International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Comparative Analysis of Booth Multiplier using Radix-2 and Radix-4 Technique using VHDL

Niharika
Electronics & Communication.
S.I.T.M., U.P.T.U., Lucknow, (U.P.)
niharikaregecide@gmail.com

***Abstract*-In VLSI circuits area, power and delay are the key design factors. However, there exists a trade-off amongst them for an optimal design. Multiplier, being a very vital part in the design of microprocessor, graphical systems, multimedia systems, DSP system etc. Nearly 15 percent of total IC power is consumed by multiplication alone. It is very important to have an efficient design in terms of performance, area, speed of the multiplier and for the same Booth's multiplication algorithm provides a very fundamental platform for all the new advances made for high end multipliers meant for faster multiplication with higher performance. The algorithm provides an efficient encoding of the bits during the first steps of the multiplication process. This paper is based on configurable logic for 16-bit Booth multiplier using Radix2 and radix 4 Method. Booth multiplier can be configured to perform multiplication on 16-bit operands. The multiplier will detect the range of the operands through configuration register. The configuration register can be configured through input ports. The multiplier has been synthesized using Xilinx 14.5 and it has achieved a minimum combinational delay. Modelsim is responsible for simulation part in this work.**

Keywords*-- *Booth Multiplier, Radix-2, Radix-4.

1. Introduction

Arithmetic and logic operations play an important role in digital circuits. Addition, multiplication, exponensation are important fundamental function in arithmetic operations and have wide applications in the field of engineering. The demand for high speed processing has been increasing as a result of expanding computer and signal processing applications. Higher throughput arithmetic operations are important to achieve the desired performance in many real-time signal and image processing applications [1]. Among these arithmetic operations multiplication is the key of almost every digital circuit. It is used extensively in many VLSI systems such as communication system architectures and microprocessors. Multiplication-based operations such as Multiply and Accumulate (MAC) and inner product are among some of the frequently used Computation-Intensive Arithmetic Functions (CIAF) currently implemented in many Digital Signal Processing (DSP) applications such as Convolution, Fast Fourier Transform (FFT), filtering, in microprocessors in its arithmetic and logic unit and in graphics [2]. Digital multipliers are the most commonly used components in any digital circuit design. They are fast, reliable and efficient components that are utilized to implement any operation. Depending upon the arrangement of the components, there are different types of multipliers available each offering different advantages and having tradeoff in terms of speed, circuit complexity, area and power consumption. Reducing the time delay and power consumption are very essential requirements for many applications [1][3].Multipliers have become a basic building block in computations especially in digital signal processing. Multipliers not only take a significant part of time delay, area cost but also cause high power consumption. To improve the speed and power dissipation of the multipliers, many techniques and design methodologies have been proposed. Most of the designs are targeted at a specific technology and require redesign for a new process technology. As a result, it is necessary to develop computation-efficient multipliers suitable for portable multimedia and digital processing systems, which require flexible processing ability, lesser switching activity and short design cycle. The biggest challenge faced with use of simple and conventional multipliers for multimedia and DSP systems is that the multiplier coefficients are not constant. If it is constant, a general multiplier can be simplified to a network of shift, adders and subtractors to reduce power consumption [4]. However, this kind of simplified multiplier is inflexible which makes it to be unsuitable for multiplication operations with varying coefficients. To achieve improved processing ability, various techniques for reconfigurable multipliers that are capable of supporting multiple-precision multiplications have been developed. Advanced VLSI technology has given designer the freedom to integrate many complex components, which was not possible in the past. Various high speed multipliers have been proposed and realized [5]. Among these multiplication algorithms Booth's multiplication is showing better performance. In this dissertation, an attempt has been made to combine configuration and range detection technique to design configurable Booth multiplier (CBM) that supports single 4-bit, single 8-bit, single 12-bit or single 16-bit multiplication. This CBM depends upon the output of the range detection technique with highly simplified circuit [16].

2. Related Work:

Different adders are compared by using critical parameters like Delay, Power, and Area etc. to make clear ideas of which adder was best suited for situation. After comparing all, it was concluded that Carry Select Adders are best suited for situations where speed is the critical concern [6]. Coming to Multipliers ,implementation of Radix-2 Booth Multiplier is done using different adder and came to final conclusion that parallel multipliers are much better than the serial multipliers due to less area consumption and hence the less power consumption [6].

