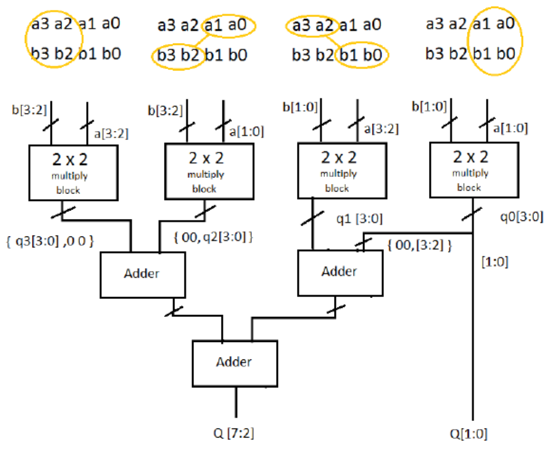


Figure 2.2 4 bit Vedic Multiplier [4]

Architecture of 8 bit Vedic multiplier [4] is shown in Fig 2.3 which is designed using 4 4bit Vedic multipliers and adders.

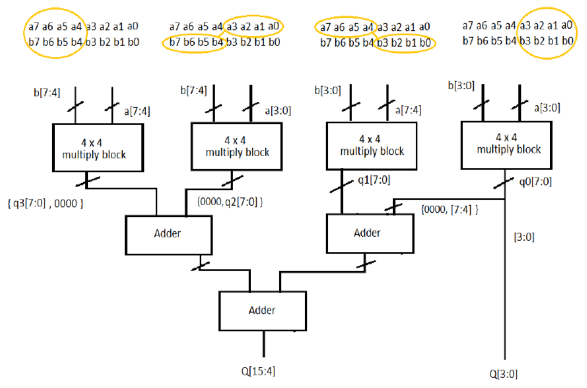


Figure 2.3 8bit Vedic Multiplier [4]

Architecture of 16 bit Vedic multiplier is shown in Fig 2.4 which is designed using 4 8bit Vedic multipliers and adders.

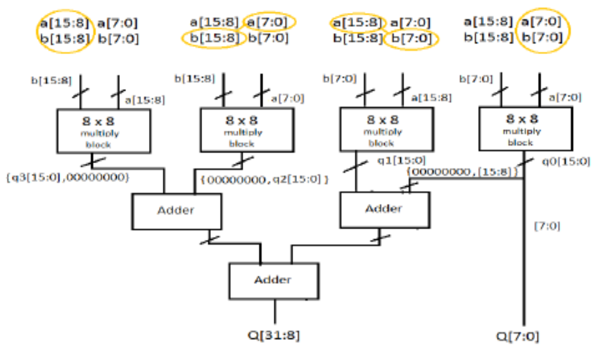


Figure 2.4 16bit Vedic multiplier [4]

3.3 WALLACE TREE MULTIPLIER

Wallace Tree Multiplier uses carry save adder scheme to add partial products generated during the operation in each stage. The delay due to carry is reduce by adding the present state carry into the next state [5]. The Tree Multiplier reduces the time of accumulation of partial product by adding them in parallel [6]. The design flow of Wallace Tree Multiplier is shown in Fig 3

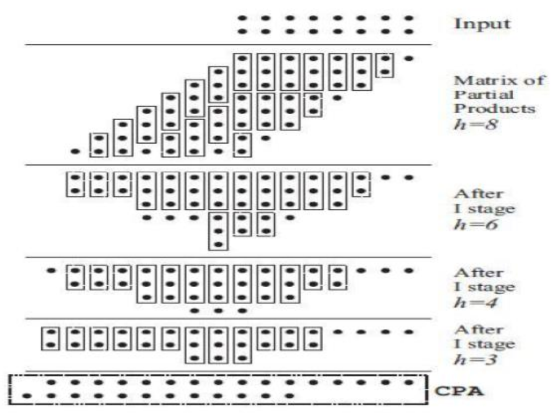


Figure 3 Flow of Wallace Tree Multiplier [5]

3.4 DADDA MULTIPLIER

The Dadda multiplier is a hardware multiplier design, invented by computer scientist Luigi Dadda in 1965. The maximum height of each stage is determined by working back from the final stage which consists of two rows of partial products. The Dadda multiplier performs least reduction in stage compared to Wallace Tree Multiplier [7]. The height of each stage should be order of 2, 3, 4, 6, 9, 13, 19, 28 etc. For 2-bit operand multiplication both Dadda and Wallace tree multipliers have same number of partial product addition levels but when operand bit size increases for 4 bit, 8bit and more the partial product addition levels of Dadda multiplier greatly reduces compared to Wallace tree multiplier and also Dadda multiplier is faster [8]. The design flow of Dadda multiplier is shown in Fig 4.

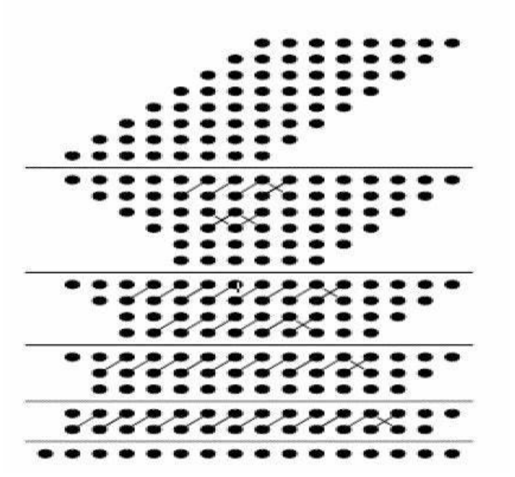


Figure 4 Flow of Dadda Multiplier [5]

3.5 MAC UNIT

MAC unit performs both addition and multiplication. It operates in two stages. Firstly it computes the product of given number and forward the result for second stage operation i.e addition /accumulate. If both are done in single stage then it is called MAC unit [8].Single cycle of MAC unit is shown in Fig 5.

