International Journal of
Science and Engineering Investigations vol. 2, issue 12, January 2013
ISSN: 2251-8843

High Speed Modified Booth's Multiplier for Signed and Unsigned Numbers

Manish Chaudhary[1], Mandeep Singh Narula[2]
[1]M.Tech (ECE), ITM University, Gurgaon, Haryana
[2]Assistant Professor (Dept. of EECE), ITM University, Gurgaon, Haryana
([1]chaudhary.manish89@gmail.com, [2]msnarula@itmindia.edu)

Abstract- In this paper, we have designed a signed booth's multiplier as well as an unsigned booth's multiplier for 4 bit, 8 bit and 16 bits performing multiplication on signed and unsigned number. The implementation is done through Verilog on xiling12.4 platform which provide diversity in calculating the various parameters. The unsigned booth multiplication is implemented by doing some modification in the booth's multiplication algorithm. In this paper we have tried to explain the modification done in booth's algorithm for signed and unsigned numbers by dividing it in to five steps. The array structures of signed and unsigned multipliers obtained from RTL Synthesis are shown. Different parameters like Power, CPU Usage, simulation etc have been compared for both signed and unsigned multipliers.

Keywords-*verilog; booth; signed multiplier; unsigned multiplier*

I. INTRODUCTION

Multiplication is an essential arithmetic operation and its applications are dated several decades back in time. Earlier ALU's adders were used to perform the multiplication originally. As the applications of Array multipliers were introduced the clock rates increased as well as timing constrains became austere. Ever since then methods to implement multiplication are proposed which are more sophisticated [1-4]. As known the use of multiplication operation in digital computing and digital electronics is very intense especially in the field of multimedia and digital signal processing (DSP) applications [6]. There are mainly three stages to perform multiplication: The first stage mainly consists of generating the partial products which are generated through an array of AND gates; Second stage consist of reducing the partial products by the use of partial product reduction schemes; and finally the product is obtained by adding the partial products [5].

The multiplication can be performed on: 1) Signed Numbers; 2) Unsigned Numbers. Signed multiplication a binary number of either sign (two numbers whose sign may are not necessarily positive) may be multiplied. But, in signed multiplication the sign-extension for negative multiplicands is not usable for negative multipliers and there are large numbers of summands due to the large sequence of 1's in multiplier. Unsigned multiplication binary number (whose sign is positive) is multiplied.

In the figure 1 we have seen the partial product array of signed and unsigned binary multiplication and the difference in their procedure of being multiplied and the calculation of the final product. Figure1 (a) in this we have firstly produced the eight partial products through on-bit multiplication, one for each bit in multiplicand. Once, the partial products are obtained then they are all added to get the final product.

Figure1 (b) Here, if one of the variable is signed integer, then there would be sign-extension in the partial products before summing. In this the array is modified by inverting several of the products which support two's compliment.(~p denotes compliment). The one's compliment sequence that is been followed by the non-complimented bits are substituted to avoid sign-extension in the final step. Here in part(b) of figure1 the last line the non complimented bits are being followed by the complimented bits this is due to the subtraction of this term as they all will start to negate out.

The Booth's algorithm is powerful for signed number multiplication (larger multiplier, lager number of multiplicands to be added). It performs multiplication by performing 2'compliment and the regular shift and adds process. As due to the extra partial bit at the least significant bit position there is irregularity produced in the array. It is seen that in booth technique if the operands are large (bit numbers) and there is a long sequence of 1's it is advantageous.

The booth's algorithm for multiplication can be modified to perform unsigned multiplication along with signed multiplication. We have done some basic changes in the algorithm to obtain the result of signed-unsigned multipliers. After implementing the signed booth multiplier and unsigned booth multiplier for 4, 8 and 16 bits binary numbers we have compared their parameters (power usage, current leakage, CPU usage, and memory usage) results with each other.

p0[7] p0[6] p0[5] p0[4] p0[3] p0[2] p0[1] p0[0]
+ p1[7] p1[6] p1[5] p1[4] p1[3] p1[2] p1[1] p1[0] 0
+ p2[7] p2[6] p2[5] p2[4] p2[3] p2[2] p2[1] p2[0] 0 0
+ p3[7] p3[6] p3 [5] p3[4] p3[3] p3[2] p3[1] p3[0] 0 0 0
+ p4[7] p4[6] p4[5] p4[4] p4[3] p4[2] p4[1] p4[0] 0 0 0 0
+ p5[7] p5[6] p5[5] p5[4] p5[3] p5[2] p5[1] p5[0] 0 0 0 0 0
+ p6[7] p6[6] p6[5] p6[4] p6[3] p6[2] p6[1] p6[0] 0 0 0 0 0 0
+ p7[7] p7[6] p7[5] p7[4] p7[3] p7[2] p7[1] p7[0] 0 0 0 0 0 0 0
p[15] p[14] p[13] p[12] p[11] p[10] p[9] p[8] p[7] p[6] p[5] p[4] p[3] p[2] p[1] p[0]

(a)

