INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 5, Issue 3)
Available online at: www.ijariit.com

Parametric analysis of multipliers

Vaishnavi K. V.
vaishnavik04101@gmail.com
Sahyadri College of Engineering and Management, Mangaluru, Karnataka

Gurusiddayya Hiremath
gurusiddayya.ec@sahyadri.edu.in
Sahyadri College of Engineering and Management, Mangaluru, Karnataka

ABSTRACT

With the recent advancements, low power, low area and the quickest algorithms are in high demand. In this project, an effort is made to implement modular arithmetic operations like addition, multiplication is executed. The hardware description language used is Verilog. Each of these is implemented in Xilinx ISE 14.2 with Vertex 6 as the family and in Cadence 45nm technology. The area, power and the timings of each of these algorithms are tabulated. The layouts and the RTL schematic of these algorithms are also included. A designer whose objective is to design a system with arithmetic operations can make decision-based on these parameters. Therefore he has a stable and efficient system on his hands.

Keywords— *Booth multiplier, ISIM*

1. INTRODUCTION

The multiplier is a standout amongst the most omnipresent arithmetic data path operation in present day digital design. In the condition of workmanship Digital Signal Processing and graphics applications, multiplication is a significant and computationally escalated task. The multiplication operation is absolutely present in numerous pieces of a computerized framework or advanced PC, most prominently in sign handling, designs and logical calculation. The multiplier can be utilized in numerous applications such as in digital processing, cryptography and contributes in updating the exhibition of the application. However, these tasks command the execution time. Therefore, there is a need for high performance less delay multiplier. The interest of fast preparing has been expanding because of growing PC and sign handling applications.

Speed and effectiveness of calculation is frequently a significant factor in the successful utilization of computerized sensors. Digital Multipliers are at the centre of all Advanced Sign Processors (DSPs) and to a great extent decide the speed of the DSPs and advanced sensors. Because of the significance of digital multipliers in DSP, it has dependably been a functioning region of research and has a very extension.

In Array multiplier makes use of adding and shifting algorithm. In this algorithm, a partial product is generated by the multiplication of multiplicand with each bit of the multiplier. After every multiplication partial product thus generated is shifted according to its bit order and then all the partial products are added to obtain the final product. Array multiplier is known for its simple, scalable and repetitive structure but occupies a large area as well as has a significant combinational delay which many times contribute to the critical path delay [1].

A relative investigation of the resources associated with the Multiplication of two N-Bit double numbers is performed utilizing a Vedic Multiplication. Nikhilam Sutra is introduced for Multiplication of large paired numbers near a picked base and remarks are made regarding how such Multiplication is successfully decreased to Multiplication of little numbers.[2]

In paper [3], proved that the Vedic multiplier is better than array multiplier in terms of speed and resources required. The regular structure of Vedic multiplier makes its implementation using wider inputs easier and simple.

In the paper [4], booth multiplier by radix 2 and radix 4 are implemented and its speed, area are compared. They concluded that radix-2 booth multiplier has less area and circuit complexity compared to radix 4 but radix 4 has higher speed and performance.

A 16-bit multiplier has been designed using a radix-8 and radix-16 Booth's multiplication that reduces the number of partial products. Radix-8 and radix-16 booth's multiplication algorithms were carried out for the DSP applications. Concluded that radix-16 takes less delay but more area than that of radix-8 booth's multiplication [5].

2. IMPLEMENTATION

Multipliers are an important unit of microprocessor and DSP applications. Area, power and latency are the factors that need to be monitored while designing a multiplier. The way that the exhibition of any framework is dictated by the presentation of the slowest component in framework, multiplier being the slowest component, voluminous research has been accomplished for low power multipliers.

2.1 Vedic Multiplier

Vedic mathematics has a total of 16 sutras and 13 sub sutras, each of the 16 sutras have been used for various mathematical operations like algebra, multiplication, addition, division, trigonometry, calculus, etc. There are 2 Vedic sutras used for multiplication, Urdhva–Tiryagbhyam and Nikhilam Navatascaramam Dasatah sutra. The latter can be applied only if the multiplicand and multiplier are closer to the bases of 10 (10, 100,1000,10000 and so on), hence usually Urdhva–Tiryagbhyam sutra is used. It is additionally alluded to as vertically and crosswise algorithm.