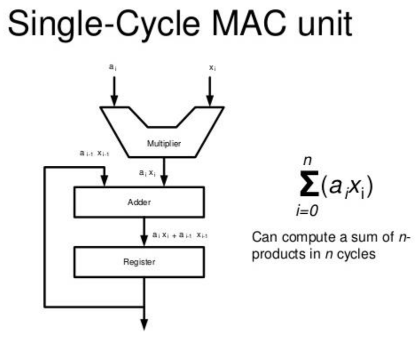
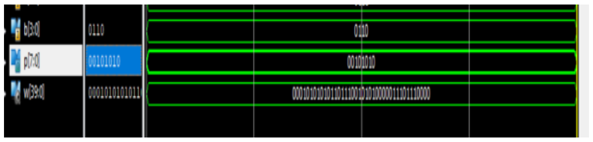

Single-Cycle MAC unit
a_i
x_i
Multiplier
$a_{i-1}\ x_{i-1}$
$a_i\ x_i$
Adder
$a_i x_i + a_{i-1} x_{i-1}$
Register
$\sum_{i=0}^{n}(a_i x_i)$
Can compute a sum of n-products in n cycles

Figure 5 single cycle MAC unit [8]

4. SIMULATION

The proposed design of multipliers is coded using Verilog HDL and the functionally verified using through simulation using Modelsim. First 4 bit of each multiplier designed followed 8bit and 16bit.

A) Simulation of 4 bit Array Multiplier-Consider A and B as the input and P as the output. The multiplier operation is verified with different inputs. A typical simulation of A=0111 and B=0110 is shown in fig and the p=00101010



```
b[3:0]   0110       0110
p[7:0]   00101010   00101010
w[39:0]  00010101010110  0001010101011011100101010000011101110000
```

Figure 6.1 output of 4bit Array Multiplier

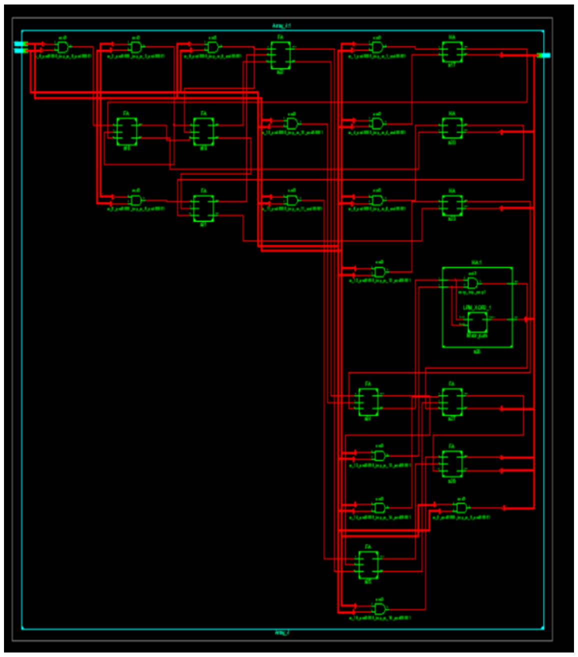
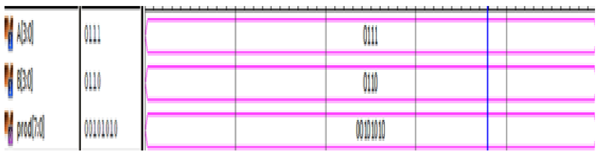


Figure 6.2 RTL Schematic view of 4bit Array Multiplier

B) Simulation of 4bit Vedic multiplier -implemented using 2*2 vedic multiplier. Consider A and B as the input and prod as the output. The multiplier operation is verified with different inputs. A typical simulation of A=0111 and B=0110 is shown in fig and the prod=00101010



```
A[3:0]     0111       0111
B[3:0]     0110       0110
prod[7:0]  00101010   00101010
```

Figure 7.1 output of 4 bit Vedic Multiplier

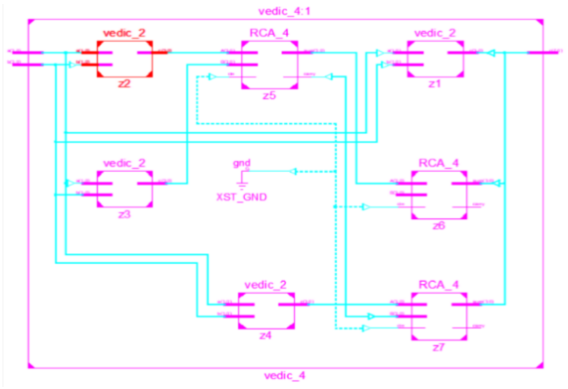


Figure 7.2 RTL Schematic view of 4bit Vedic Multiplier

C) Simulation of 4bit Wallace Tree Multiplier - Consider A and B as the input and prod as the output. The multiplier operation is verified with different inputs. A typical simulation of A=0111 and B=0110 is shown in fig and the prod=00101010