1 ~`p0[7] p0[6] p0[5] p0[4] p0[3] p0[2] p0[1] p0[0]
~ p1[7]+p1[6]+p1[5]+p1[4]+p1[3]+p1[2]+p1[1]+p1[0] 0
~p2[7]+p2[6]+p2[5]+p2[4]+p2[3]+p2[2]+p2[1]+p2[0] 0 0
~ p3[7]+p3[6]+p3[5]+p3[4]+p3[3]+p3[2]+p3[1]+p3[0] 0 0 0
~ p4[7]+p4[6]+p4[5]+p4[4]+p4[3]+p4[2]+p4[1]+p4[0] 0 0 0 0
~p5[7]+p5[6]+p5[5]+p5[4]+p5[3]+p5[2]+p5[1]+p5[0] 0 0 0 0 0
~p6[7]+p6[6]+p6[5]+p6[4]+p6[3]+p6[2]+p6[1]+p6[0] 0 0 0 0 0 0
1+p7[7]~p7[6]~p7[5]~p7[4]~p7[3]~p7[2]~p7[1]~p7[0] 0 0 0 0 0 0 0
p[15] p[14] p[13] p[12] p[11] p[10] p[9] p[8] p[7] p[6] p[5] p[4] p[3] p[2] p[1] p[0]

(b)

Figure1: Partial Product array for (a) unsigned (b) signed multiplication

II. BASIC TECHNOLOGY

A. *Booth's Multiplication Algorithm*

The algorithm is based on checking the bits of multiplier Y in two's compliment, which also include the implicit bit below the LSB,$Y_{-1}=0$. Here we consider the an addition of multiplicand 2^i to the product accumulator, considering $y_i = 0$ and $y_{i-1} = 0$ and 2^i multiplicand is subtracted from the product accumulator. Hence, the final product is achieved.

B. *Steps of Booth's Algorithm implementation(Signed)*

Booth's algorithm is executed by repeatedly adding one of the two multiplicands and multipliers and then performing the rightward arithmetic shift.

Consider e and f a multiplicand and multiplier respectively these two values which are going to be multiplied and give product. Let, the bits of e and f be represented by x and y.

1.) Firstly we arbitrate the values of two predetermined A and S to obtain the product P. length of all these numbers should be equal(x+y+1)

   a.) A: Substitute the value of **e**(binary) in the MSB(most significant bit) and append the remaining bits with y+1 zeros.

   b.) S: Substitute the value of **-e**(two's compliment notation) in the MSB (most significant bit) and append the remaining bits with y+1 zeros.

   c.) P: Substitute the MSB with x bits of zeros. Then to the right of this insert the value of **f** and append the LSB(least significant bit) bits with zero.

2.) Now, consider the two least significant bits of **P.**

   a.) If the two least significant bits are 01, then the values of P becomes P+A. ignoring overflow

   b.) If the two least significant bits are 10, then the values of P becomes P+S. ignoring overflow

   c.) If the two least significant bits are 00, then the values of P is used as it is.

   d.) If the two least significant bits are 11, then the values of P is used as it is.

3.) After the completion of the 2nd step we will do the Arithmetic shift by single place towards the right. Let the new value be equal to P now.

4.) For y number of times repeat the 2nd and 3rd step.

5.) To obtain the final product result of e and f we have to drop the least significant bit from P.

Example1:Signed Booth
Considering e=0010(2) , -e=1101 and f=0011(3).
x=4;y=4
A:0010 0000 0

S=1101 0000 0

P=0000 0011 0
Performing the 2nd step and 3rd step for y times where y=4.

1. P= 0000 0011 0. The last two bits are 10
   P=P+S, 0000 0011 0 + 1101 0000 0=1101 0011 0
   Arithmetic shift .P=1110 1001 1.

2. P=1110 1001 1. The last two bits are 11.
   Arithmetic shift. P=1111 0100 1

3. P=1111 0100 1. The last two bits are 01
   P=P+A, 1111 0100 1+0010 0000 0=0001 0100 1
   Arithmetic shift .P=0000 1010 0

4. P=0000 1010 0. The last two bits are 00
   Arithmetic shift.P=0000 0101 0

   To obtain the final result drop the least significant bit from P, hence we get
   P= 0000 0101=6

C. *Steps of Booth's Algorithm i*mplementation(Unsigned)*

The booth's algorithm for unsigned multiplication is almost same but the only difference is the along with repeatedly adding the two multiplicands and multipliers they are also repeatedly subtractedand then the airthmetic shift is performed. Secondly two's compliment is not done.

Consider e and f a multiplicand and multiplier respectively these two values which are going to be multiplied and give product. Let, the bits of e and f be represented by x and y.

1.) Firstly we arbitrate the values of two predetermined A and S to obtain the product P. length of all these numbers should be equal(x+y+1)

   a.) A: Substitute the value of e(binary) in the MSB(most significant bit) and append the remaining bits with y+1 zeros.

   b.) S: Substitute the value of e in the MSB (most significant bit) and append the remaining bits with y+1 zeros.

   c.) P: Substitute the MSB with x bits of zeros. Then to the right of this insert the value of f and append the LSB (least significant bit) bits with zero.

