Design of High Performance Digital Multiplier

Vaishakhi Savalia
Student M.Tech
Department of Electrical Engineering Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda, Vadodara, India

Sunil Patel
Assistant Professor
Department of Electrical Engineering Faculty of Technology and Engineering
The Maharaja Sayajirao University of Baroda, Vadodara, India

Rohit Negi
Assistant Professor
Department of Electrical Engineering
School Of Engineering and Technology, Navrachana University, Vadodara, India

ABSTRACT-Thispaper presents design of a high speed digital multiplier using Vedic mathematics. Digital multipliers are an integral part of Digital signal processing systems or DSP processors. Vedic mathematics is based on 16 sutras (formulas) out of which Urdhva tiryakbhyam (UT) sutra is used to simplify multiplication process and to perform multiplication with minimum amount of delay. All three multiplier algorithmsare coded in VHDL, simulated using Xilinx Vivado, synthesized and ImplementedusingXilinxvivado for nexys4 ddr artix-7 (xc7a100t-1csg324c) FPGA. The design is implemented for 8x8 bit, 16x16 bit, 32x32 bit and 64x64 bit multiplication. The results for each design are compared and analyzed which shows that multipliers based on algorithm 3 are comparatively faster and also requires less number of LUT's for its implementation.

Keywords-Vedic Multiplier, Urdhva Tiryakbhyam, LUT, VHDL.

INTRODUCTION

Multiplication is the fundamental arithmetic operation in digital signal processing. Most signal Processing and data processing applications involve multiplications. So speed of Multiplier is very important factor to determine the efficiency of the circuit design.To make faster mechanism of the system, speed is dominant factor. There are some factors which determine the efficiency of the system, are speed, area, power requirement. Therefore, designers are looking forward to the new algorithms and different methods to speed up the arithmetic operations.

Vedic mathematics has a several number of sutras to solve arithmetic operations in easy and faster way. It was discovered by Indian mathematician JagadguruShriBharathi Krishna Tirthaji. Vedic maths is a very interesting field. Veda is a Sanskrit word which means 'knowledge'.

Vedic mathematics has been formulated on sixteen sutras and thirteen sub-sutras. These sutras offer magical short cut methods to all basic mathematical operations. All the advantages drives from the fact that Vedic mathematics approach is totally different and considered very close to the way a human mind works. Vedic mathematics can be applied to every branch of mathematics including arithmetic, algebra and geometry [3].

Out of the many Sutras available, the proposed algorithms are based on Urdhva-Tiryakbhyam technique with certain improvisations while designing. As it had been proved that Vedic multipliers are fast when compared with normal array multiplier or booth multiplier, the work here presents comparison between three different algorithms using Vedic method for 8-bit, 16-bit, 32-bit and 64-bit Vedic multipliers.

ILLUSTRATION OF URDHVA TIRYAKBHYAM SUTRA

In UrdhvaTiryakbhyamsutra[3], the 4x4 multiplication has been done in a single line as shown in Fig. 1, whereas in the conventional method, four partial products have to be added to obtain the result. Thus, by using UTSutra in binary multiplication, the number of steps required to calculate the final product, will be reduced and hencethere is a reduction in computational time and increase in speed of the multiplier.

Consider two 4-bit binary numbers a3a2a1a0 and b3b2b1b0.

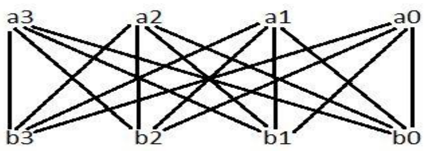


Fig. 1. Illustration of UT sutra for 4 bit numbers

The UrdhvaTiryakbhyam Sutra is also known as ‘Vertical and Cross-wise’ Technique.

A. *Algorithm 1: Design of 4x4 bit Vedic Multiplier[3]*

The 4-bit Vedic multiplier is designed using UrdhvaTiryakbhyam sutra for partial productaddition. This is a normal Vedic multiplier where the carry from each partial product addition is given to the next partial product to generate the final product. For this proposed algorithm partial products(P7P6P5P4P3P2P1P0) generatedaregiven by the following equations:

- P0 = a0b0
- P1 = a1b0 + b0a1
- P2 = a2b0 + b2a0 + a1b1 + carry from P1
- P3 = a3b0 + b3a0 + a2b1 + b2a1 + carry from P2
- P4 = a3b1 + b3a1 + a2b2 + carry from P3
- P5 = a3b2 + b3a2 + carry from P4
- P6 = a3b3 + carry from P5
- P7 = carry from P6(1)

The gate level description of 4-bit Vedic multiplier isshown in Fig. 2. It gives detailed description of how each partial product is obtained in the multiplication process. It uses half adders and full adders.

