

Design, Comparison and Implementation of Multipliers on FPGA

B Naga Venkata Satya Durga Jahnavi, Shivani Mupparaju, Dr.L Padmasree

Abstract— In this paper comparison of array, booth radix-2 and booth radix-4 multipliers have been presented. Low power consumption and smaller area are some of the most important criteria for the fabrication of DSP systems and high performance systems. Optimizing the speed and area of the multiplier is a major design issue. In this paper we determine the best solution to this problem by comparing a few multipliers. The proposed architectures are synthesized using Xilinx tool and implemented on FPGA. Based on the theoretical and experimental estimation, analysis was carried on results such as the amount of hardware resources and delay. Proposed multipliers can be used for high performance applications like signal processing, image processing.

Index Terms—Array multiplier, Booth multiplier, Xilinx, FPGA

I. INTRODUCTION

Multiplication is an important fundamental function in arithmetic operations. Multiplication-based operations such as Multiply and Accumulate (MAC) is used in many Digital Signal Processing (DSP) applications such as convolution, Fast Fourier Transform (FFT), filtering and in the arithmetic and logic unit of microprocessors. Since multiplication dominates the execution time of most DSP algorithms and determines the speed of ALU, there is a need of high speed multiplier. The power dissipation arises from the charging and discharging of the circuit node capacitances found on the output of every logic gate. Power management is the careful planning of power budget for every subsystem of a VLSI chip. This is especially important issue for today's complex systems. The most important and successful use of power management is to deactivate a portion of circuit when its computation is not required. Every low-to-high logic transition in a digital circuit incurs a change of voltage, drawing energy from the power supply.

In this paper, analysis is done on these multipliers and they are compared in terms of number of gates, LUT's used and power consumption. Among these multipliers, Booth Multiplier is optimized in terms of power consumption

II. OBJECTIVES AND TOOLS EMPLOYED

A. Objective of the project

The main objective of this paper is to compare various

Manuscript received September 20, 2013

B Naga Venkata Satya Durga Jahnavi, Electronics and Communication Engineering, VNR VJIEET, Hyderabad, India.

Shivani Mupparaju, Electronics and Communication Engineering, VNRVJIEET, Hyderabad, India.

L.Padmasree, Electronics and Communication Engineering, VNR VJIEET, Hyderabad, India.

multipliers and determine the best among them in terms of delay and power consumption with the help of Xilinx Timing and Power Analyzers. Then they are targeted on FPGA

B. Tools Used

Simulation Software: ISE 10.1 for design and implementation and Model Sim 6.1 is used for Modelling and Simulation

C. Hardware Used

Xilinx vertex 2p (Family), XC2VP30 (Device), FG (Package) FPGA device.

III. TYPES OF MULTIPLIER ARCHITECTURES

A. Array multiplier

A Binary multiplier is an electronic hardware device used in digital electronics or a computer or other electronic device to perform rapid multiplication of two numbers in binary representation. It is built using binary adders.

The rules for binary multiplication can be stated as follows

1. If the multiplier digit is a 1, the multiplicand is simply copied down and represents the product.
2. If the multiplier digit is a 0 the product is also 0.

For designing a multiplier circuit we should have circuitry to provide or do the following three things:

1. It should be capable identifying whether a bit 0 or 1 is.
2. It should be capable of shifting left partial products.
3. It should be able to add all the partial products to give the products as sum of partial products.
4. It should examine the sign bits. If they are alike, the sign of the product will be a positive, if the sign bits are opposite product will be negative. The sign bit of the product stored with above criteria should be displayed along with the product.

From the above discussion we observe that it is not necessary to wait until all the partial products have been formed before summing them. In fact the addition of partial product can be carried out as soon as the partial product is formed.