2.) Now, consider the two least significant bits of P.

   a.) If the two least significant bits are 01, then the values of P becomes P+A. ignoring overflow

   b.) If the two least significant bits are 10, then the values of P becomes P-S. ignoring overflow

   c.) If the two least significant bits are 00, then the values of P is used as it is.

   d.) If the two least significant bits are 11, then the values of P is used as it is.

3.) After the completion of the 2^{nd} step we will do the Arithmetic shift by single place towards the right. Let the new value be equal to P now.

4.) For y number of times repeat the 2^{nd} and 3^{rd} step.

5.) To obtain the final product result of e and f we have to drop the least significant bit from P.

Example2 Unsigned
Considering e=0010(2) , -e=1101 and f=0011(3).
x=4;y=4
A:0010 0000 0
P=0000 0011 0

Performing the 2^{nd} step and 3^{rd} step for y times where y=4.

5. P= 0000 0011 0. The last two bits are 10
   P=P-A, 0000 0011 0 + 0010 0000 0=111 010
   Arithmetic shift .P=111 101.

6. P=11110 1. The last two bits are 01.
   P=P+A=111 101 + 0010 0000 0= 000101000
   Arithmetic shift.P=0000 1010 0
7. P=0000 1010 0. The last two bits are 00
   Arithmetic shift .P=0000 0101 0

   To obtain the final result drop the least significant bit from P, hence we get
   P= 0000 0101=6

III. RESULTS

In this section we have written and simulated the Verilog code on xilings of the booth's multipliers (signed and unsigned) for 4-bit and 8-bit and have verified it through a test bench, which has generated a waveform representing the output function.

The name of simulator is ISim. In this simulator we observe test benches. In test bench we get waveforms representation of the given parameters and response of the system according to given parameters.

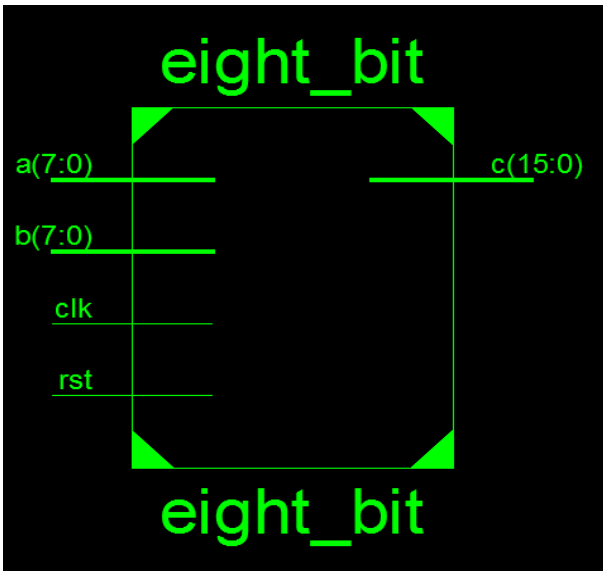

Figure 2: RTL Booth's 8 Bit signed

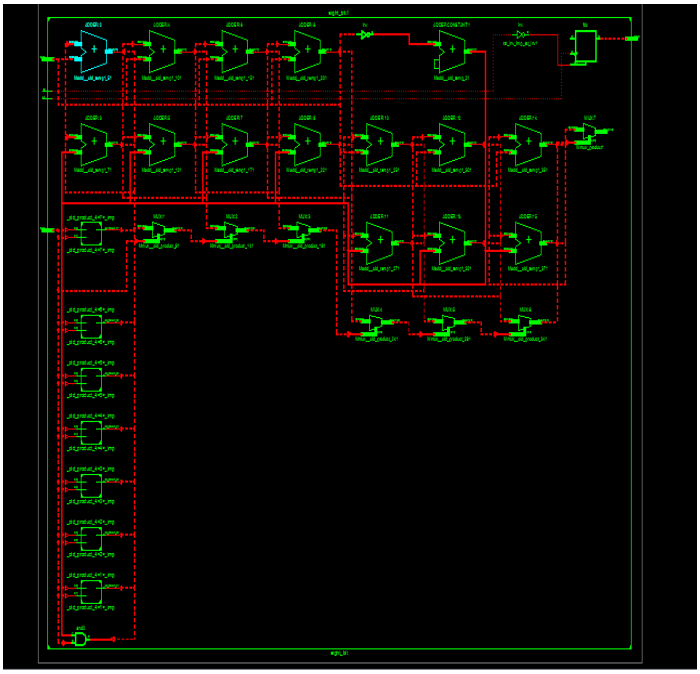

Figure 3: Array Structure 8-Bit signed

Figure3 shows the array structure and arrangement of the 8-bit signed booth multiplier.