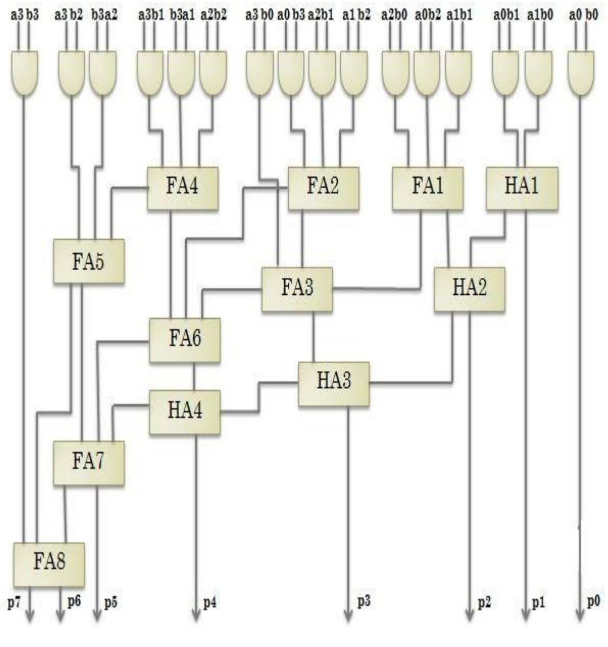


Fig. 2. Gate Level description of normal 4-bit Multiplier

B. *Algorithm2: Design of 4x4 bit Vedic Multiplier*

This is a modified Vedic Multiplier where the carries are not only rippled to the next partial product bit calculations but also to the subsequent bits using carry skip technique so as to reduce the carry propagation delay [1].

The Vedic multiplier equations given in (1) are modified in the proposed multiplier as follows:

- P0 = a0b0
- P1 = a1b0 + b0a1
- P2 = a2b0 + b2a0 + a1b1 + a0a1b0b1
- P3 = a3b0 + b3a0 + a2b1 + b2a1 + carry from P2
- P4 = a3b1 + b3a1 + a2b2 + carry from P2 + carry from P3
- P5 = a3b2 + b3a2 + carry from P3+ carry from P4
- P6 = a3b3 + a1a2b1b2 + carry from P4
- P7 = carry from P6(2)

The products P0, P1, P3 and P7 in (2) are same as in (1) but the other equations have been modifiedto incorporate carry skip technique in the adder blocks of Fig.3. In P2 a new term a0b0a1b1 is added so that all the fourterms are ready for addition at the same time to reduce the delay in waiting for carry from the previous stage. In P4 andP5 the carry from two previous stages are added thus carry propagation time is reduced. For example if all four terms ofP2 are 1, the carry is directly transferred to P4. Similarly P6 is modified by adding carry from P4 and term a1a2b1b2 [1].

The gate level description of modified Vedic multiplier is shown in Fig.3. The design methodology combinesUTsutra [3] of Vedic maths with carry skip technique. The Fig.3givesdetailed description of how each partial product is obtained inthe multiplication process. If any stage produces a twobit carry, then the immediate stage is skipped and the carry is given to the subsequent stages after the immediate stagethus increasing the speed. It uses only half adders and the input bits are added using carry save method.

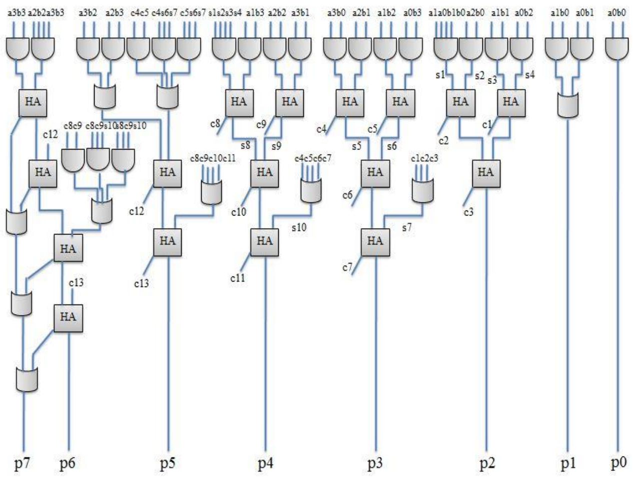


Fig. 3. Gate Level description of modified 4-bit Multiplier

C. *Algorithm 3: Design of 8x8 bit Vedic Multiplier*

UrdhvaTiryakbhyam sutra for two 8-bit numbers is as shown in Fig. 4.