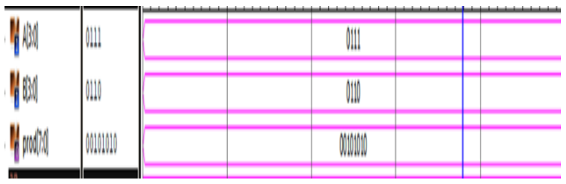


Figure 8.1 output of 4 bit Wallace Tree Multiplier

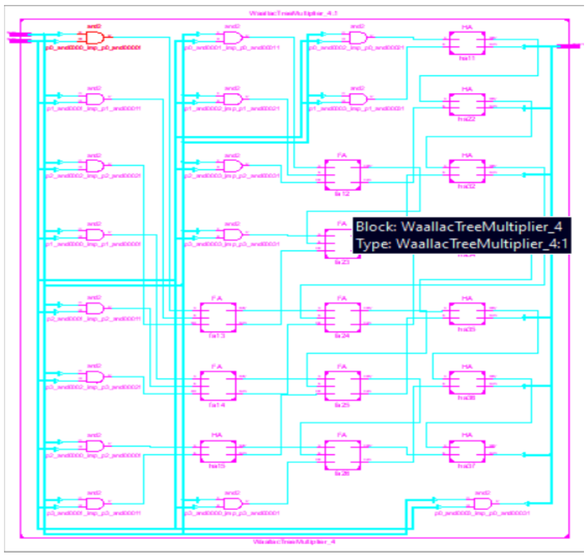

Block: WaallacTreeMultiplier_4
Type: WaallacTreeMultiplier_4:1

Figure 8.2 RTL Schematic view of 4bit Wallace Tree Multiplier

D) Simulation of 4bit Dadda Multiplier- Consider A and B as the input and prod as the output. The multiplier operation is verified with different inputs. A typical simulation of A=0111 and B=0110 is shown in fig and the prod=00101010

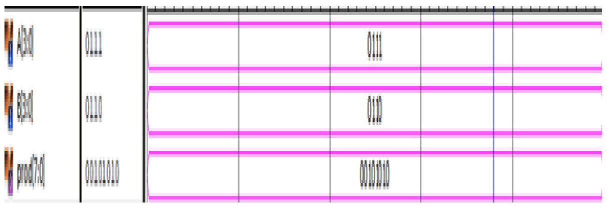


Figure 9.1 output of 4bit Dadda Multiplier

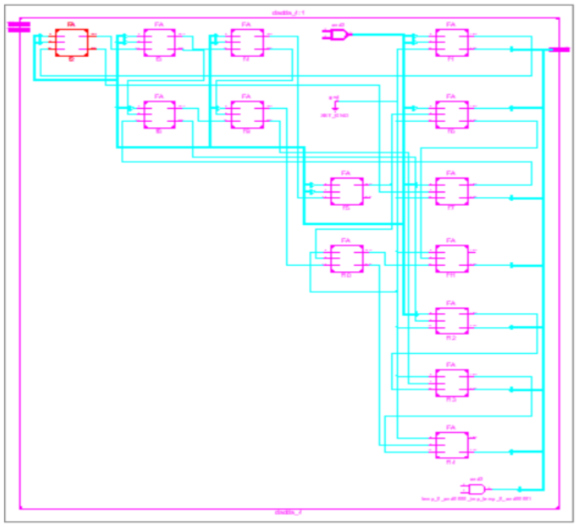


Figure 9.2 RTL Schematic view of 4bit Dadda Multiplier

E) Simulation of 8bit Array Multiplier- Consider A and B as the input and prod as the output. The multiplier operation is verified with different inputs. A typical simulation of A=00001000 and B=00000010 is shown in fig and the prod=0000000000010000