In many DSP applications, all of multiplier output bits were not used, but only upper bits of output were used. Kidamhi

proposed truncated unsigned multiplier for this idea. This truncation scheme can be applied to Booth multiplier which can be used in real DSP systems more efficiently. Also, truncated Booth multiplier guaranteed 0 input to 0 output that was not provided before. Truncated Booth multiplier reduced about 37-48 % of area and about 44 % of power consumption [7].

Different design methods are proposed to develop a modular approach for optimizing power consumption[7]. It is found that algorithm based design reduce gate switching activity considerably and as result power consumption in multiplier is reduced [8]. It is found that data complexity and various combination of gate level digital circuit has considerable impact in power dissipation. Beside this physical design of the chip can be optimized by using Genetic Algorithm by analyzing placement option subject to optimum space allocation. Similarly selection of Booth Algorithm and Modified Booth Algorithm may reduce power consumption as consequence of data complexity. It is found that in multiplier circuit, Modified Booth Algorithm reduces power consumption as compared to other methods of multiplication [8].

On making a comparison between radix 2 and radix 4 Booth multiplier in terms of power saving experimental results demonstrate that the modified radix 4 Booth multiplier has 22.9% power reduction than the conventional radix 2 Booth Multiplier [9].

Booth multiplier can be configured based on dynamic range detection of multipliers and optimized for low power and high speed operations and which can be configured either for single 16-bit multiplication operation, single 8-bitmultiplication or twin parallel 8-bit multiplication is designed [10]. It was found that Booth Multiplier can efficiently deactivate ineffective circuitry which were not produced effective result so that speed of operation gets increases and device area is reduced so that power gets reduced .

3. Methodology:

Signed multiplication is a vigilant process. Through unsigned multiplication there is no need to take the sign of the number into consideration. Even though in signed multiplication the same procedure cannot be applied for the reason that the signed number is in a 2‘s compliment form which would give inaccurate result if multiplied in an analogous manner to unsigned multiplication [11]. Unsigned multipliers cannot be applied to most of the multimedia and DSP applications due to their signed multiplication operation [4]. For the standard add-shift operation, each multiplier bit generates one multiple of the multiplicand to be added to the partial product. If the multiplier is very large, then a large number of multiplicands have to be added. In this case the delay of multiplier is determined mainly by the number of additions to be performed. If there is a way to reduce the number of the additions, the performance will get better. Thus here Booth‘s algorithm comes in rescue. Booth‘s algorith m conserves the sign of the end result.

In 1951, Andrew Donald Booth devised a multiplication algorithm, while doing study on crystallography at Birkbeck College in Bloomsbury, London known as Booth’s Algorithm. Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed [12]. It is a powerful algorithm for signed-number multiplication, which treats both positive and negative numbers uniformly. This algorithm multiplies two signed binary numbers in two’s complement notation. The main objective of designing the booth multiplier is to perform the partial products to reduce the delay and to increase the speed of the circuit. In this it also reduce the area of the chip used so the power consumption is also reduced in this circuit.

Donald Booth made an improvement in the multiplier by reducing the number of partial products generated. The Booth recording multiplier scans the three bits at a time to reduce the number of partial products [13]. These three bits are: the two bit from the present pair; and a third bit from the high order bit of an adjacent lower order pair. After examining each triplet of bits, the triplets are converted by Booth logic into a set of five control signals used by the adder cells in the array to control the operations performed by the adder cells.

To speed up the multiplication Booth encoding performs several steps of multiplication at once. Booth’s algorithm takes advantage of the fact that an adder, subtractor is nearly as fast and small as a simple adder.

From the basics of Booth Multiplication it can be proved that the addition/subtraction operation can be skipped if the successive bits in the multiplicand are same. If 3 consecutive bits are same then addition/subtraction operation can be skipped. Thus in most of the cases the delay associated with Booth Multiplication are smaller than that with Array Multiplier. However the performance of Booth Multiplier for delay is input data dependant. In the worst case the delay with booth multiplier is on par with Array Multiplier [14].