Notations:

a – Multiplicand

b – Multiplier

p – Product

Binary multiplication (eg n=4)

$p = a \times b$

a: $a_{n-1} a_{n-2} \dots a_1 a_0$

b: $b_{n-1} b_{n-2} \dots b_1 b_0$

p: $p_{n-1} p_{n-2} \dots p_1 p_0$

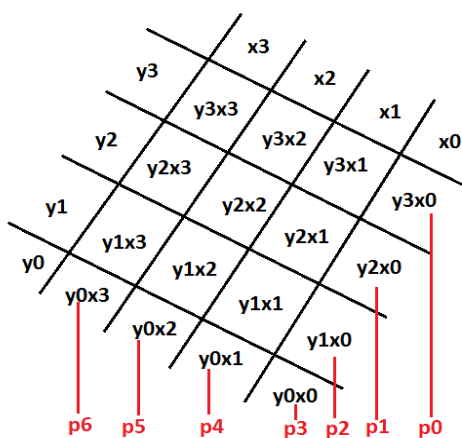
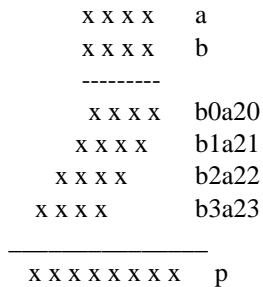


Fig .1 Proposed Architecture of 4x4 array multiplier

B .Booth Multiplier

Major limitation of array multiplier is its size. As operand sizes increase, arrays grow in size at a rate equal to the square of the operand size hence speed of multiplier reduces .In order to increase the speed of multiplier booth algorithm is used. The Booth multiplier makes use of Booth encoding algorithm in order to reduce the number of partial products by considering two bits of the multiplier at a time, thereby achieving a speed advantage over other multiplier architectures. This algorithm is valid for both signed and unsigned numbers. It accepts the number in 2's complement form, based on radix-2 computation.

Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation. I will illustrate the booth algorithm with the following example:

Example, 2 ten x (- 4) ten

$0010 \text{ two} * 1100 \text{ two}$

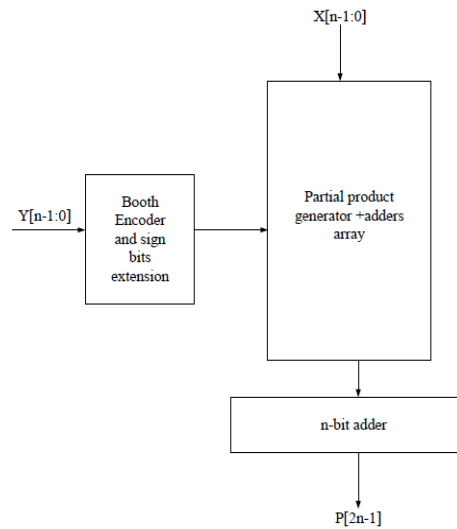


Fig. 2 Block Diagram of Booth Multiplier

Booth Encoder

1) *Making the Booth table:*

- I. From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier.
 - i. i.e., 0010 -- From 0 to 0 no change, 0 to 1 one change, 1 to 0 another change, so there are two changes on this one.
 - ii. 1100 -- From 1 to 1 no change, 1 to 0 one change, 0 to 0 no change, so there is only one change on this one.
 - iii. Therefore, multiplication of 2 x (- 4), where 2 ten (0010 two) is the multiplicand and (- 4) ten (1100two) is the multiplier.

II. Let X = 1100 (multiplier)

Let Y = 0010 (multiplicand)

Take the 2's complement of Y and call it -Y

$-Y = 1110$

III. Load the X value in the table.

IV. Load 0 for X-1 value it should be the previous first least significant bit of X.

V. Load 0 in U and V rows which will have the product of X and Y at the end of operation.

VI .Make four rows for each cycle; this is because we are multiplying four bits numbers.

Table 2.1. Step1 ofRadix-2 Booth multiplication

U	V	X	X-1	
0000	0000	1100	0	Load the value
				1 st cycle
				2 nd cycle
				3 rd Cycle
				4 th Cycle

2) *Booth Algorithm*

Booth algorithm requires examination of the multiplier bits, and shifting of the partial product. Prior to the shifting, the multiplicand may be added to partial product, subtracted from the partial product, or left unchanged according to the following rules:

Look at the first least significant bits of the multiplier “X”, and the previous least significant bits of the multiplier “X - 1”.