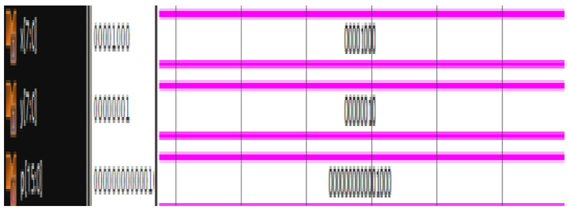


Figure 10.1 output of 8bit Array Multiplier

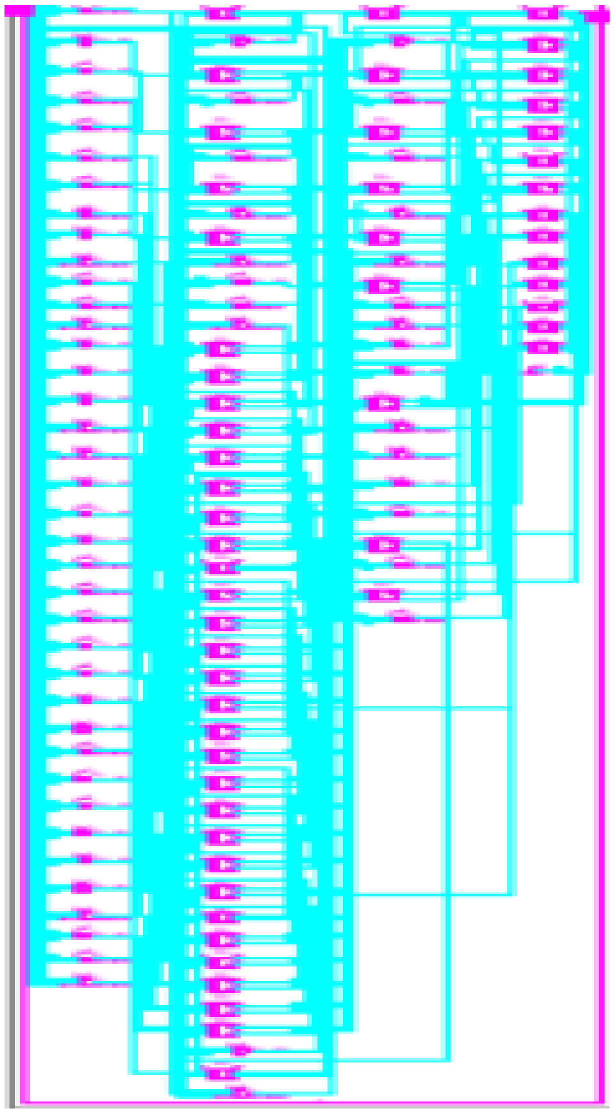


Figure 10.2 RTL Schematic view of 8bit Array Multiplier

F) Simulation of 8 bit Vedic Multiplier- Consider A and B as the input and prod as the output. The multiplier operation is verified with different inputs. A typical simulation of A=10000000 and B=00000010 is shown in fig and the prod=0000000100000000

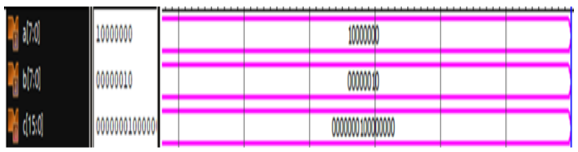


Figure 11.1 output of 8bit Vedic Multiplier

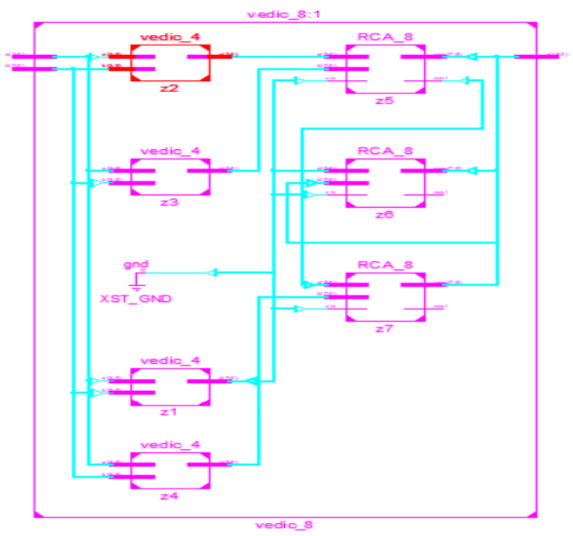


Figure 11.2 RTL Schematic view of 8bit Vedic Multiplier

G) Simulation of 8bit Wallace Tree Multiplier- Consider A and B as the input and product as the output. The multiplier operation is verified with different inputs. A typical simulation of A=00000100 and B=00000001 is shown in fig and the product=0000000000000100