In this project, the Vedic methodology is implemented as shown in figure 1, which describes the Vedic multiplier methodology. 2×2 Vedic multiplier is the leaf module for an N×N multiplication. For a 4×4 multiplication, the multiplicand and multiplier are divided into two groups of 2bits each. 2×2 multiplication is performed on these groups and the partial products are obtained. These partial products are then modified to get an 8-bit product term.

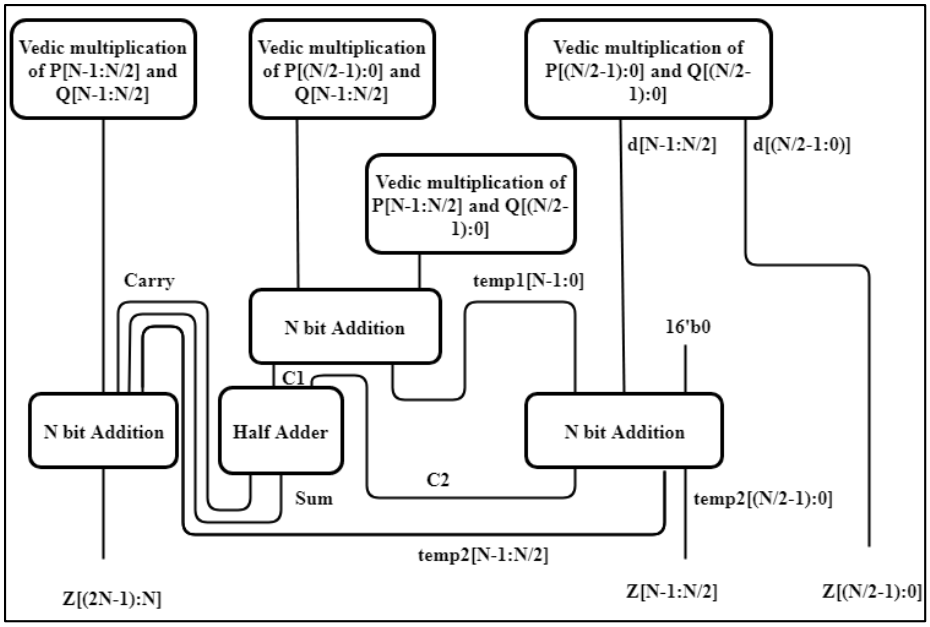

Fig. 1: Structure of Vedic Multiplier

2.2 Booth's multiplication algorithm

Signed multiplication is a cautious procedure. With unsigned multiplication compelling reason to keep the sign of the number into deliberation. Despite in signed multiplication, the alike procedure cannot be enforced because of the signed number in 2's compliment structures which would yield and erroneous result if multiplied in an analogues manner to unsigned multiplication. Booth's algorithm preserves the sign of the outcome. There are three noteworthy strides to any multiplication. In the initial step, the incomplete items are created. In the second step, the partisan products are downsized to one row of final sums and carries. In the third step, the final sums and carries are added to produce the outcome.

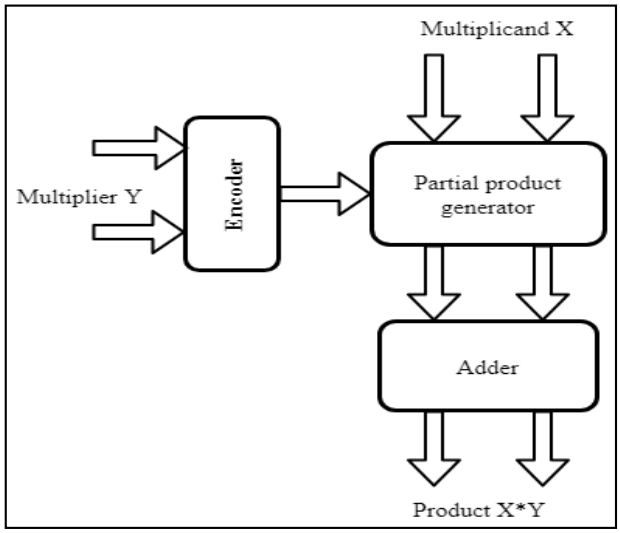

Fig. 2: Structure of Booth's algorithm

A modified booth multiplier should focus on the accompanying things: 1) on decreasing the total number of partial items created. This may incorporate any coding strategies or decrease of calculation unpredictability of the generation of partial products. 2) A substantial amount of delay is devoured in discovering two's complement of the multiplicand. So this delay ought to be diminished. 3) The optimization of the adder structure. Once partial products generated, they have to be gathered and included in an orderly way devouring less deferral. This may consider the utilization of parallelism of the procedure. The structure of Booth's method is illustrated in figure 2.