- I. 0 0 Shift only
- II. 1 1 Shift only.
- III. 0 1 Add Y to U, and shift
- IV. 1 0 Subtract Y from U, and shift or add (-Y) to U and shift
- V. Take U & V together and shift arithmetic right shift which preserves the sign bit of 2’s complement number. Thus a positive number remains positive, and a negative number remains negative.
- VI. Shift X circular right shift because this will prevent us from using two registers for the X value.
- VII. Repeat the steps until four cycles are completed.

Table 2.2 Step3 in Radix-2 Booth Multiplication Process

U	V	X	x-1
0000	0000	1100	0
0000	0000	0110	0
0000	0000	0011	0
1110	0000	0011	0 ← Add-Y(0000+1110)=1110
1111	0000	1001	1 ← shift

Table 2.3 Step4 in Radix-2 Booth Multiplication Process and Final Result in the Final Cycle

U	V	X	x-1
0000	0000	1100	0
0000	0000	0110	0
0000	0000	0011	0
1110	0000	0011	0
1111	0000	1001	1
1111	1000	1100	1 ← Shift only

After finishing four cycles, so the answer is shown, in the last rows of U and V which is: 11111000 two

NOTE: By the fourth cycle, the two algorithms have the same values in the Product register.

Group of consecutive 0’s in multiplier - no new partial product - only shift partial product right one bit position for every 0
Group of m consecutive 1’s in multiplier - less than m partial products generated

$$\dots 01\dots 110\dots = \dots 10\dots 000\dots - \dots 00\dots 010\dots$$

Using SD (signed-digit) notation = $\dots 100\dots 010\dots$

Example:

$$\dots 011110\dots = \dots 100000\dots - \dots 000010\dots = \dots 100010\dots$$

(Decimal notation: $15=16-1$)

Instead of generating all m partial products - only 2 partial products generated First partial product added - second subtracted-number of single-bit shift-right operations still m.

Table 2.4. Radix-2 Booth Multiplier Logic

X_i	X_{i-1}	Operation	comments	Y_i
0	0	Shift only	String of zeros	0
1	1	Shift only	String of ones	0
1	0	Subtract and shift	Beginning of a string of ones	1
0	1	Add and shift	End of a string of ones	1

Recoding multiplier $x_{n-1} x_{n-2} \dots x_1 x_0$ in SD code. Recoded multiplier $y_{n-1} y_{n-2} \dots y_1 y_0$. x_i, x_{i-1} of multiplier examined to generate y_i . Simple recoding is $y_i = x_{i-1} - x_i$.

Example: Multiplier 0011110011(0) recoded as 0100010101 - 4 instead of 6 add/subtracts.

Drawbacks Of Radix-2 Booth Multiplier

There is a problem of isolated one’s. To overcome this drawback, instead of comparing two bits at a time, we are going for comparison three bits at a time

3) Modified Booth Encoder

Modified Booth encoding is most often used to avoid variable size partial product arrays. Before designing a MBE, the multiplier B has to be converted into a Radix-4 number by dividing them into three digits respectively according to Booth Encoder Table given afterwards. Prior to convert the multiplier, a zero is appended into the Least Significant Bit (LSB) of the multiplier. In this approach instead of eight partial products being generated using conventional multiplier only 4 partial products are generated. Table 4.2. shows the truth table for a Booth encoder. The encoder takes inputs Y_{N+1}, Y_N and Y_{N-1} from the multiplier bus and produces a 1 or a 0 for each operation: single, double, negative(X, 2X and Neg).

Table:3.1 Truth table for Modified Booth Encoder

Y_{N+1}	Y_N	Y_{N-1}	Operation	X	2X	Neg
0	0	0	+0 x X	0	0	0
0	0	1	+1 x X	1	0	0
0	1	0	+1 x X	1	0	0
0	1	1	+2 x X	0	1	0
1	0	0	-2 x X	0	1	1
1	0	1	-1 x X	1	0	1
1	1	0	-1 x X	1	0	1
1	1	1	-0 x X	0	0	1

This Booth multiplier technique is to increase speed by reducing the number of partial products by half. The operand that is booth encoded is called multiplier, and the other operand is called multiplicand.