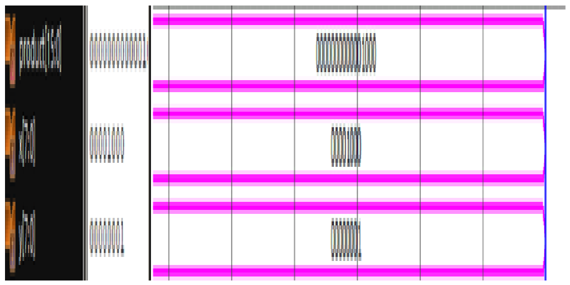


Figure 12.1 output of 8bit Wallace Tree Multiplier

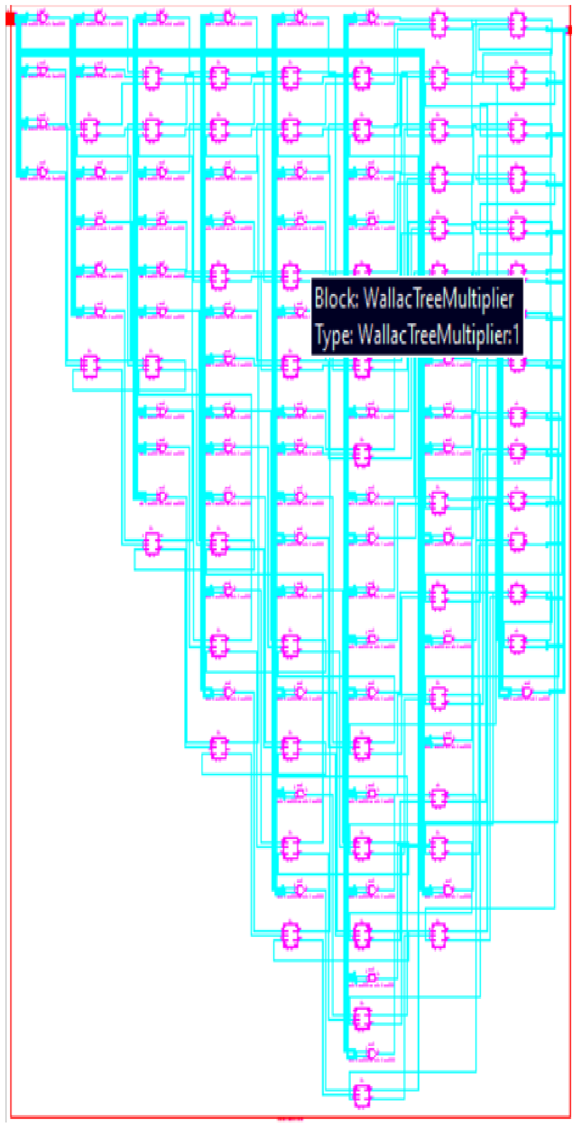


Figure 12.2 RTL Schematic view of 8bit Wallace Tree Multiplier

H) Simulation of 8bit Dadda Multiplier- Consider A and B as the input and product as the output. The multiplier operation is verified with different inputs. A typical simulation of A=255 and B=255 is shown in fig and the product=65025

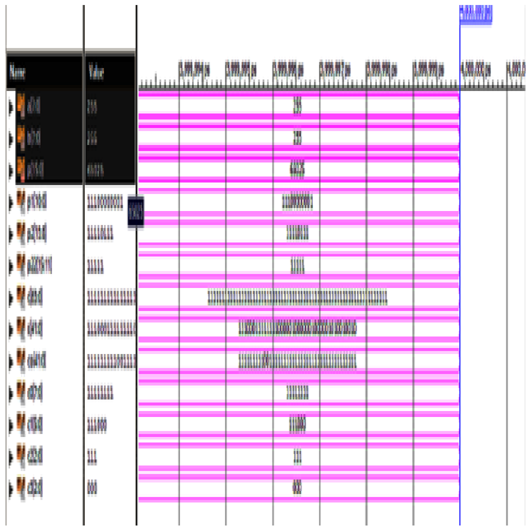


Figure 13.1 output of 8bit Dadda Multiplier

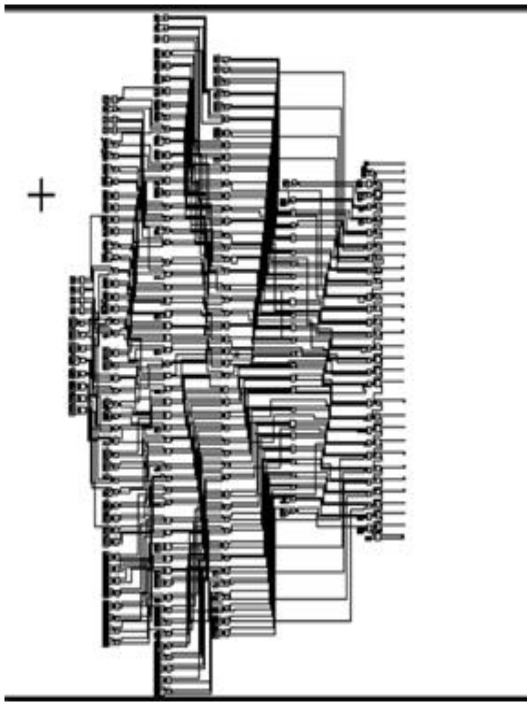


Figure 13.2 RTL Schematic view of 8bit Dadda Multiplier

I) Simulation of 16bit Dadda Multiplier- Consider A and B as the input and prod as the output. The multiplier operation is verified with different inputs. A typical simulation of A=65535 and B=65535 is shown in fig and the prod=4294836225

Figure 14.1 output of 16bit Dadda Multiplier

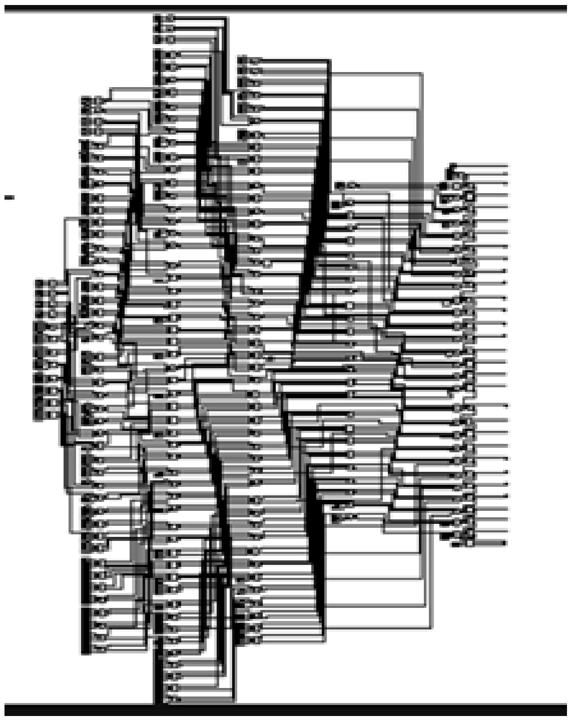
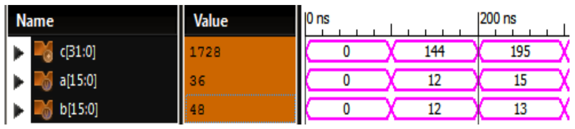


Figure 14.2 RTL Schematic view of 16bit Dadda Multiplier

J) Simulation of 16 bit Vedic Multiplier- Consider A and B as the input and prod as the output. The multiplier operation is verified with different inputs. A typical simulation of A=12 and B=12 is shown in fig and the prod=144