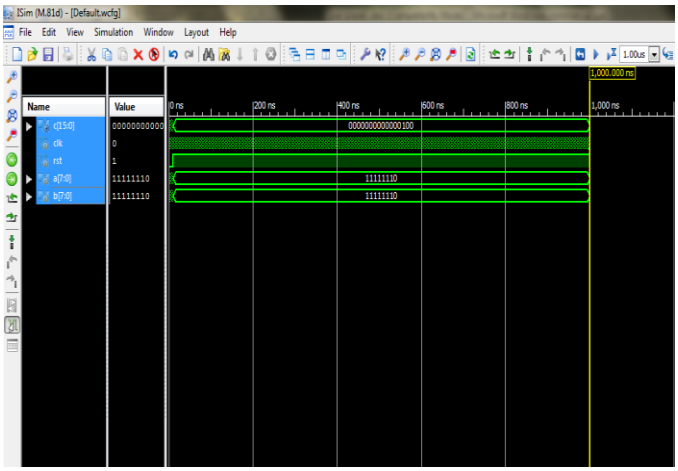

Figure 4: Waveform Of The Product (8-bit signed)

Figure4 shows the output waveform. Here, the input given is 8 bit value a = 8'b11111110; b = 8'b11111110. After the executing the test bench according to the verilog code the output waveform is achieved which is c=0000000000000100

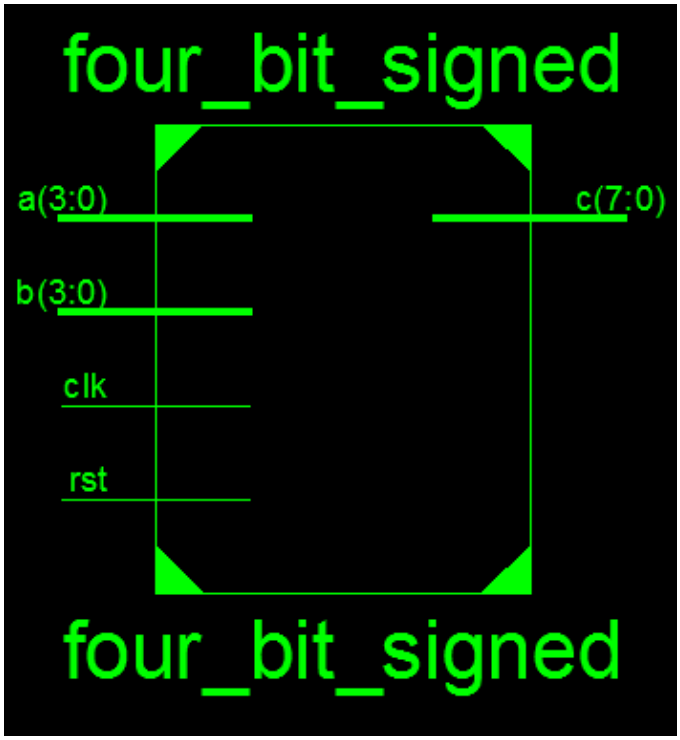

Figure 5: RTL Booth's 4-Bit Signed

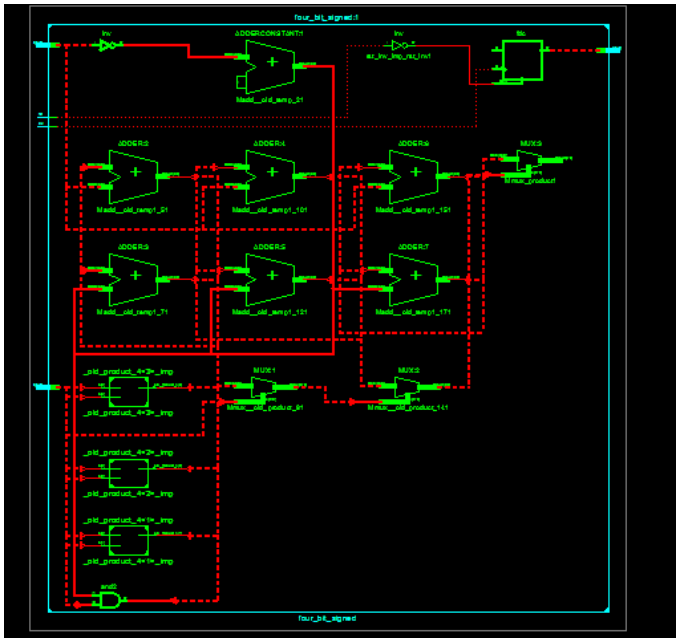

Figure 6:Array Structure Of 4-bit Signed

Figure6 is the representation of the array structure consisting of adders and multipliexers.

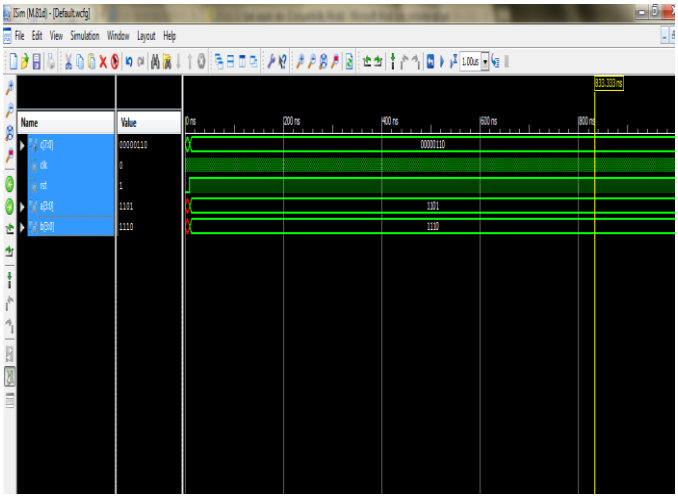

Figure 7: Waveform of The Product (4-bit signed)

Figure7 shows the output waveform. Here, the input given is 8 bit value a = 4'b1101; b = 4'b1110;. After the executing the test bench according to the Verilog code the output waveform is achieved which is c=00000110.