Booth's algorithm depends on recoding the multiplier, A, to a recoded esteem C_k, leaving the multiplicand, B unaltered. In Booth recoding, every digit of the multiplier can accept negative as well as positive and zero quantities.

Let us consider Multiplicand B 'm' bits wide depicted as $B_{m-1}B_{m-2}\ldots B_2B_1B_0$ and multiplier A again 'm' bits wide portrayed as $A_{m-1}A_{m-2}\ldots A_2A_1A_0$. Both of these are signed (two's compliment) binary numbers. As per Booth's algorithm,

$$B\times A=B\times\{(S_{n-1}\times 2^{m-1})+(S_{m-2}\times 2^{m-2})\ldots\ldots(S_2\times 2^2))+(S_1\times 2^1))+(S_0\times 2^0) \quad (1)$$

Where each S_k for m-1<=k<=0, is a value which depends upon the value of A, and can be found as explained in the following steps:

Step 1: Affix A by a '0' on LSB, we will call this bit as Z

Step 2: Now a prompt set of q bits, where q = 2, for radix 2 booth algorithm, and label each collection C_k , where m-1<=k<=0, The decree to compose each set C_k is such that

$$C_k = (A_k\ A_{k-1}),\ \text{if m-1<=k<=0 and } C_k=(A_kZ)\ \text{for k=0.}$$

This process will result in 'm' collections such that:

$$C_{m-1}=(A_{m-1}\ A_{m-1)\ldots}\ C_1=(A_1\ A_0),\ C_0=(A_0\ Z)$$

Step 3: Now based on the value of C_k, where m-1<=k<=0, find out S_k, where the value of S_k is outlined in the subsequent table for all possible combinations of values of a pair C_k

Table 1: Radix-2 Operations to be performed on the multiplicand

C_k	S_k
00	0
01	+1*B
10	-1*B
11	0

Now equation (1) can re-written as,

$$B\times A=(B\times p_{m-1})+(B\times p_{m-2})\ldots\ldots+(B\times p_1)+(M\times p_0)$$

Where,

$$p_{m-1}=S_{m-1}\times 2^{m-1},\ p_{m-2}=S_{m-2}\times 2^{m-2}\ldots p_1=S_1\times 2^1,\ p_0=S_0\times 2^0$$

$$B\times A=pp_{m-1}\times 2^{m-1}+pp_{m-2}\times 2^{m-2}\ldots..+pp_1\times 2^1+pp_0\times 2^0 \quad (2)$$

Where $pp_{n-1}=(B\times p_{m-1})$, $ppm_{-1}=(B\times p_{m-2})\ldots pp_1=(B\times p_1)$, $pp_0=(B\times p_0)$ are referred to as partial products.

Step 4: Add these 'm' partial products as shown in the equation below to get the final product.

$$B\times A=pp_{m-1}\times 2^{m-1}+pp_{m-2}\times 2^{m-2}\ldots..+pp_1\times 2^1+pp_0\times 2^0 \quad (3)$$

Note that equation (3) is the same as equation (2), reproduced for lucency.

It is important to note that, we will have to sign extend the partial products, to give rise to equal in width before adding.

The Modified Booth Multiplier or Radix-4 Booth's multiplier was recommended by O. L. Macsorley in 1961. The recoding strategy is generally adapted to generate the partial products for execution of huge parallel multipliers, which receives the parallel encoding plan.

In radix-2 booth's algorithm, in the event that multiplying 2 m bits values, we possess' partial products to add. Utilizing Radix-4 booth's multiplier, the number of partial products is diminished to 'm/2' on the off chance that we are multiplying two' bits numbers, if 'm' is even number, or '(m+1)/2', if 'm' is an odd number. By decreasing the number of partial products, one can adequately accelerate the multiplier by a factor roughly generally equivalent to 2. On the off chance that takes 'q'=3, and makes accumulations of 'q' bits taken in one C_k. Where N-1<=k<=0 where N = m/2, for 'n' is even, N= (m+1)/2 if 'm' is odd. From the multiplier R, such that:

$$C_k = (A_{2k+1}\ A_{2k}\ A2k_{-1}),\ \text{for N-2<=k<=1}$$

$$C_k = (A_{2k+1}\ A_{2k}\ Z),\ \text{for k=0 and Z=0}$$

$$C_k = (A_{2k+1}\ A_{2k}\ A2k_{-1}),\ \text{if 'm' is even and k=N-1}$$

We may need to sign-extend 'B' by one bit if the C_{N-1} contains less than 'q' or 3 bits. This will dependably occur when 'm' is odd. The quantity of partial products will likewise diminish from 'm' to 'N'. All the fractional items in condition (3), which are multiplied by 2×, where '×' is odd will vanish. That implies as opposed to moving an incomplete item (before summing them up for the last item) by '1' bit, for example, increasing it by 2^1 in every fractional item, we will move by 2 bits rather, for example multiplying it by 2^2. Table 1 will presently change to table 2 given beneath:

Table 2: Radix-4 Operations to be performed on multiplicand

C_k	S_k
000	0
001	+1*B
010	+1*B
011	+2*B
100	-2*B
101	-1*B
110	-1*B
111	0

Final product equation will now become:

$$B\times A= pp_{m-1}\times 2^{2*(m-1)} + pp_{m-2}\times 2^{2*(m-2)} \ldots\ldots + pp_1\times 2^1 + pp_0\times 2^0 \quad (4)$$

Where,

$$pp_{N-1} = B\times S_{N-1},\ pp_{N-2} = B\times S_{N-2} \ldots., pp_1=B*S_1, pp_0 = B*S_0$$

pp_k are called partial products.

The radix-4 booth multiplier equation can be written as,

$$B\times A= B\times S_{N-1}\times 2^{2*(N-1)} + B\times S_{N-2}\times 2^{2*(N-1)} + \ldots. + B\times S_1\times 2^1 + B\times S_0\times 2^0 \quad (5)$$

Note that all the terms which enclose multiplication by 2 to the power 'x', where 'x' is an odd number have vanished, referring that while the addition of the partial products, each partial product will be moved by 2 bits rather than 1 bit.

Radix 8 booth encoding multiplier utilizes 4-bit encoding plan. Because of that number of incomplete item diminish by 33% factor. Be that as it may, the circuit intricacy additionally increments as a contrast with a past rendition of booth multiplier. Execution of the radix multiplier can be improved by presenting parallelism which brings about lessening the number of estimation stages.

For a gathering of 4-bits, the Signed Multiplier Digit is determined in the table. 3 which characterize Radix-8 Booth are recoding technique for every conceivable mix in the binary input where B is the Multiplier.

Table 3: Radix-8 Booth's recoding technique

C_k	S_k
0000, 1111	0
0001, 0010	+1*B
0011, 0100	+2*B
0101, 0110	+3*B
0111	+4*B
1000	-4*B
1001, 1010	-3*B
1011	-2*B
1101, 1110	-1*B

There will be four partial products.

To diminish the number of partial products included while multiplying the multiplicand higher radix Booth's method is a standout amongst the most notable systems used. Radix-16 Booth method which examines a series of five bits with the algorithm shown here:

(a) Elongate the sign bit 1 position if necessary to assure that m is even.

(b) Affix 0 to the right of the LSB of the multiplier.

Signed multiplier digit for the group is defined in table 4 as per the Booth's recoding technique for every binary combination.

Table 4: Radix-16 Booth's recoding strategy

C_k	S_k
00000, 11111	0
00001, 00010	+1*B
00011, 00100	+2*B
00101, 00110	+3*B
00111, 01000	+4*B

01001, 01010	+5*B
01011, 01100	+6*B
01101, 01110	+7*B
01111	+8*B
10000	-8*B
10001, 10010	-7*B
10011, 10100	-6*B
10101, 10110	-5*B
10111, 11000	-4*B
11001, 11010	-3*B
11011, 11100	-2*B
11101, 11110	-1*B

3. SIMULATION RESULTS

3.1 Vedic Multiplier

Figure 3 shows the simulation result of the Vedic multiplier.