Name	Value	0 ns		200 ns
c[31:0]	1728	0	144	195
a[15:0]	36	0	12	15
b[15:0]	48	0	12	13

Figure 15.1 output of 16bit Vedic Multiplier

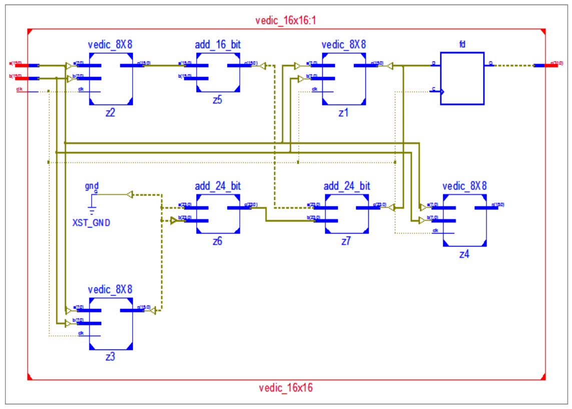


Figure 15.2 RTL Schematic view of 16bit Vedic Multiplier

K) Simulation of MAC unit using 16 bit Vedic multiplier and accumulator-register C contains value of 16 bit Vedic multiplier (1000*10000=1000000) which stored in the temp register. The value temp register keeps on adding to the accumulator value during every positive transition of the clock.

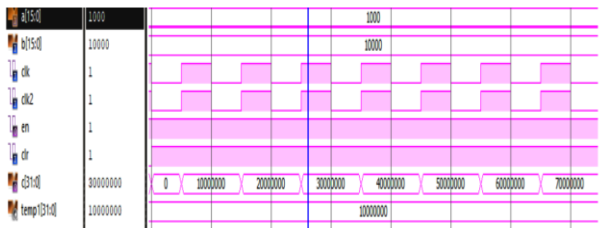


Figure 16.1 output of 32bit MAC unit

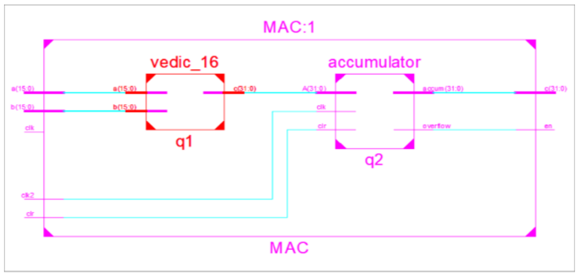


Figure 16.2 RTL Schematic view of 32bit MA unit

L) Performance comparison

Performance comparison of various multiplier in term of number of slices used, number of LUTS used, number of IOB used and delay is shown in table

Table 1. Performance comparison of various multipliers

Parameters	No of slices used out of 768	No of LUTS out of 1536	No of IOB used out of 124	Delay(ns)
ARRAY MULTIPLIER (4 BIT)	19	33	16	19.358
WALLACE TREE MULTIPIER (4 BIT)	18	32	16	15.455
VEDIC MULTIPIER (4 BIT)	18	31	16	16.68
DADDA MULTIPLIER (4 BIT)	18	32	16	15.318
ARRAY MULTIPLIER (8 BIT)	72	126	32	35.024
WALLACE TREE MULTIPIER (8 BIT)	75	130	32	33.192
VEDIC MULTIPIER (8 BIT)	87	152	32	27.156
DADDA MULTIPLIER (8 BIT)	86	149	32	18.404
ARRAY MULTIPLIER (16 BIT)	278	642	64	49.277
WALLACE TREE MULTIPIER (16 BIT)	267	625	64	46.154
VEDIC MULTIPIER (16 BIT)	206	363	56	43.741
DADDA MULTIPLIER (16 BIT)	365	635	64	31.459

Table 2. Performance of MAC unit

Application	Power	Delay	Number of slices
MAC unit	14	62.0ns	48

From the above performance comparison table is can be concluded that Dadda Multiplier have least delay.

The performance comparison in number of slices used is illustrated using bar charts.