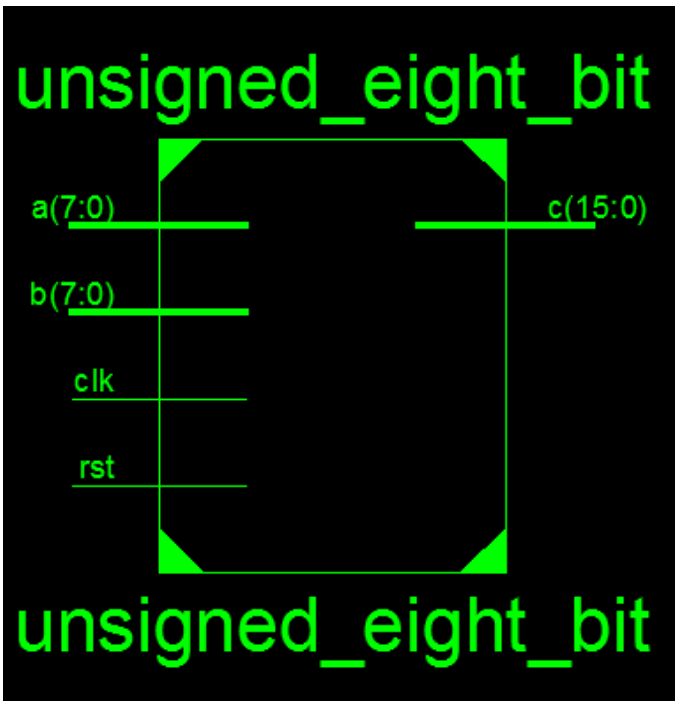

Figure 8: RTL Booth's 8-bit Unsigned

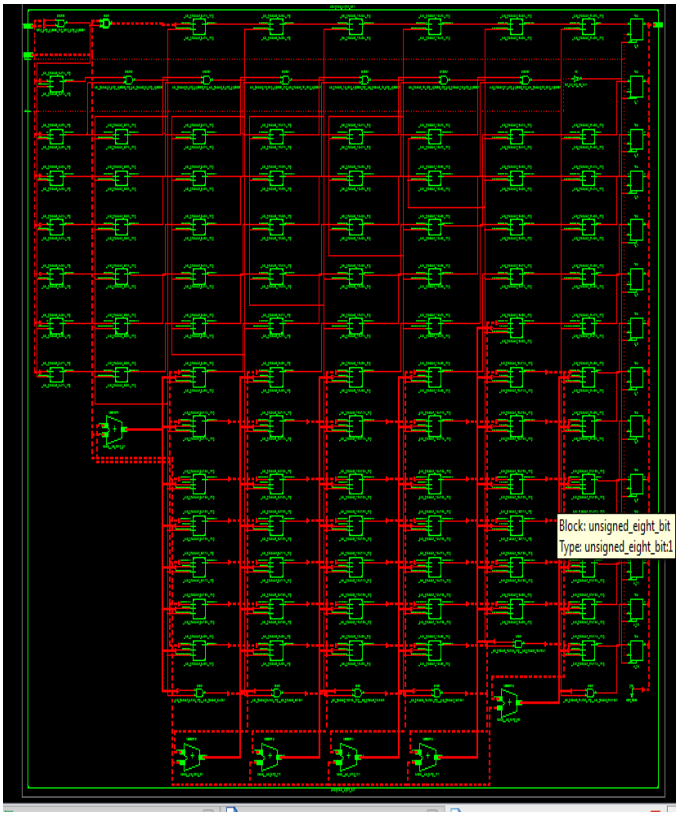

Figure 9: Array Structure of 8-Bit Unsigned

Figure9 shows the array structure and arrangement of the 8-bit unsigned booth multiplier.