In most of the cases MBE scheme is used for generating PP, because of its ability to reduce the number of PP by half[7]. The truth table shows the function of booth encoder. If a 3-bit binary input sequence is given at the input, and perform the operation as mentioned in front of it, the partial products will be generated.

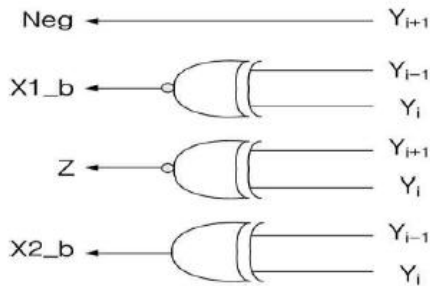


Fig.3 Booth Encoder

IV. RESULTS AND ANALYSIS

A) SIMULATION

The different multipliers are simulated using Xilinx 10.1 ISE simulator after writing the code.

1) Simulated Output of Array Multiplier

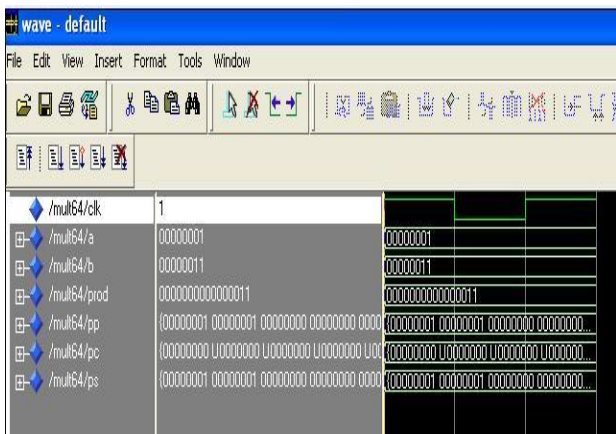


Figure: 4.1 Simulation of Array Multiplier

Table: 4.1 Various identifiers used in the simulation of Array Multiplier

IDENTIFIER	MODE	TYPE	NO OF BITS
A	In	Std_logic_vector	8
B	In	Std_logic_vector	8
Prod	Out	Std_logic_vector	16

Description:

As the Array Multiplier can handle only unsigned bits. From the simulated wave form it is observed as below

Table: 4.2 Inputs and Outputs of the Array Multiplier

INPUT1(a)	INPUT2(b)	OUTPUT(prod)
00000001	00000011	0000000000000011

2) Simulated Output of Radix-2 Booth Multiplier

Booth Multiplier can handle both signed and unsigned numbers. This algorithm is efficient in terms of reducing time delay and power consumption. The reduction of time delay and power consumption is possible only by reducing the partial products. Finally it has been proved to be more efficient when compared to all the multipliers in terms of time delay and power consumption.

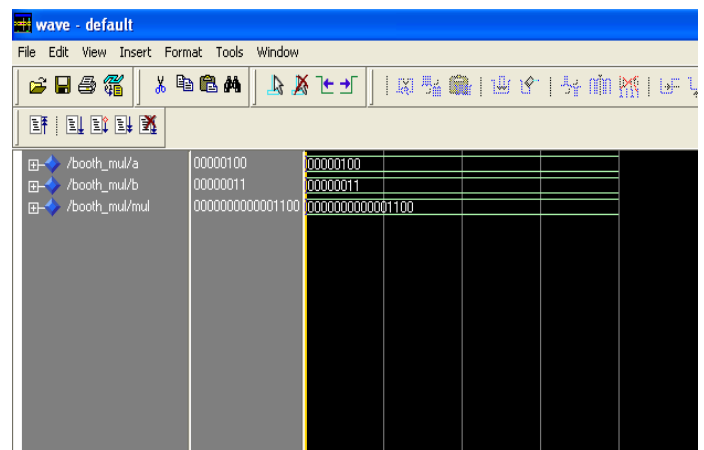


Figure: 4.2 Simulation Of Radix-2 Booth Multiplier

Table 4.3 Various Identifiers Used In the Radix-2 Booth Multiplier

Simulation

IDENTIFIER	MODE	TYPE	NO OF BITS
A	In	Std_logic_vector	8
B	In	Std_logic_vector	8
Mul	Out	Std_logic_vector	16

Description

Booth Multiplier can handle both signed and unsigned numbers. From the simulated wave form it is observed.