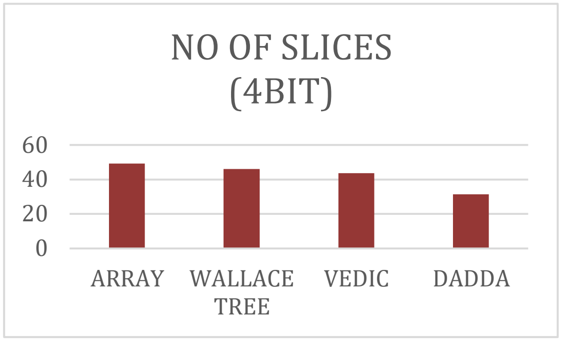


Figure 17.1 Analysis of number of Slices used for 4bit

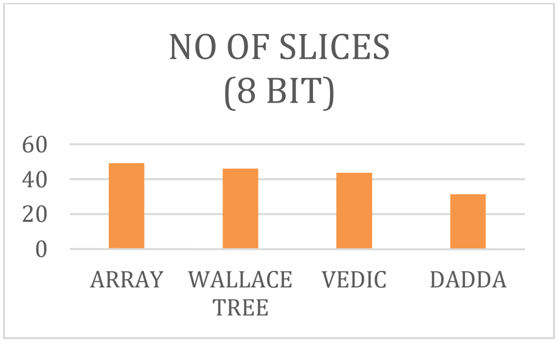


Figure 17.2 Analysis of number of Slices used for 8bit

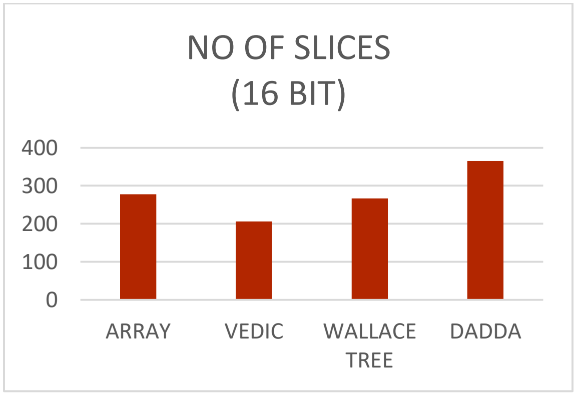


Figure 17.3 Analysis of number of Slices used for 16bit

The performance comparison in number of LUTS used is illustrated using bar charts.

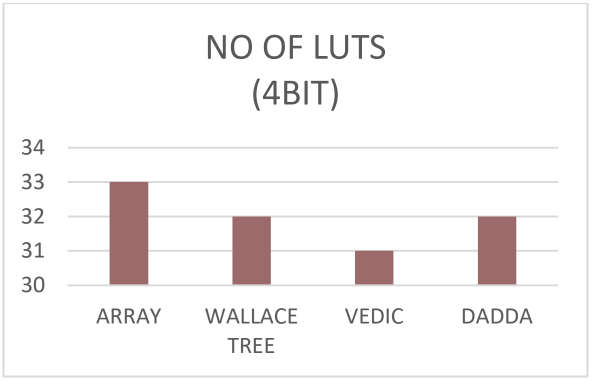


Figure 18.1 Analysis of number of LUT used for 4bit

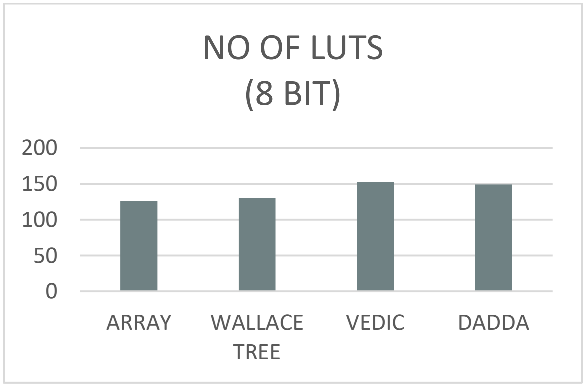


Figure 18.2 Analysis of number of LUT used for 8bit

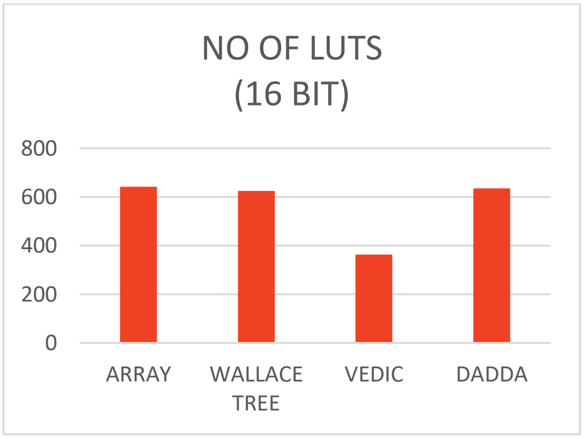


Figure 18.3 Analysis of number of LUT used for 16bit

The performance comparison in number of IOBS used is illustrated using bar charts.