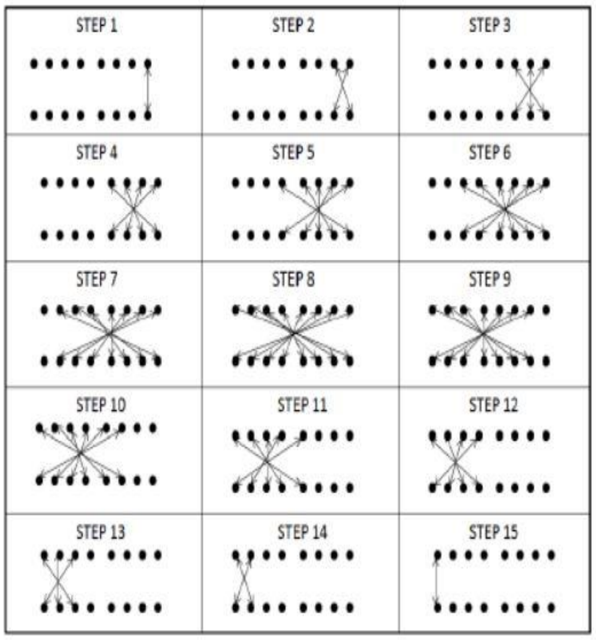


Fig. 4. Illustration of UT sutra for 8-bit numbers

The 8-bit Vedic multiplier is designed using UrdhvaTiryakbhyam sutra for partial productaddition. For this proposed algorithm partial products(P7P6P5P4P3P2P1P0) generated are given by the following equations:

- P0 = a0b0
- P1 = a1b0+b0a1
- P2 = a2b0+b2a0+a1b1+carry from P1
- P3 = a3b0+b3a0+a2b1+b2a1+carry from P2
- P4 = a4b0+b4b0+a3b1+b3a1+a2b2+carry from P3
- P5=a5b0+b5a0+a4b1+b4a1+a3b2+b3a2+carry from P4
- P6 = a6b0+b6a0+a5b1+b5a1+a4b2+b4a2+a3b3+carry from P5
- P7 = a7b0+b7a0+a6b1+b6a1+a5b2+b5a2+a4b3+b4a3+carry from P6
- P8 = a7b1+b7a1+a6b2+b6a2+a5b3+b5a3+a4b4+carry from P7
- P9 = a7b2+b7a2+a6b3+b6a3+a5b4+b5a4+carry from P8
- P10 = a7b3+b7a3+a6b4+b6a4+a5b5+carry from P9
- P11 = a7b4+b7a4+a6b5+b6a5+carry from P10
- P12 = a7b5+b7a5+a6b6+carry from P11
- P13 = a7b6+b7a6+carry from P12
- P14 = a7b7+carry from P13
- P15 = carry from P14 (3)

DESIGN OF 2NX2N MULTIPLIER USING NXN

The architecture of 2n-bit Vedic Multiplier is shown in Fig. 5. The architecture consists of four n-bit multipliers used for calculating partial products. [2]

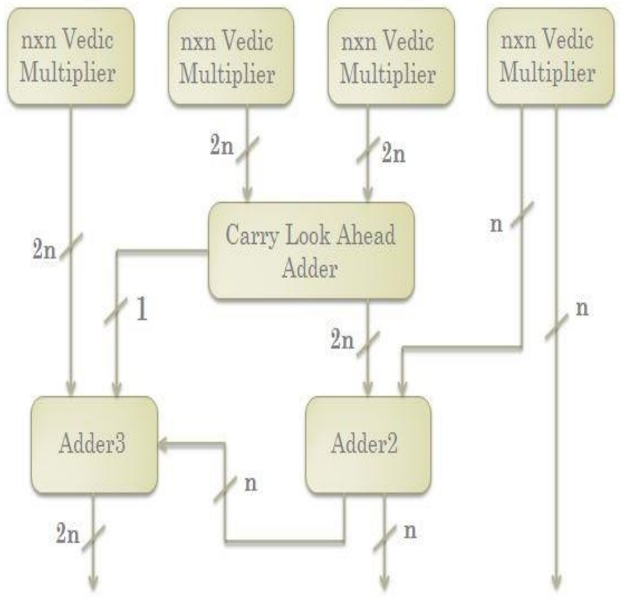


Fig. 5. Design of 2n-bit numbers

The above method is used to design higher bit multipliers such as 16-bit multiplier using 8-bit module, 32-bit multiplier using 16-bit module and 64-bit multiplier using 32-bit module.