The method of Booth recording reduces the numbers of adders and hence the delay required to produce the partial sums by examining three bits at a time. The high performance of booth multiplier comes with the drawback of power consumption. The reason is large number of adder cells required that consumes large power [15].

A. Flow Chart of Booth Multiplier

Booth's multiplication algorithm is an algorithm which multiplies 2 signed integers in 2's complement. The algorithm is depicted in the figure 1 with a brief description. This approach uses fewer additions and subtractions than more straightforward algorithms.

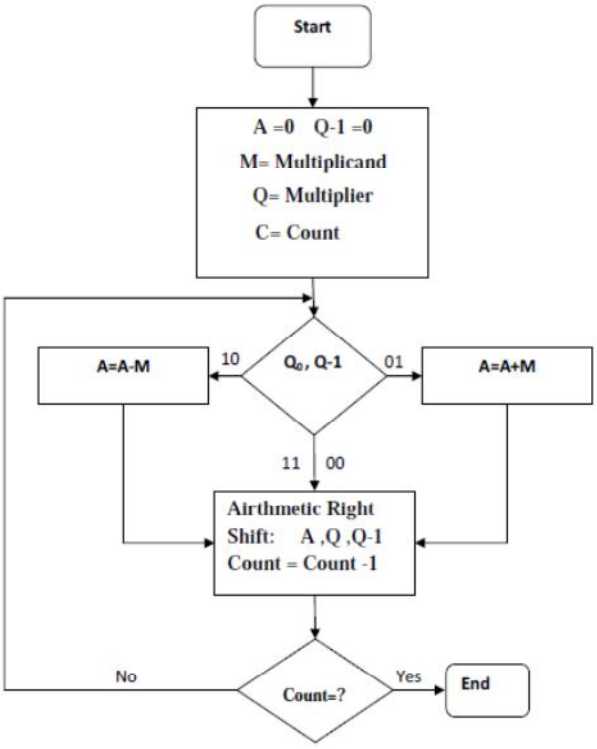

Fig. 1. Flow Chart of Booth Multiplier

The multiplicand and multiplier are placed in the M and Q registers respectively. A 1-bit register is placed logically to the right of the LSB (least significant bit) Q0 of Q register. This is denoted by Q_{-1}. A and Q_{-1} are initially set to 0. Control logic checks the two bits Q_0 and Q_{-1}. If the two bits are same (00 or 11) then all of the bits of A, Q and Q_{-1} are shifted 1 bit to the right. If they are not the same and if the combination is 10 then the multiplicand is subtracted from A and if the combination is 01 then the multiplicand is added with A. In both the cases results are stored in A, and after the addition or subtraction operation, A, Q and Q_{-1} are right shifted. The shifting is the arithmetic right shift operation where the left most bit namely, A_{n-1} is not only shifted into A_{n-2} but also remains in A_{n-1}. This is to preserve the sign of the number in A and Q. The result of the multiplication will appear in the A and Q.

B. Booth Multiplier Architecture

In the booth multiplier architecture there are various components that are performing certain task as follows:

(1) A 16-bit register M that stores the multiplicand.
(2) 16-bit parallel-load shift register Q that stores the multiplier initially and the least significant 16 bits of the final multiplication product.
(3) The 16-bit parallel-load shift registers ACCUMULATOR that is cleared initially and will store the most significant 16 bits of the final multiplication product. ACCUMULATOR and Q are concatenated such that the bit that shifted out of ACCUMULATOR will be shifted into Q. Also, the right shift operation is sign extended.

For example, if ACCUMULATOR stores 1111111111111101 and Q stores 0100000000001111 then after concatenated right shift operation, ACCUMULATOR = 1111111111111110 and Q =1010000000000111. Note that the least significant bit in ACCUMULATOR (= 1) originally has been shifted to Q as most significant bit.

(4) The 16-bit arithmetic logic unit performs (ALU) will perform the arithmetic operation of addition, subtraction or shifting according to the control signal given by control block and finally pass its output to 16-bit ACCUMULATOR.
(5) For controlling the operation of the multiplier CONTROL block Y is used. The control block provides the necessary control signals depending upon the input operands.
(6) A 4-bit binary COUNTER is used that give reference count to the CONTROL block.

Figure 2 shows the diagram of the Booth's multiplier which multiplies two 16-bit numbers in 2's complement.