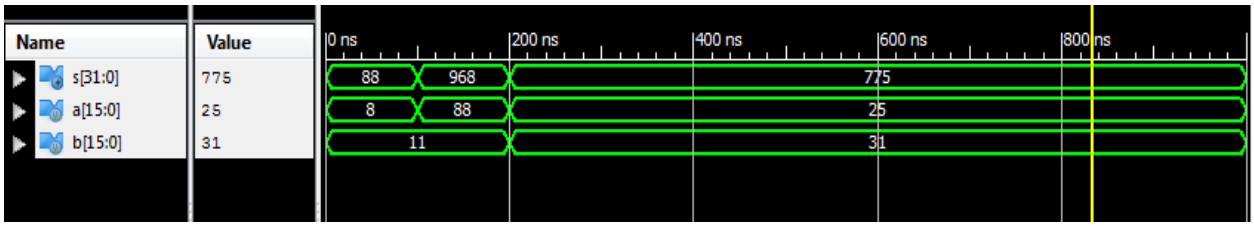

Fig. 3: Waveform of Vedic multiplier

The RTL schematic of the Vedic multiplier is as shown below:

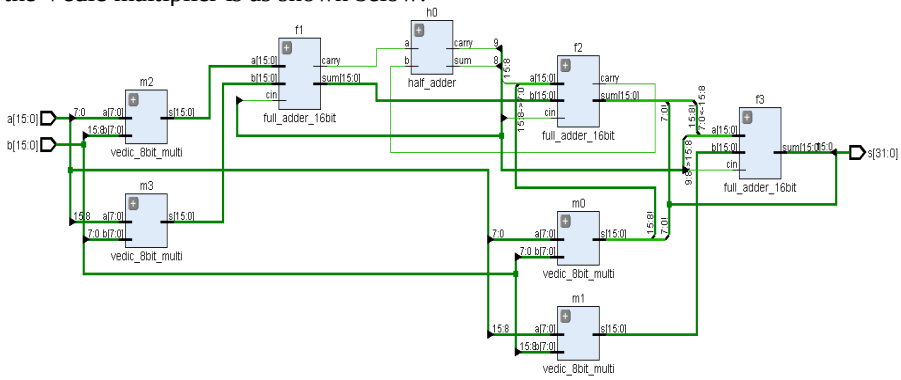

Fig. 4: RTL schematic

3.2 Booth's Multiplier

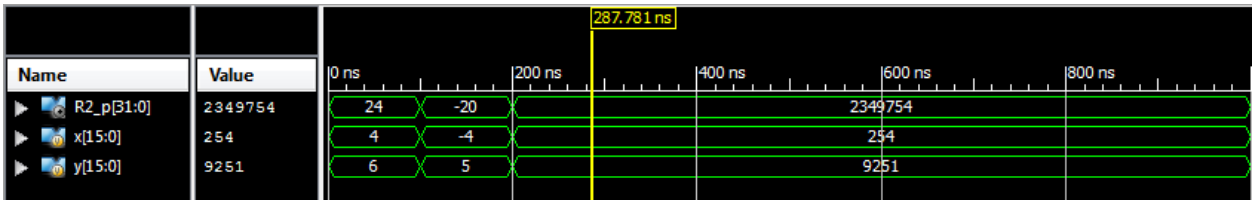

Fig. 5: Radix-2 booth multiplier waveform

The RTL schematic is as shown below:

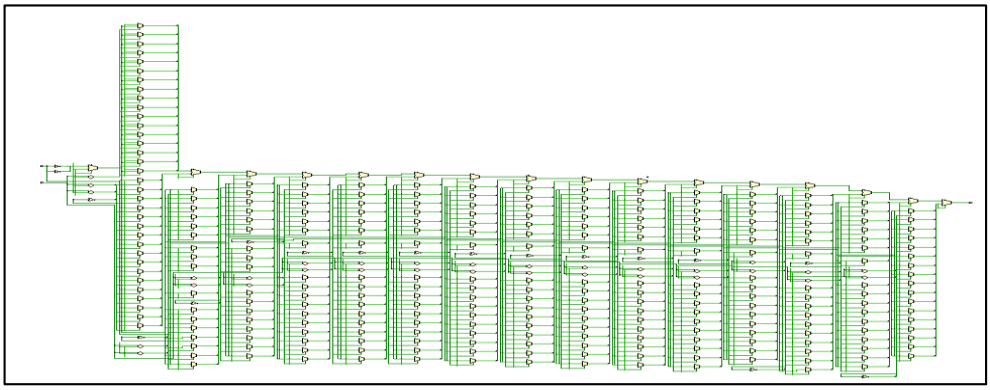

Fig. 6: Radix-2 booth multiplier RTL Schematic

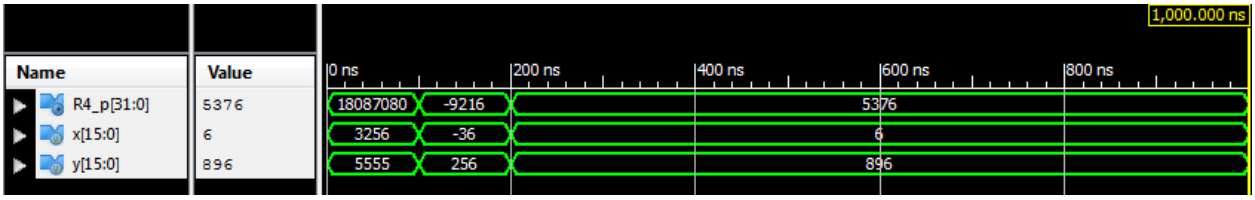

Fig. 7: Waveform of Radix-4 Booth's multiplier

The figure below shows the RTL schematic of the Radix-4 multiplier.

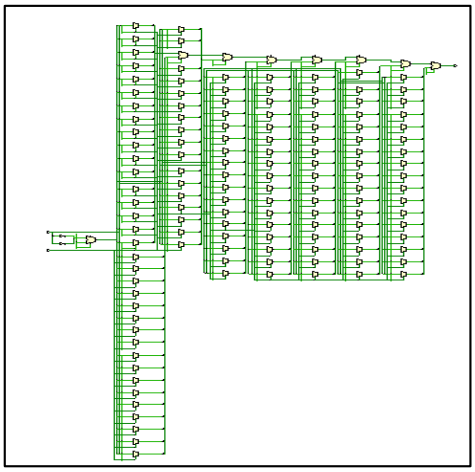

Fig. 8: RTL schematic of Radix-4 Booth's multiplier

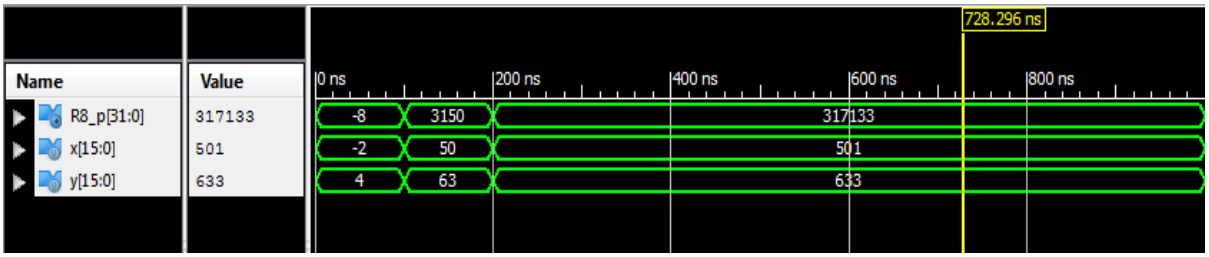

Fig. 9: Waveform of Radix-4 Booth's multiplier

The RTL schematic is as shown in the below figure:

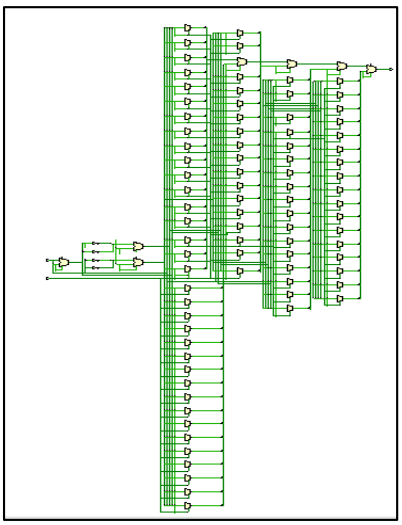

Fig. 10: RTL schematic of Radix-8 Booth's multiplier

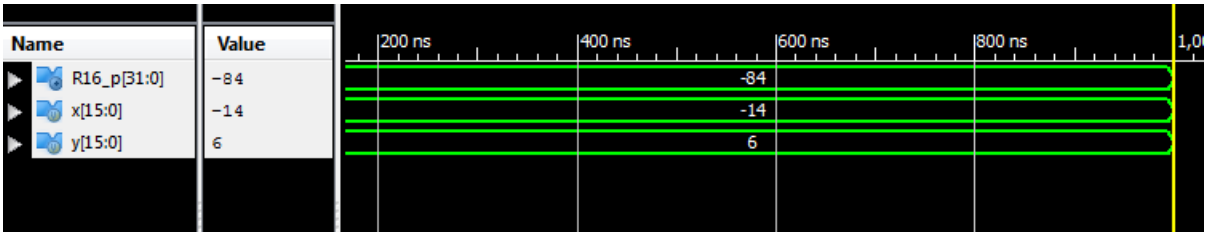

Fig. 11: Waveform of Radix-16 Booth's multiplier

The RTL schematic is as shown in the below figure.

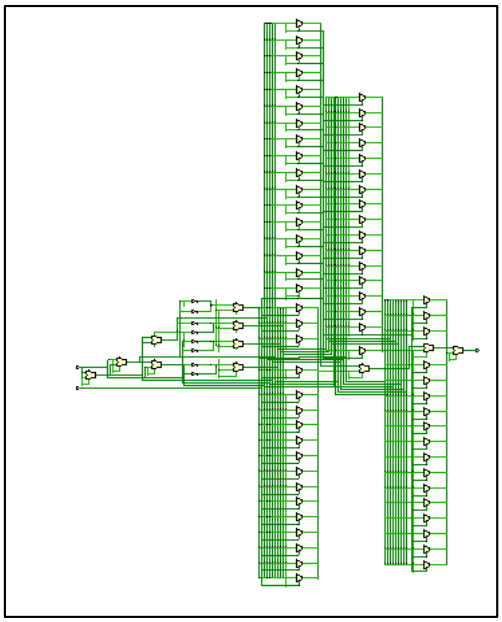

Fig. 12: RTL schematic of Radix-16 Booth's multiplier

The Device utilization summary of Vedic, Radix-2, 4, 8 and 16 Booth's multipliers are shown below:

Table 5: Summary of Vedic, Radix-2, 4, 8 and 16 Booth's multipliers

Spartan 6				
Design	**Timings(ns)**	**Power (mW)**	**Area**	
			Slices LUT's	**IOB's**
Vedic Multiplier	30.45	20	558	64
Radix-2 Booth's Multiplier	14.19	20	379	64
Radix-4 Booth's Multiplier	13.03	20	331	64
Radix-8 Booth's Multiplier	23.93	32	337	63
Radix-16 Booth's Multiplier	16.40	20	584	64

4. CONCLUSION

In this paper, 16 bit Vedic, Radix-2, 4, 8 and 16 are carried out. It can be seen that the delay consumed for Vedic multiplier is highest and the lowest for the Radix-4 Booth's multiplier. The area utilized for Radix-16 Booth's multiplier is largest.

The selection of algorithm is done by the designer based on values of constraints i.e. area, power and timing. An algorithm that requires a larger area but is the fastest cannot be used in an application in which area is critical. The utilization of an algorithm depends on the application.

5. REFERENCES

[1] Patel, Pranav, Ashish Shandilya, Nisarg Brahmbhatt, Kshitij Raval, and Dipankar Deb. "Vedic and conventional methods of N× N Binary Multiplication with hardware implementation." In 2015 International Conference on Smart Sensors and Systems (IC-SSS), pp. 1-6. IEEE, 2015.

[2] Meghana, V., S. Sandhya, R. Aparna, and C. Gururaj. "High speed multiplier implementation based on Vedic Mathematics." In 2015 International Conference on Smart Sensors and Systems (IC-SSS), pp. 1-5. IEEE, 2015.

[3] Kaur, Sukhmeet, and Manpreet Singh Manna. "Implementation of modified booth algorithm (radix 4) and its comparison with booth algorithm (radix-2)." Advance in Electronic and Electric Engineering 3, no. 6 (2013): 683-690.

[4] Design and Comparison of High Speed Radix-8 and Radix-16 Booth's Multiplier, International Journal of Computer Applications (0975 – 8887) Volume 181 – No.2, July 2018

[5] Madrid, Philip E., Brian Millar, and E. E. Swartzlander. "Modified booth algorithm for high rdix multiplication." In Proceedings 1992 IEEE International Conference on Computer Design: VLSI in Computers & Processors, pp. 118-121. IEEE, 2002