Table: 4.4 Inputs And Outputs Applied To The Radix-2 Booth Multiplier

INPUT1(a)	INPUT2(b)	OUTPUT(mul)
00000100	00000011	00000000000001100

3)Simulated Output of Radix-4 Booth Multiplier

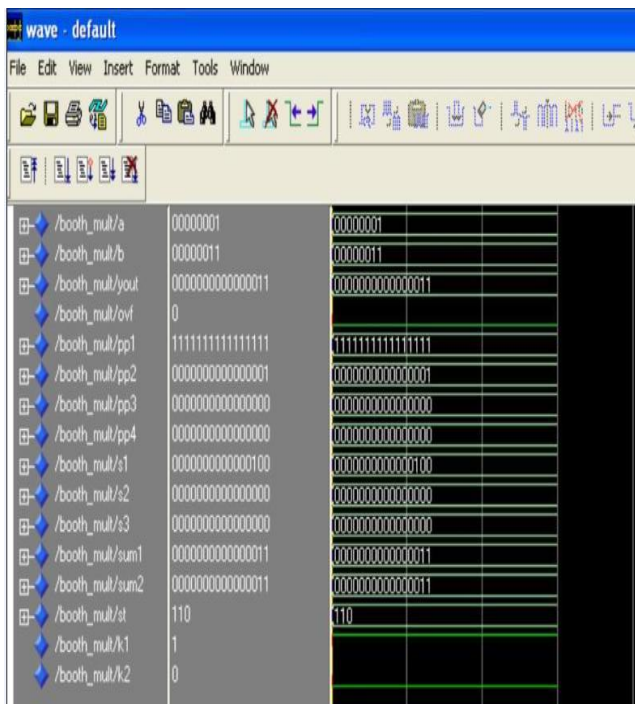


Figure: 4.3 Simulation Of Radix-4 Booth Multiplier

Table 4.5 Various Identifiers Used In The Radix-4 Booth Multiplier Simulation

IDENTIFIER	MODE	TYPE	NO OF BITS
A	In	Std_logic_vector	8
B	In	Std_logic_vector	8
Yout	Out	Std_logic_vector	16

Description

Booth Multiplier can handle both signed and unsigned numbers. From the simulated wave form it is observed.

Table: 4.6 Inputs And Outputs Applied To The Radix-4 Booth Multiplier

INPUT1(a)	INPUT2(b)	OUTPUT(yout)
00000001	00000011	0000000000000011

The MSB bit represents the signed bit. If it is '1', it represents the negative number and if it is '0', it represents the positive number. It is mainly meant to reduce the time delay by reducing the partial products and it is mainly advantageous when there are group of 1's and finally it is found to be efficient in terms of power consumption.

B) ANALYSIS REPORT

The Timing report of the multiplier is generated by synthesizing the given code, After synthesizing the given code, we have to target this code on to the Field Programmable Gate logic Array (FPGA).Initially, the

package pins are assigned to each port and after that we have to configure the devices in Slave/Serial mode and finally we have to dump the code in the FPGA and verify the outputs.