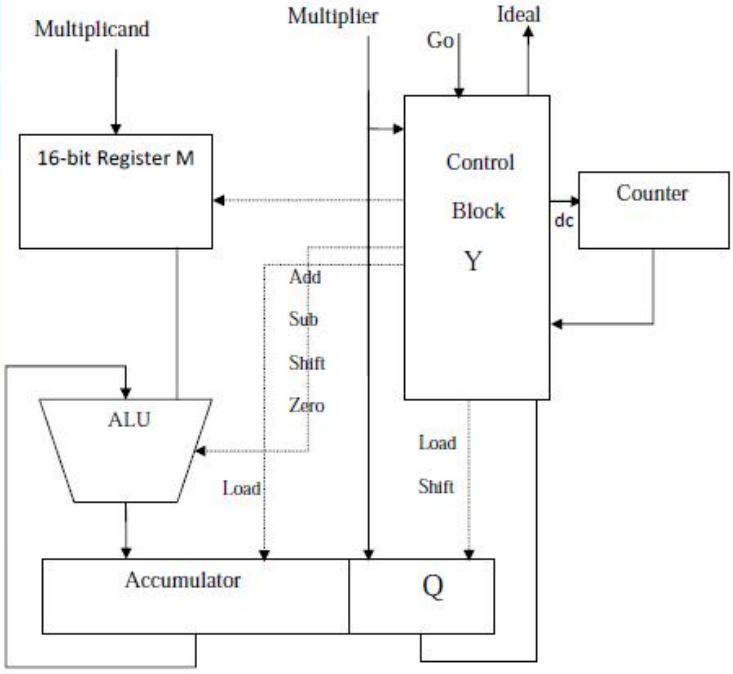

Fig. 2. Booth Multiplier Architecture

Booth's algorithm can be implemented in many ways. This experiment is designed using a controller and a data path. The operations on the data in the data path are controlled by the control signal received from the controller. The data path contains registers to hold multiplier, multiplicand, intermediate results, data processing units like ALU, adder/subtractor, counter and other combinational units. Here the adder/subtractor unit is used as data processing unit. M holds the multiplicand, Q holds the multiplier. The counter is a down counter which counts the number of operations needed for the multiplication. The data flow in the data path is controlled by the five control signals generated from the controller, these signals are *load* (to load data in registers), *add* (to initiate addition operation), *sub* (to initiate subtraction operation), *shift* (to initiate arithmetic right shift operation), *dc* (this is to decrement counter). The controller generates the control signals according to the input received from the data path. Here the inputs are the

least significant Q_0 bit of Q register, Q_{-1} bit and count bit from the down counter.

Recoding scheme used in radix-2 booth multiplier is shown in the Table 1.

Table 1: Recoding Table for Booth Multiplier

Qn	Qn+1	Recoded Bits	Operation
0	0	0	Shift
0	1	+1	Add X
1	0	-1	Subtract X
1	1	0	Shift

Table. 2. Modified Radix 4 Recoding Rules

B	Zn	Partial Product
000	0	0
001	1	1×Multiplicand
010	1	1×Multiplicand
011	2	2×Multiplicand
100	-2	-2×Multiplicand
101	-1	-1×Multiplicand
110	-1	-1×Multiplicand
111	0	0

4. Result and Discussion:

Simulation is the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed, this model represents the key characteristics of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time. For the model under consideration the simulation results are carried using VHDL as simulation language and Modelsim as simulator. In the design of Booth multiplier for radix 2 there are two inputs namely multiplier [7 :0] and multiplicand [7 : 0] and the single output Product [16:0]. Output is verified for inputs of different ranges. In this case the simulation results are shown for input value in binary for multiplier "00000010" and for multiplicand "00000010" and we will get the output in product is "0000000000000100" shown in fig 3:

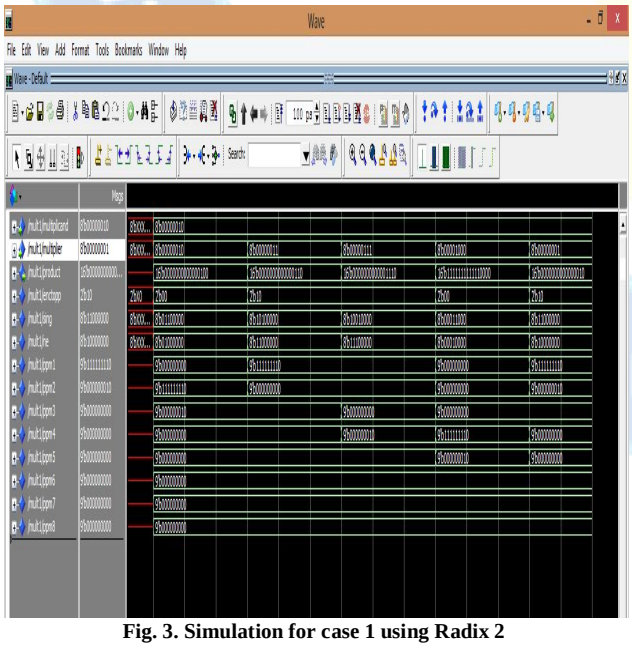

Fig. 3. Simulation for case 1 using Radix 2

In the design of Booth multiplier for radix 2 there are two inputs namely multiplier [7 :0] and multiplicand [7 : 0] and the single output Product [16:0]. Output is verified for inputs of different ranges. In this case the simulation results are shown for input value in binary for multiplier "00000010" and for multiplicand "00000011" and we will get the output in product is "0000000000000110" shown in fig 4:

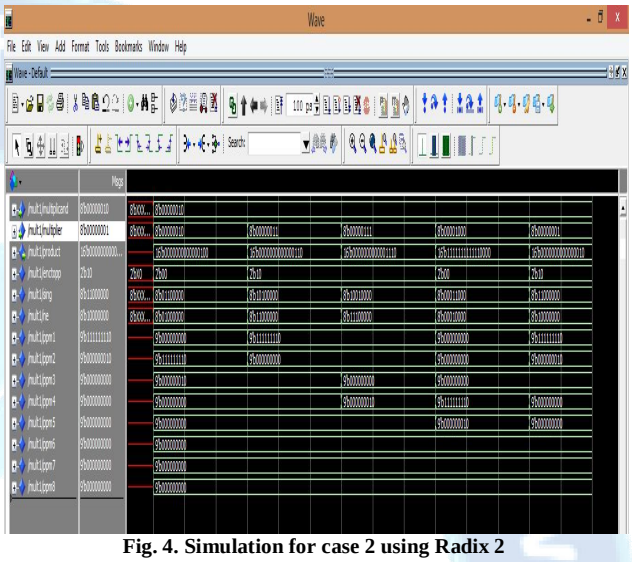

Fig. 4. Simulation for case 2 using Radix 2

The multiplier has been synthesized using Xilinx ISE 8.1i. The RTL Schematic of Booth Multiplier using Radix 2 has been shown in figure 5 In the RTL schematic of Booth multiplier multiplicand[7:0] and Multiplier [7:0] represents the 8-bit input operands and Product(16:0) represents 16-bit output product, fig 6 shows the internal RTL of the Booth Multiplier.