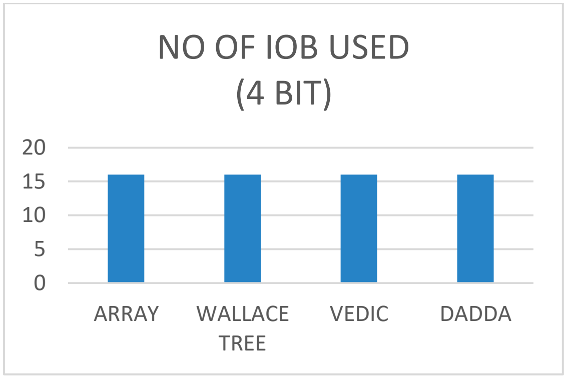


Figure 19.1 Analysis of number of IOB used for 4bit

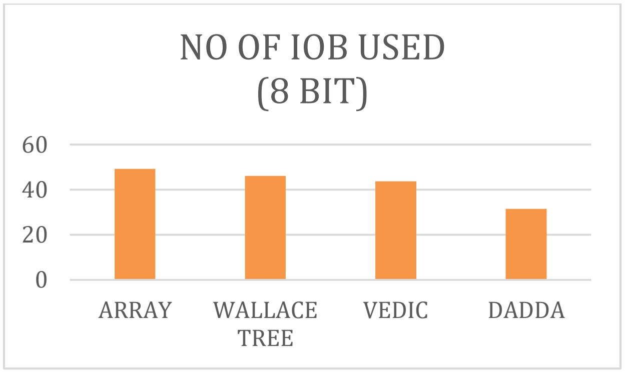


Figure 19.2 Analysis of number of IOB used for 8bit

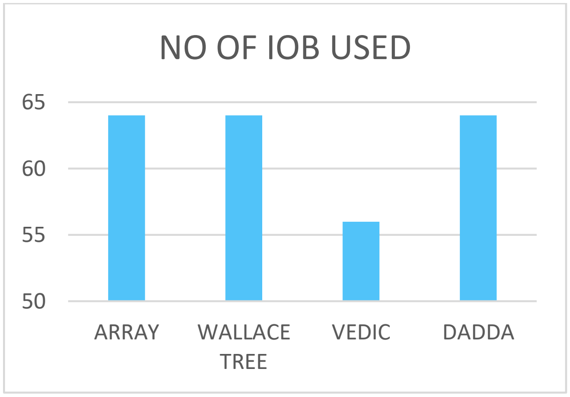


Figure 19.3 Analysis of number of IOB used for 16bit

The performance comparison in delay is illustrated using bar charts.

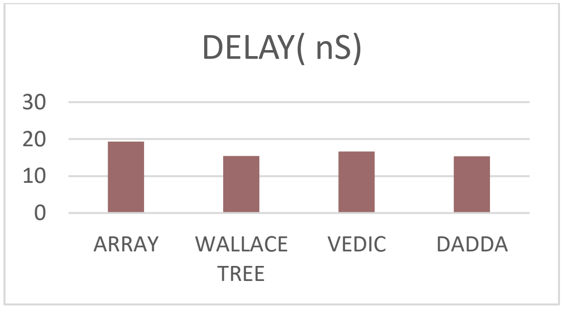


Figure 20.1 Analysis of 4bit multiplier

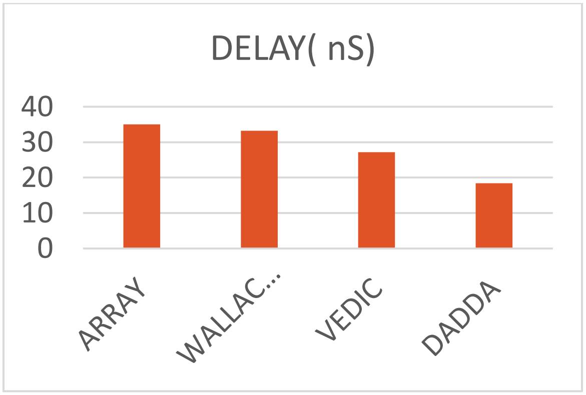


Figure 20.2 Analysis of delay for 8bit multipliers

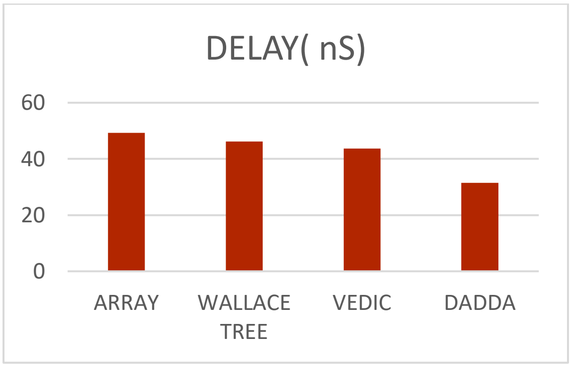


Figure 20.3 Analysis of delay for 16bit

5. CONCLUSION

Various multiplier are designed and simulated using Verilog in Xilinx ISE 13.2 with Spartan 3E family with a speed grade of -5. Out of all the multipliers Dadda Multipliers have least delay and least number of LUTS are used. Further this work can be extended to implement 32bit, 64bit multipliers. These architectures of the multiplier can be used in high performance application such as FFT.

6. REFERENCES

[1] Deepak kurmi, V.B baru"Design of high speed 32 bit multiplier using vedic mathematics and compressors" International Journal of Science and Research (IJSR) (2013).

[2] S. Koushighan, K. Hariharan and V. Vaithiyanathan "Design of an Optimized High Speed Multiplier Using Vedic Mathematics" Contemporary Engineering Sciences, Vol. 7, 2014, no. 9, 443 - 448

HIKARI Ltd

[3] S.Tulasi,Ch.Rajesh Babu"MAC Design Using Vedic Multiplier"International journel and magazine of engineering ,technology,Management and research.

[4] Maroju SaiKumar, P. Samundiswary, "design and performance analysis of various multipliers using Verilog HDL.

[5] Chris Y.H. Lee, Lo Hai Hiung, Sean W.F. Lee, Nor Hisham Hamid, "A Performance Comparison Study on Multiplier Designs", Proceedings of IEEE International Conference on Intelligent and Advanced Systems, Malaysia, pp.1-6, June 2010.

[6] Sumant Mukherjee, Dushyant Kumar Soni, "Energy Efficient Multiplier for High Speed DSP Application", IJCSMC, Vol. 4, Issue. 6, June 2015, pg.66 – 75.

[7] N.Prathima*, K.HariKishore," Design of a low power and high-performance digital multiplier using a novel 8T adder", International Journal of Engineering Research and Applications, (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 3, Issue 1, January -February 2013, pp.1832-1837.

[8] Maroju Sai Kumar, D Ashok Kumar, "Design And Performance Analysis Of MAC unit"

[9] Saji M Antony "Design Of high Speed Vedic Multiplier using Mux Based Adders"

[10] R Uma, "Logic Optimisation Using Technology Independent MUX Based Adders in FPGA"