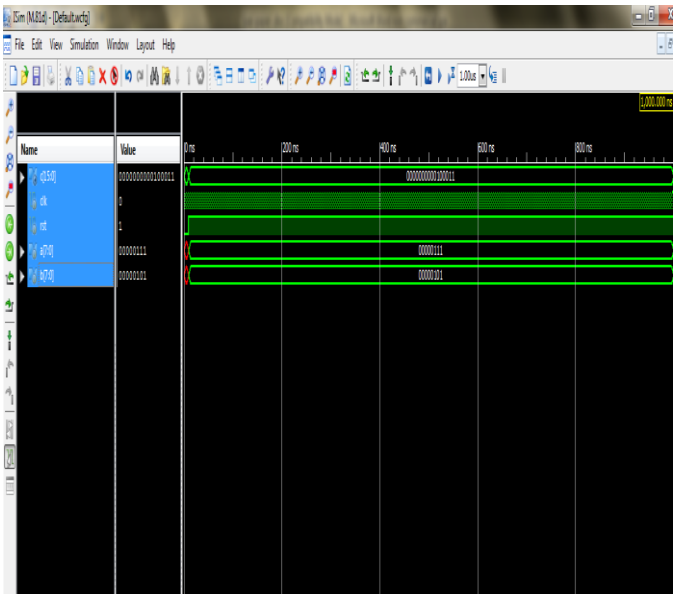

Figure 10:Waveform Of The Product(8-bit unsigned)

Figure10 shows the output waveform. Here, the input given is 8 bit value a = 8'b00000111; b = 8'b00000101. After

the executing the test bench according to the verilog code the output waveform is achieved which is c=0000000000100011.

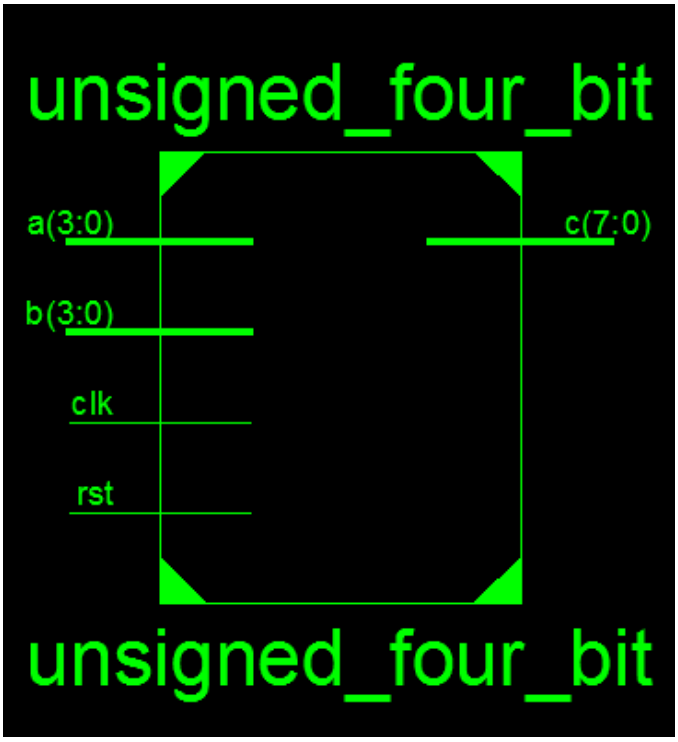

Figure 11: RTL Booth's 4-bit Unsigned

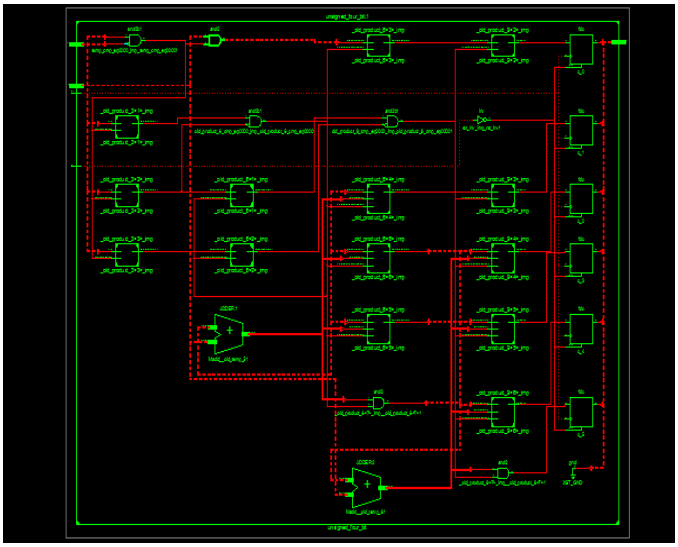

Figure 12: Array Structure of 4-Bit Unsigned

Figure12 shows the array structure and arrangement of the 4-bit unsigned booth multiplier.