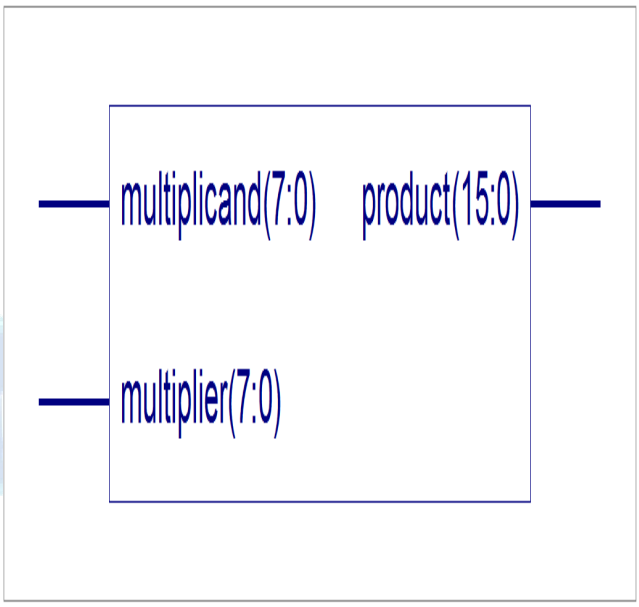

Fig. 5. RTL of Booth Multiplier using Radix 2

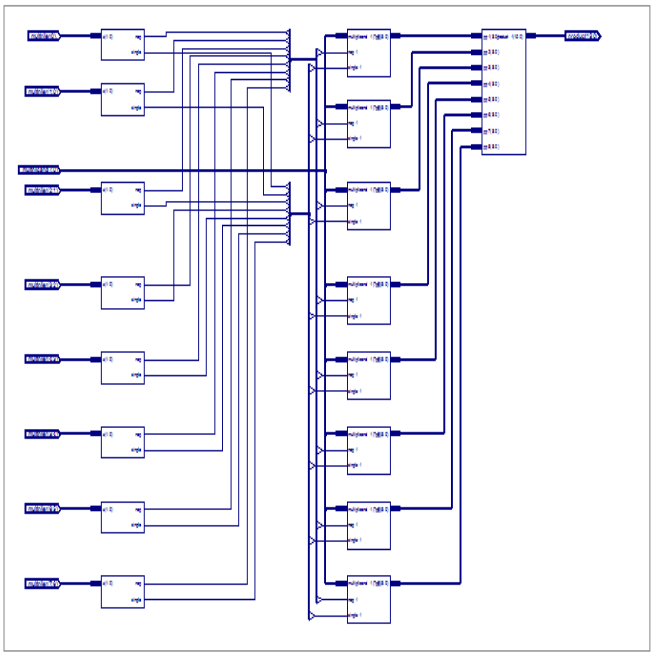

Fig. 6. Internal RTL of Booth Multiplier using Radix 2

In the design of Booth multiplier for radix 4 there are two inputs namely multiplier [7 :0] and multiplicand [7 : 0] and the single output Product [16:0]. Output is verified for inputs of different ranges. In this case the simulation results are shown for input value in binary for multiplier "00000010" and for multiplicand "00000010" and we will get the output in product is "0000000000000100" shown in fig 7:

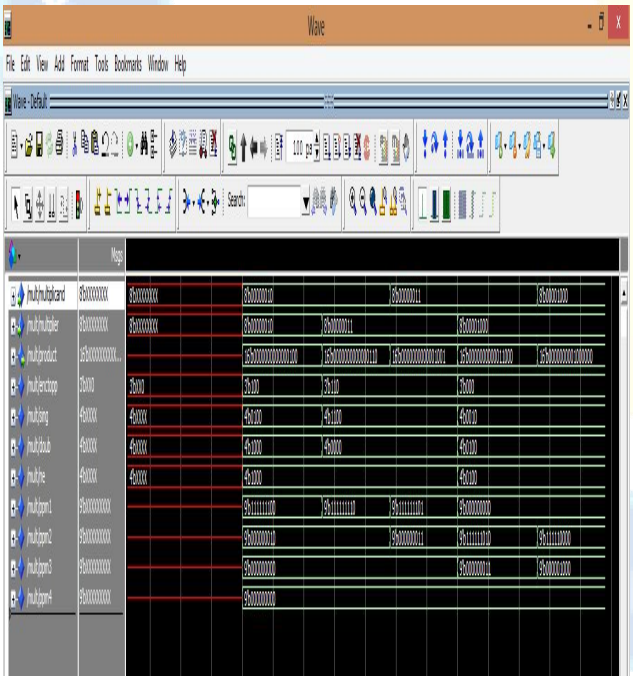

Fig. 7. Simulation for case 1 using Radix 4

In the design of Booth multiplier for radix 4 there are two inputs namely multiplier [7 :0] and multiplicand [7 : 0] and the single output Product [16:0]. Output is verified for inputs of different ranges. In this case the simulation results are shown for input value in binary for multiplier "00000010" and for multiplicand "00000011" and we will get the output in product is "0000000000000110" shown in fig 8:

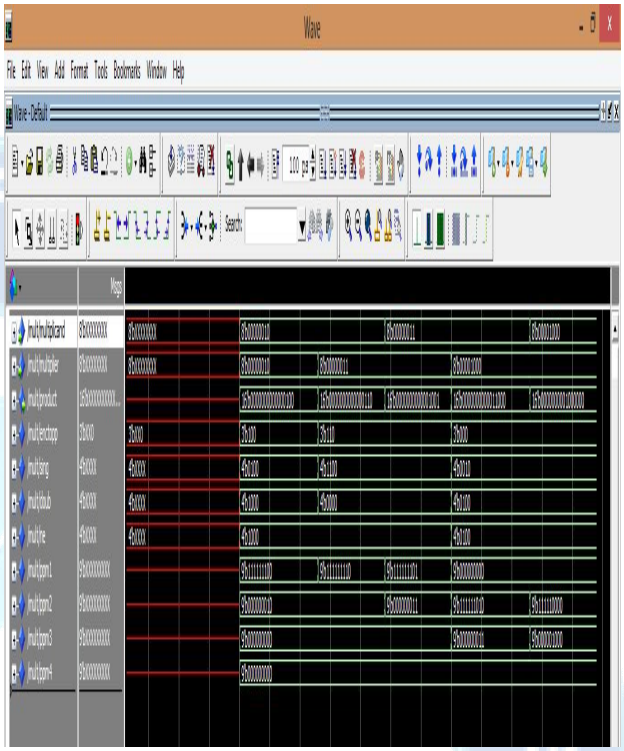

Fig. 8. Simulation for case 2 using Radix 4

The multiplier has been synthesized using Xilinx ISE 8.1i. The RTL Schematic of Booth Multiplier using Radix 4 has been shown in figure 9 In the RTL schematic of Booth multiplier multiplicand[7:0] and Multiplier [7:0] represents the 8-bit input operands and Product(16:0) represents 16-bit output product, fig 4.10 shows the internal RTL of the Booth Multiplier.