Figure: 4.4 Design Summary of Array Multiplier

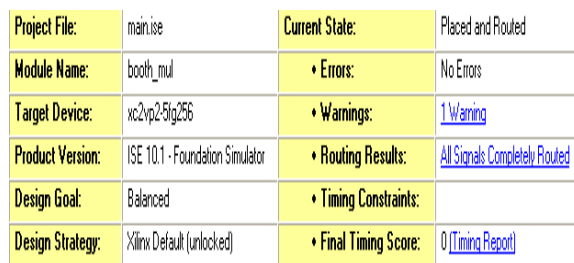


Figure: 4.5 Design Summary of Radix-2 Booth Multiplier

Project File:	main.isc	Current State:	Placed and Routed
Module Name:	booth_mult	• Errors:	No Errors
Target Device:	xc2vp2-5fg256	• Warnings:	8 Warnings
Product Version:	ISE 10.1 - Foundation Simulator	• Routing Results:	All Signals Completely Routed
Design Goal:	Balanced	• Timing Constraints:	
Design Strategy:	Xilinx Default (unlocked)	• Final Timing Score:	0 (Timing Report)

main Partition Summary		[E]
No partition information was found.		

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	178	2,816	6%	
Logic Distribution				
Number of occupied Slices	96	1,408	6%	
Number of Slices containing only related logic	96	96	100%	
Number of Slices containing unrelated logic	0	96	0%	
Total Number of 4 input LUTs	178	2,816	6%	
Number of bonded IOBs	33	140	23%	

Figure: 4.6 Design Summary of Radix-4 Booth Multiplier

C) RTL AND TECHNOLOGY SCHEMATICS

RTL VIEW

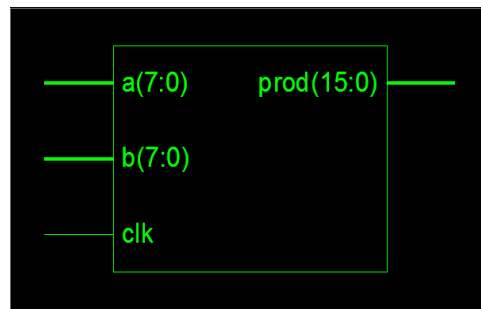
RTL View is a Register Transfer Level graphical representation of your design. This representation (.ngr file produced by Xilinx Synthesis Technology (XST)) is generated by the synthesis tool at earlier stages of a synthesis process when technology mapping is not yet completed. The goal of this view is to be as close as possible to the original HDL code. In the RTL view, the design is represented in terms of macro blocks, such as adders, multipliers, and registers. Standard combinatorial logic is mapped onto logic gates, such as AND, NAND, and OR.

Technology schematic:

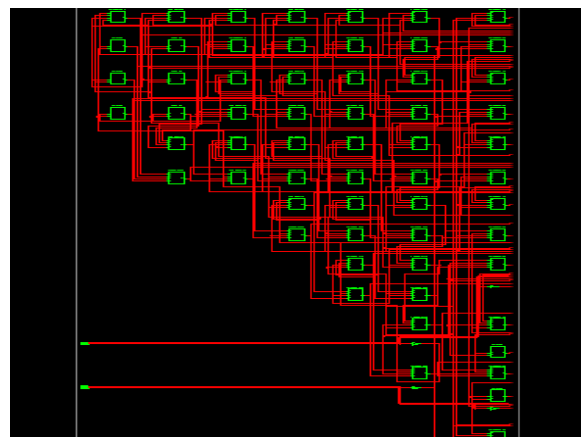
This schematic shows the representation of design in terms of logic elements optimized to the target Xilinx device or “technology”, for example, in terms of LUTs, carry logic, IOB’S and other technology specific components. Viewing this schematic allows you to see a technology level representation of your HDL optimized for a specific Xilinx architecture, which may help you discover design issues early in the design process.