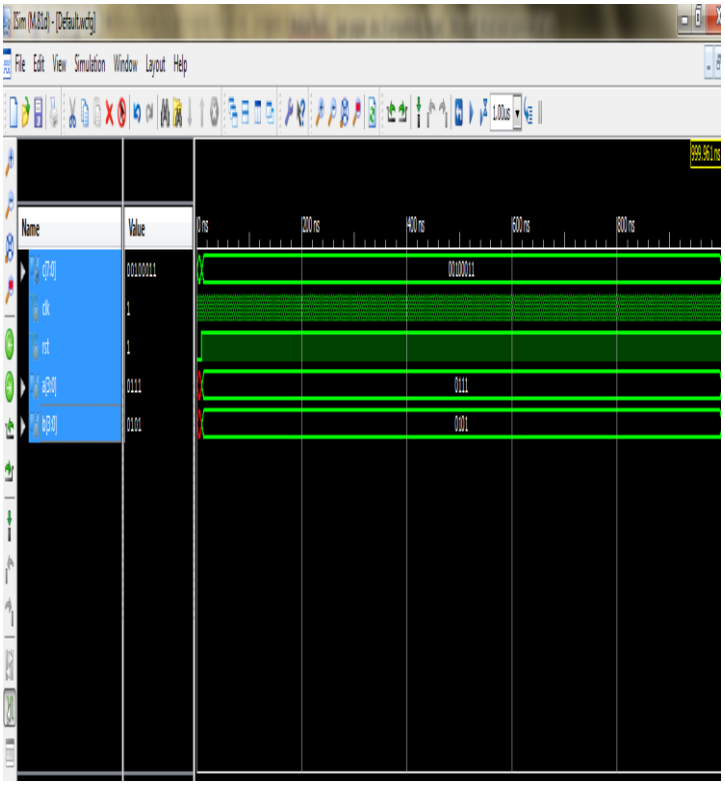

Figure13: Waveform of The Product (4-bit unsigned)

Figure13 shows the output waveform. Here, the input given is 4 bit value a = 4'b0111;b = 4'b0101;. After the executing the test bench according to the Verilog code the output waveform is achieved which is c=00100011

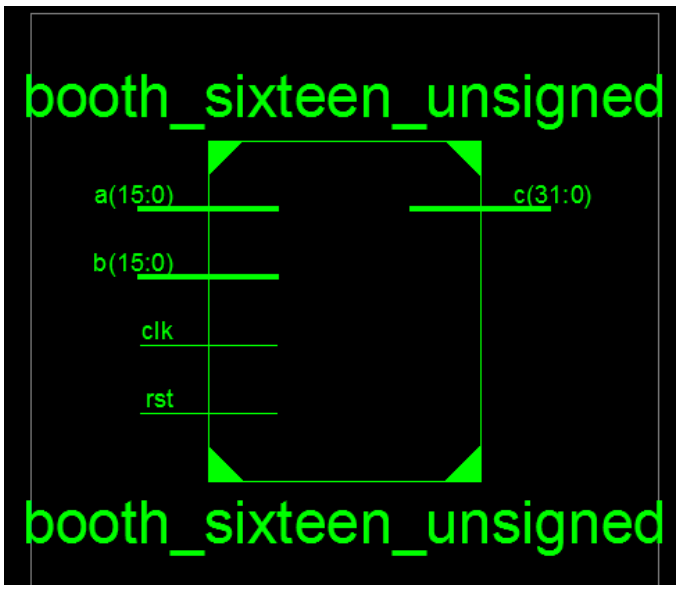

Figure 14: RTL Booth's 16-bit unsigned

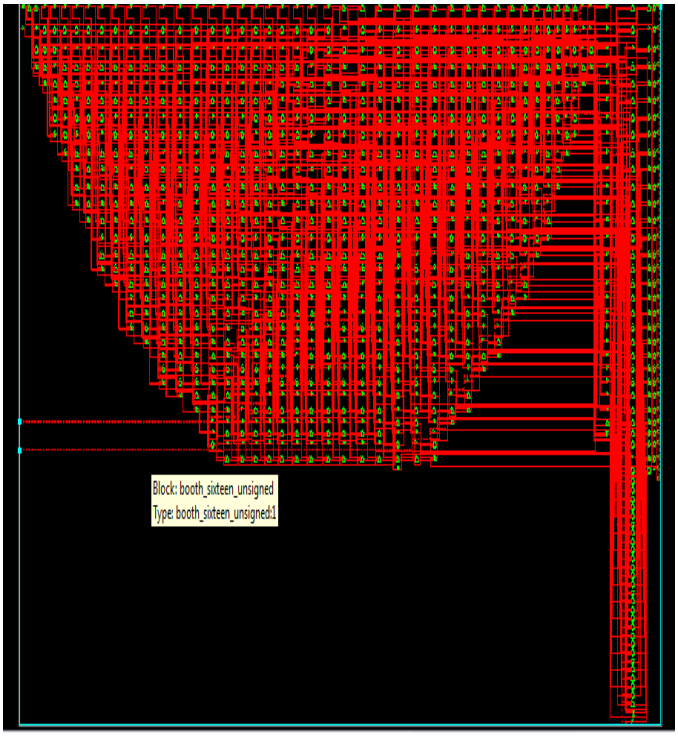

Figure 15: Array Structure Of 16-bit Unsigned

Figure15 shows the array structure and arrangement of the 16-bit unsigned booth multiplier.