The above three algorithms are used to design 8 bit, 16 bit, 32 bit and 64 bit multipliers using the design shown in Fig. 5. Here carry look ahead technique is used to calculate partial products at the first level. This technique helps in parallel generation of carry bits and therefore speeds up the addition process. Similarly adder2 and adder3 are used from reference [2], to calculate the final result.

SIMULATION WAVEFORMS AND RESULTS

A. *Waveforms*

The output of an 8 bit multiplier is verified for the following set of inputs which are generated using VHDL test bench. The simulation result is shown in the Fig. 6 below.

CASE - 1: Inputs x = 55 h,y = aa h

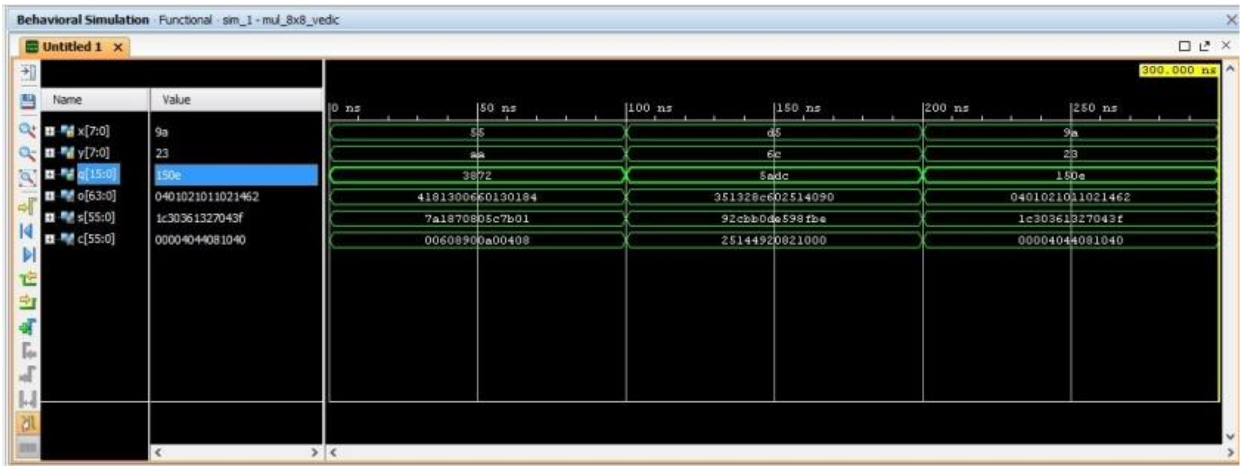
Multiplier's output q = 3872 h

CASE - 2: Inputs x = d5 h ,y = 6c h

Multiplier's outputq = 5adc h

CASE - 3: Inputs x = 9a h, y = 23 h

Multiplier's outputq = 150e h



Behavioral Simulation · Functional · sim_1 - mul_8x8_vedic

Untitled 1

Name	Value			
x[7:0]	9a	55	d5	9a
y[7:0]	23	aa	6c	23
q[15:0]	150e	3872	5adc	150e
o[63:0]	0401021011021462	4181300660130184	351328c602514090	0401021011021462
s[55:0]	1c30361327043f	7a187080 5c7b01	92cbb0de598fbe	1c30361327043f
c[55:0]	00004044081040	006089 00a00408	25144920821000	00004044081040

Fig. 6. **Waveform of 8-bit Multiplier**

The simulation result for 16-bit multiplier is shown in the Fig.7 for 3 different inputs given below.