1). ARRAY MULTIPLIER

1.1. RTL SCHEMATIC

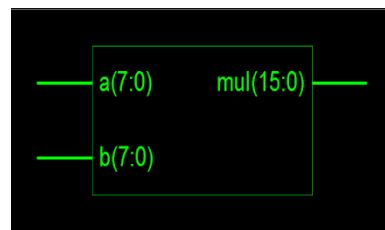


1.2. TECHNOLOGY SCHEMATIC

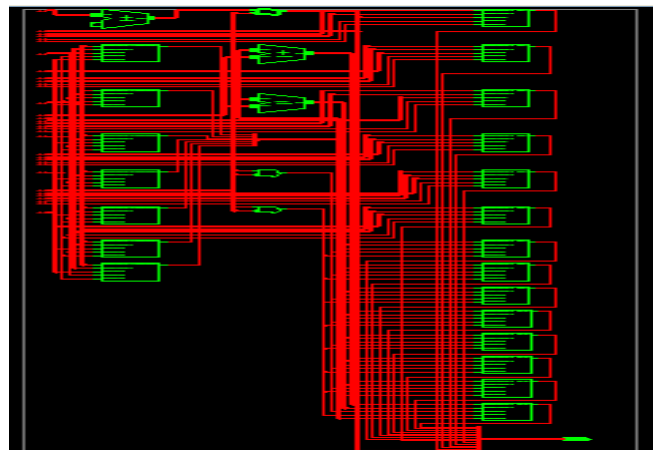


2.) BOOTH MULTIPLIER (RADIX-2)

2.1. RTL SCHEMATIC

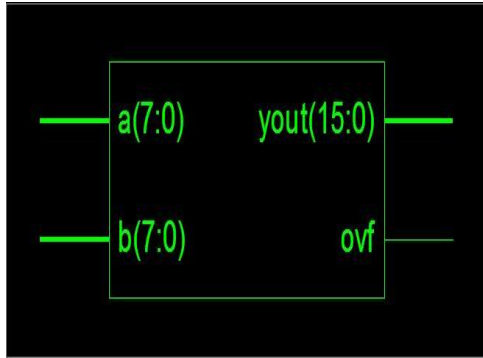


2.2. TECHNOLOGY SCHEMATIC

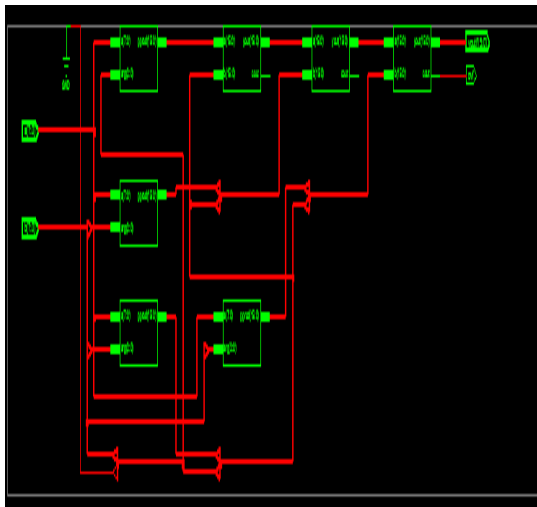


5.3.) BOOTH MULTIPLIER (RADIX-4)

3.1. RTL SCHEMATIC



3.2. TECHNOLOGY SCHEMATIC



V. COMPARISON OF MULTIPLIERS

A. Time Delay Analysis of Various Multipliers

S.No	Multiplier	Time Delay
1	Array Multiplier	15.529 ns
2	Radix-2 Booth Multiplier	14.338 ns
3	Radix-4 Booth Multiplier	30.078 ns

B. Performance Comparison of Various Multipliers

Specifications	Array multiplier	Radix-2 booth multiplier	Radix-4 booth multiplier
No. of slices	4%	8%	6%
No. of LUTs	4%	8%	6%
No. of bonded IOBs	22%	22%	23%

VI. FPGA IMPLEMENTATION

We have used Spartan-3 FPGA kit to show the outputs of the three multipliers, Array, Booth Radix-2 and Radix-4. The following are the results obtained for 2-bit multiplication on Spartan-3.