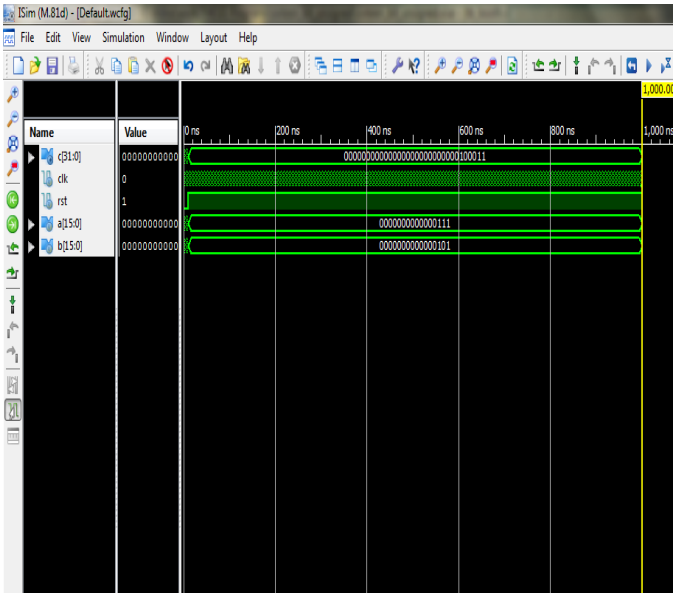

Figure 16: Waveform of The Product (16-bit unsigned)

Figure16 shows the output waveform. Here, the input given is 16bit valuea=16'b0000000000000111;b=16'b0000000000000101; After the executing the test bench according to the verilog code the output waveform is achieved which is c=00000000000000000000000000100011

TABLE 1: COMPARISON OF VARIOUS PARAMETERS

Parameter	4-bit Signed	4-bit Unsigned	8-bit Signed	8-bit Unsigned	16-bit Unsigned
Leakage Power	35.75 mW	35.57 mW	35.57 mW	35.57 mW	41.57 mW
Dynamic Power	1.30 mW	1.30 mW	2.09 mW	1.66 mW	2.12 mW
Total Power	36.87 mW	36.87 mW	37.66 mW	37.23 mW	43.69 mW
Inputs/Outputs used	18	18	34	34	66
Dynamic Current	1.09 mA	1.09 mA	1.74 mA	1.38 mA	1.77 mA
Quiescent Current	13.14 mA	13.14 mA	13.15 mA	13.14 mA	13.55 mA
Total Current	14.23 mA	14.23 mA	14.88 mA	14.53 mA	14.91 mA

From the above table of comparison we can see the difference in the various parameters of the multiplier of 4,8,16 bits. The 16-bit unsigned multiplier consumes the maximum power of 37.69 mW as compared to others the minimum power consumption is done by 4-bit singed and unsigned which is 36.87 mW. Also we can see the maximum current is consumed by the 16-bit unsigned multiplier 14.91 mA as compared to others the minimum current consumption is done by 4-bit singed and unsigned which is 14.23 ma.

IV. CONCLUSION

Modified Booth's multiplier has been successfully implemented for both signed and unsigned numbers using Xilinx 12.4 platform. Result obtained from simulation waveforms matches exactly with the desired value for 4 bit, 8 bit and 16 bit signed and unsigned multiplier. Different parameters like leakage power, dynamic power, total power, I/O pads used, dynamic and quiescent current etc have been compared for different bits. As expected, total power dissipated for 16 bit multiplier exceeds power for 8 bit multiplier. However the difference is not large.

REFERENCES

[1] W. C. Yeh and C. W. Jen, "High Speed Booth encoded Parallel Multiplier Design," IEEE transactions on computers, vol. 49, no. 7, pp. 692-701, July 2000.

[2] Shiann-Rong Kuang, Jiun-Ping Wang, and Cang-Yuan Guo, "Modified Booth multipliers with a Regular Partial Product Array," IEEE Transactions on circuits and systems-II, vol 56, No 5, May 2009.

[3] Li-Rong Wang, Shyh-Jye Jou and Chung-Len Lee, "A well-tructured Modified Booth Multiplier Design" 978-1-4244-1617-2/08/$25.00 ©2008 IEEE.

[4] Soojin Kim and Kyeongsoon Cho "Design of High-speed Modified Booth Multipliers Operating at GHz Ranges" World Academy of Science, Engineering and Technology 61 2010.

[5] Shiann-Rong Kuang, *Member, IEEE*, Jiun-Ping Wang, and Cang-Yuan Guo" Modified Booth Multipliers With a Regular Partial Product Array" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS, VOL. 56, NO. 5, MAY 2009 1549-7747

[6] *Ravindra P Rajput and M. N Shanmukha Swamy"* High speed Modified Booth Encoder multiplier for signed and unsigned numbers" , 2012 14th International Conference on Modelling and Simulation 978-0-7695-4682-7/12 © 2012 IEEE.

Manish Chaudhary born in Maharashtra on 25 september 1989. He completed the B.Tech in the field of Electronics and Communications from Greater Noida Institute of Technology (G.B.T.U, University) from Greater Noida in the year of 2011. Now, pursuing Master's Degree (M.Tech) in the field of Electronics and Communications from ITM University Gurgaon, Haryana.

Mandeep Singh Narula has completed M.Tech in Microelectronics and VLSI from IIT Kharagpur in the year 2008. His areas of interest are Low Power VLSI, RTL Verification, and Analog VLSI. He is working as Assistant Professor in ECE Department at ITM University, Gurgaon. He has over two years of experience in VLSI Industry and 3 years of teaching experience.