CASE - 1: Inputs x = 5555 h, y = aaaa h

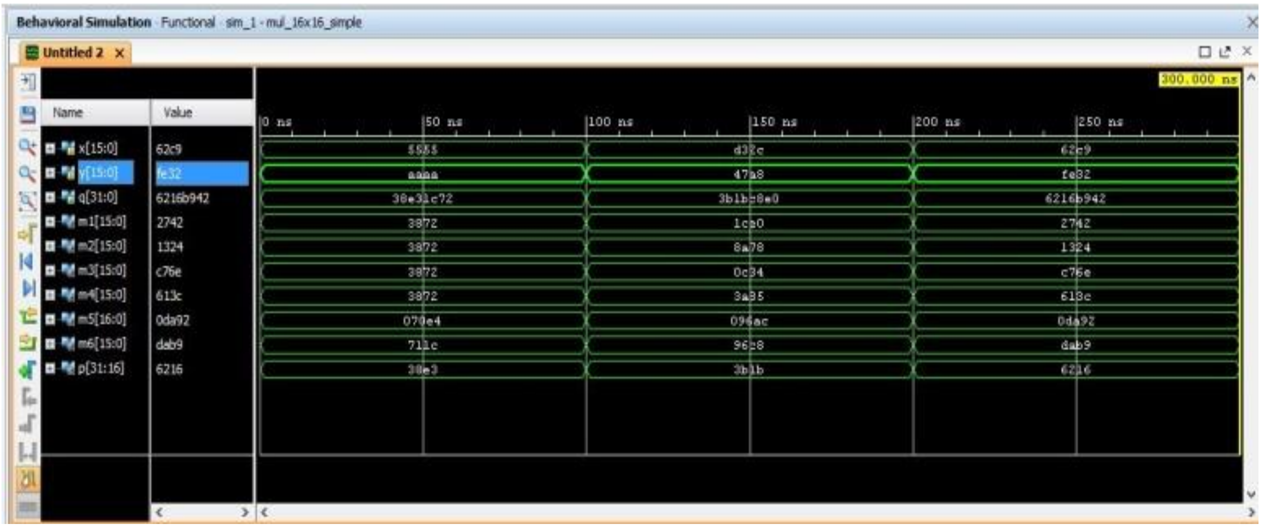
Multiplier’s output q = 38e31c72 h

CASE - 2: Inputs x = d32c h, y = 47a8 h

Multiplier’s output q = 3b1bc8e0 h

CASE - 3: Inputs x = 62c9 h, y = fe32 h

Multiplier’s output q = 6216b942 h



Behavioral Simulation · Functional · sim_1 - mul_16x16_simple

Untitled 2

Name	Value			
x[15:0]	62c9	5555	d32c	62c9
y[15:0]	fe32	aaaa	47a8	fe32
q[31:0]	6216b942	38e31c72	3b1bc8e0	6216b942
m1[15:0]	2742	3872	1cb0	2742
m2[15:0]	1324	3872	8a78	1324
m3[15:0]	c76e	3872	0c34	c76e
m4[15:0]	613c	3872	3a35	613c
m5[16:0]	0da92	070e4	096ac	0da92
m6[15:0]	dab9	711c	9628	dab9
p[31:16]	6216	38e3	3b1b	6216

Fig. 7. **Waveform of 16-bit Multiplier**

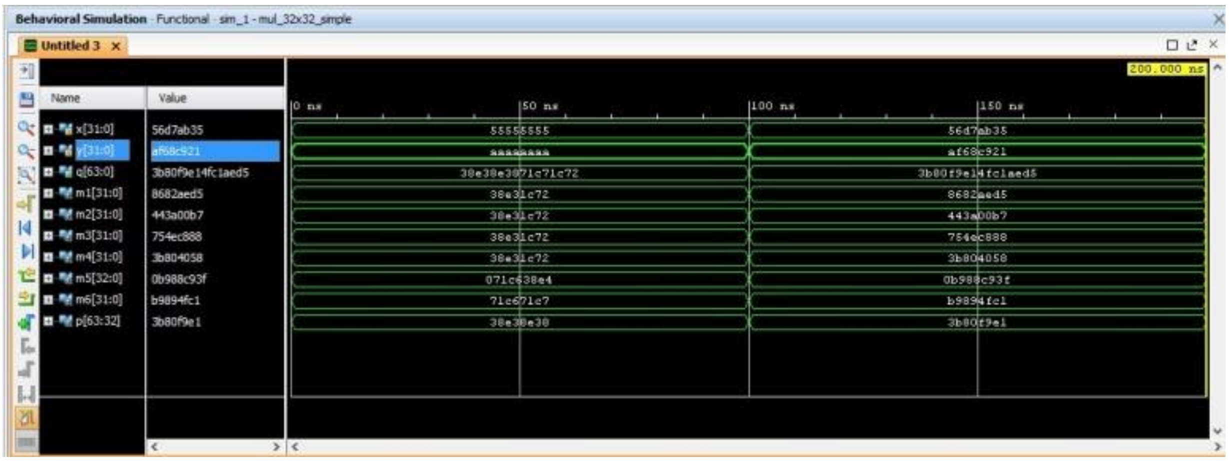
The simulation result for 32-bit multiplier is shown in the Fig.8 for 2 different inputs given below.