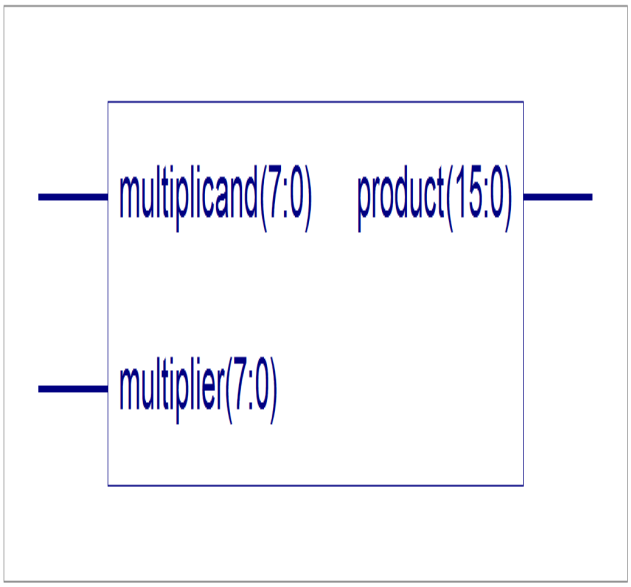

Fig. 9. RTL of Booth Multiplier using Radix 4

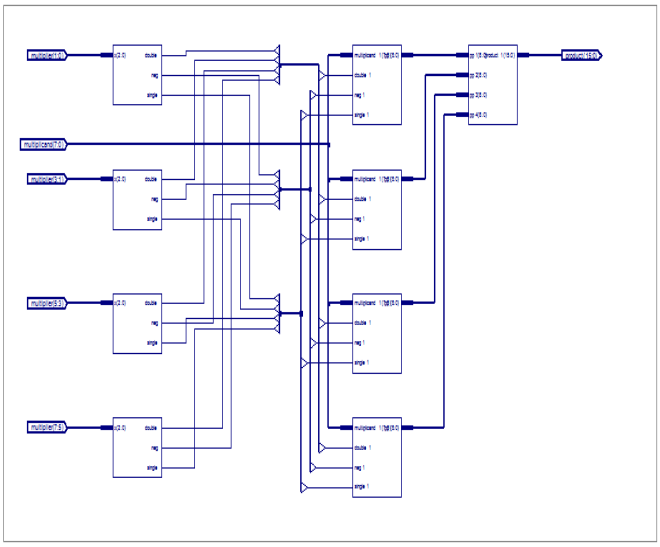

Fig. 10. Internal RTL of Booth Multiplier using Radix 4

5. Conclusion:

After going through all the hard work and facing problems, this project managed to complete its objectives that are to study the Booth Multiplier and design of Delay Efficient Booth Multiplier using Radix-2 and Radix-4 Method. The main objective of designing the Booth multiplier is to perform the partial products to reduce the delay, increase the speed of operation and to reduce the power consumption in the circuit. We have presented a 16-bit Booth multiplier using Radix2 and Radix4 Method. This multiplier can be configured to perform 16 bit multiplication depending upon the output of configuration register. The multiplier will detect the range of the operands through configuration register. The configuration register can be configured through input ports. It also deactivates the redundant switching activities in ineffective ranges as much as possible. Moreover, the output product of the multiplier can be truncated to further decrease power consumption by sacrificing a bit of output precision. The multiplier has been synthesized using Xilinx.

References:

[1] Himanshu Thapliyal and Hamid R. Arabnia, “A Time-Area-Power Efficient Multiplier and Square Architecture Based On Ancient Indian Vedic Mathematics”, Department of Computer Science, The University of Georgia, 415 Graduate Studies Research Center Athens, Georgia 30602-7404, U.S.A ,2003.

[2] Purushottam D. Chidgupkar and Mangesh T. Karad, “The Implementation of Vedic Algorithms in Digital Signal Processing”, Global J. of Engng. Educ., Vol.8, UICEE Published in Australia. 2004.

[3] E. Abu-Shama, M. B. Maaz, M. A. Bayoumi, “A Fast and Low Power Multiplier Architecture”, The Center for Advanced Computer Stu dies, The University of Southwestern Louisiana Lafayette, 2007.

[4] Shiann Rong Kuang and Jiun-Ping Wang “ Design of Power Efficient Configurable Booth Multiplier” Vol. 57, No.3, March 2010

[5] A. D. Booth, “A Signed Binary Multiplication Technique”, Quarterly J. Mech. Appli. Math., vol 4, part2, pp. 236-240 , 1951

[6] Sakshi Rajput,Priya Sharma,Gitanjali and Garima “High Speed and Reduced Power – Radix-2 Booth Multiplier” International Journal of Computational Engineering & Management, Vol. 16, Issue 2, March 2013

[7] Kwang Hyun Lee , Chong Suck Rim, “ A Hardware Reduced Multiplier for Low Power Design” Proceedings of the second IEEE Asia Pacific Conference,Korea, 2000.

[8] Zamin Ali Khan ,S. M. Aqil Burney , Jawed Naseem, Kashif Rizwan. “Optimization of Power Consumption in VLSI Circuit” International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011

[9] Nishat Bano “VLSI Design of Low Power Booth Multiplier” International Journal of Scientific & Engineering Research, Volume 3, Issue 2, February -2012

[10] J. Sreenivasulu Reddy , Y.Avanija , M. Mahesh Babu “Dynamic Range Detection Based Booth multiplier for Low Power and high Speed Applications” International Journal of Emerging trends in Engineering and Development Issue 2, Vol.6, September 2012.

[11] Laxman S, Darshan Prabhu R, Mahesh S. Shetty ,Mrs. Manjula BM, Dr. Chirag Sharma, “FPGA Implementation of Different Multiplier Architectures” , International Journal of Emerging, vol 3, 2009.

[12] L. D. Van and C. C. Yang, “Generalized low-error ar ea efficient fixed width multipliers,” IEEE Trans. Circuits System, Reg. Papers, vol. 52, no. 8, pp. 1608–1619, Aug. 2005.

[13] Oscal T. ,C. Chen, Sandy Wang, and Yi-Wen Wu, “Minimization of Switching Activities of Partial Products for Designing Low-Power Multipliers”, IEEE Transactions on VLSI Systems, vol. 11, no. 3, June 2003

[14] Tam Anh Chu, “Booth Multiplier with Low Power High Performance Input Circuitary”, US Patent, 6.393.454 B1, May 21, 2002.

[15] Pravin kumar Parate “ASIC Implementation of 4 Bit Multipliers”, IEEE Computer society. ICETET, 2008.

[16] R. Sharma et. al., "Implementation of N-Bit Divider using VHDL" International Journal of Research and Development in Applied Science and Engineering, Volume 3, Issue 1, March 2013.