CASE - 1: Inputs x = 55555555 h, y = aaaaaaaa h

Multiplier’s output q = 38e38e3871c71c72 h

CASE - 2: Inputs x = 56d7ab35 h, y = af68c921 h

Multiplier’s output q = 3b80f9e14fc1aed5 h



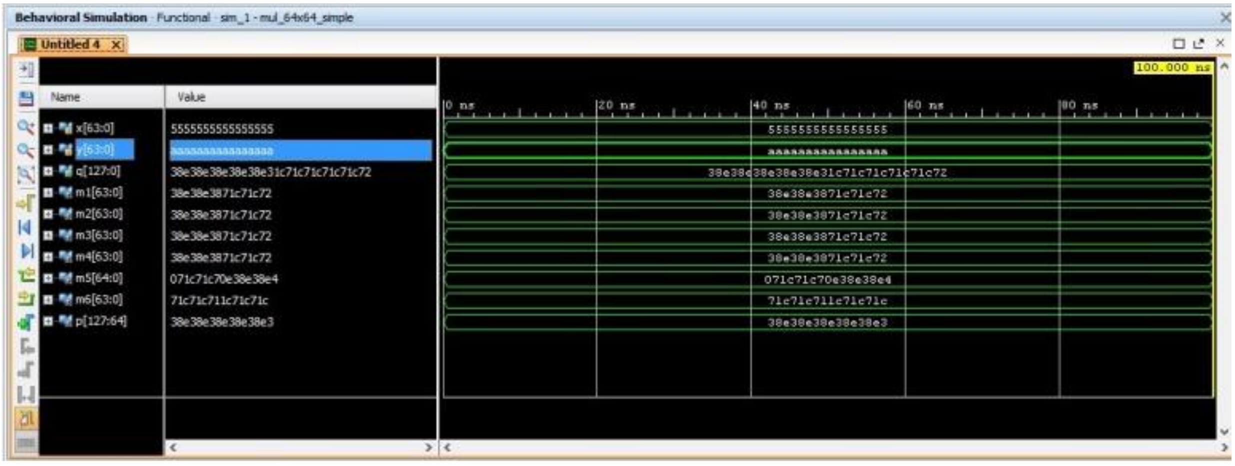
```
Behavioral Simulation - Functional - sim_1 - mul_32x32_simple
Untitled 3

Name        Value
x[31:0]     56d7ab35
y[31:0]     af68c921
q[63:0]     3b80f9e14fc1aed5
m1[31:0]    8682aed5
m2[31:0]    443a00b7
m3[31:0]    754ec888
m4[31:0]    3b804058
m5[32:0]    0b988c93f
m6[31:0]    b9894fc1
p[63:32]    3b80f9e1
```

Fig. 8. Waveform of 32-bit Multiplier

The simulation result for 64-bit multiplier is shown in the Fig.9 for inputs given below.

CASE - 1: Inputs x = 5555555555555555 h, y = aaaaaaaaaaaaaaaa h

Multiplier's output q = 38e38e38e38e38e31c71c71c71c71c72 h



```
Behavioral Simulation - Functional - sim_1 - mul_64x64_simple
Untitled 4

Name        Value
x[63:0]     5555555555555555
y[63:0]     aaaaaaaaaaaaaaaa
q[127:0]    38e38e38e38e38e31c71c71c71c71c72
m1[63:0]    38e38e3871c71c72
m2[63:0]    38e38e3871c71c72
m3[63:0]    38e38e3871c71c72
m4[63:0]    38e38e3871c71c72
m5[64:0]    071c71c70e38e38e4
m6[63:0]    71c71c711c71c71c
p[127:64]   38e38e38e38e38e3
```

Fig. 9. Waveform of 64-bit Multiplier

B. *Comparison*

This section compares three algorithms mentioned above where an 8x8 bit multiplier is designed using 4x4 bit in 1st and 2nd algorithm and directly using 8x8 bit multiplication mentioned in algorithm 3.

The higher bit multipliers are designed using technique mentioned in part III of the paper.

TABLE 1. Comparison of 3 Algorithm of 8x8 Multiplier

	Algorithm 1	*Algorithm 2*	*Algorithm 3*
No. of LUTs Used	93/63400	108/63400	81/63400
Bounded I/O	32/210	32/210	32/210
Total Delay (ns)	10.679	11.020	10.159
Total On-chip Power	104 mW	105 mW	104 mW
Logic Levels	9	10	9
LUT Utilization	0.15%	0.17%	0.13%

TABLE 2. Comparision of 3 Algorithm of 16x16 Multiplier

	Algorithm 1	*Algorithm 2*	*Algorithm 3*
No. of LUTs Used	397/63400	482/63400	353/63400
Bounded I/O	64/210	64/210	64/210
Total Delay (ns)	15.834	16.505	15.475
Total On-chip Power	127 mW	128 mW	127 mW
Logic Levels	17	17	17
LUT Utilization	0.63%	0.76%	0.56%

TABLE 3. Comparision of 3 Algorithm of 32x32 Multiplier

	Algorithm1	*Algorithm 2*	*Algorithm 3*
No. of LUTs Used	1631/63400	1928/63400	1471/63400
Bounded I/O	128/210	128/210	128/210
Total Delay (ns)	24.770	25.393	24.368
Total On-chip Power	190 mW	193 mW	188 mW
Logic Levels	31	31	31
LUT Utilization	2.57%	3.04%	2.32%

TABLE 4. Comparision of 3 Algorithm of 64x64 Multiplier

	Algorithm 1	*Algorithm 2*	*Algorithm 3*
No. of LUTs Used	6625/63400	7737/63400	5976/63400
Bounded I/O	256	256	256
Total Delay (ns)	41.915	42.556	41.747
Total On-chip Power	365 mW	378 mW	373 mW
Logic Levels	59	59	57
LUT Utilization	10.45%	12.20%	9.43%

The tables above compare 8 bit, 16 bit, 32 bit and 64 bit multipliers using three different algorithms where 3rd algorithm has the least LUT utilization and minimum amount of delay. However the I/O pins required for its implementation and on chip power dissipation remains almost same.

CONCLUSION

The three algorithms have been compared for different multipliers like 8x8 bit, 16x16 bit, 32x32 bit and 64x64 bit using Xilinx ISE design suite and implemented on Artix-7 Xilinx FPGA. It can be articulated from the results that algorithm 3 gives the least amount of worst case delay in each multiplier and also has minimum LUT utilization. However designing higher bit multipliers directly like 8x8 bit was designed in algorithm 3, becomes very complicated due to its size of implementation physically and it also becomes prone to human errors.

REFERENCES

[1] Premananda B.S., Samarth S. Pai, Shashank B. and ShashankS.Bhat, 2013.“Design and Implementation of 8-bit vedicMultiplier”.International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering.

[2] RohitNegi, 2016. “Design and Implementation of an Efficient Vedic Multiplier for High Performance and Low Power Applications”. International Journal of Engineering Technology, Management and Applied Sciences.

[3] Jagadguru Swami Sri Bharati Krishna TirthaJiMaharaj, 1986. “Vedic Mathematics”. MotilalBanarasidas, Varanasi, India.

[4] Ms.S.V.Mogre, Mr.D.G.Bhalke, 2015. “Implementation of high speed Matrix Multiplier Using Vedic Mathematics on FPGA” International Conference on computing Communication Control and Automation.

[5] KunalJadhav, AdityaVibhute, ShyamIyer, R.Dhanabal, 2015. “Novel Vedic mathematics based ALU using application specific Reversibility” IEEE Sponsored 9th International Conference on intelligent systems and control (ISCO).

[6] Anju and V.K. Agrawal, “FPGA Implementation of Low Power and High Speed Vedic Multiplier using Vedic Mathematics”, IOSR Journal of VLSI and Signal Processing (IOSR-JVSP) Volume 2, Issue 5 June 2013.

[7] G.Ganesh Kumar, V.Charishma, 2012. "Design of high speed Vedic Multiplier using Vedic Mathematics Techniques", International Journal of Scientific and Research Publications, Vol- 2, Issue -3, March 2012.