A. Array Multiplier

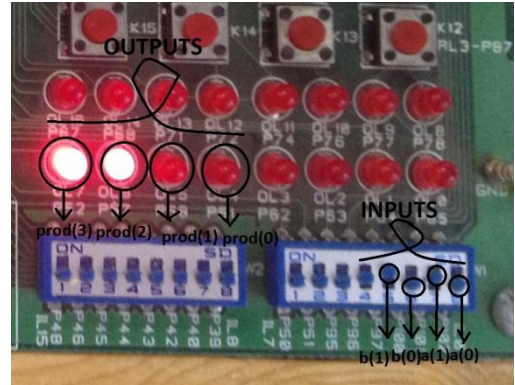


Figure 6.1. Array Multiplier output

B. Booth Radix-2 Multiplier:

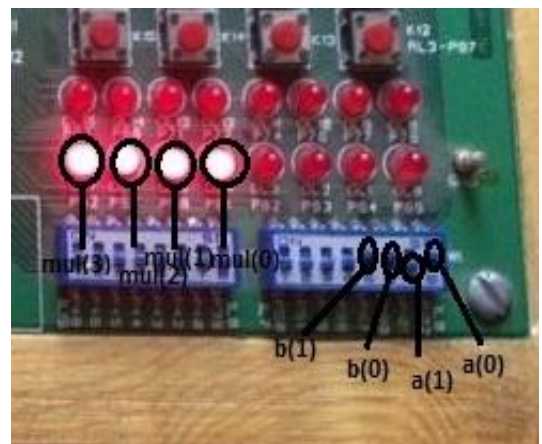


Figure 6.2. Booth Radix-2 Multiplier output

C. Booth radix-4 Multiplier:

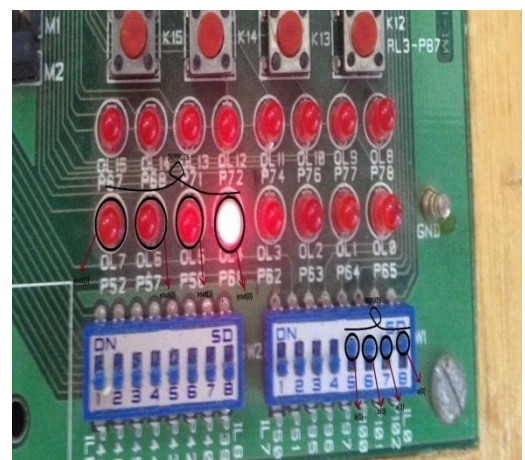


Figure 6.3. Booth Radix-4 Multiplier output

VII. CONCLUSION

This paper attempts to give a clear concept of different multipliers and their comparison. We found that the parallel multipliers are better than the serial multiplier. We concluded this from the result of power consumption and the total area. In case of parallel multipliers, the total area is much less than that of serial multipliers. Hence the power consumption is also less. This is clearly depicted in our results. This speeds up the calculation and makes the system faster. While comparing the radix 2 and the radix 4 booth multipliers we found that radix 4 consumes lesser power than that of radix 2. This is because it uses almost half number of iteration and adders when compared to radix 2. When all the three multipliers were compared we found that array multipliers are most power consuming. This is because it uses a large number of adders. As a result it slows down the system because the system has to perform more number of calculations in case of array multiplier. Multipliers are one of the most important components of many systems. So we always need to find a better solution in case of multipliers. Preferably multipliers should always consume less power and cover less area. So through our project we try to determine which of the three algorithms works the best. In the end we determine that radix 4 modified booth algorithm works the best. Also the implementation using FPGA and the time delay obtained concludes that same as stated above.

ACKNOWLEDGMENT

I would like to express my gratitude to the following people for their support and guidance for the success of this paper. I express my deep sense of gratitude to Dr.L.Padmasree, Professor, ECE, VNRVJIET, Hyderabad, India. I am deeply indebted to Dr.C.D Naidu, Principal, VNRVJIET, Hyderabad, India

REFERENCES

- [1] "Modeling of Time Delay in VHDL-based Design of a Multiplier" Z. A. bid Electrical Engineering Department Pobox 800, King Saud University, Riyadh 11421.
- [2] "A Two's Complement Parallel Array multiplication Algorithm" By Charles.R.Baugh and Bruce.A.Wooley.
- [3] Computer System Architecture by "Marris Mano".
- [4] M. Miyamoto et al., "High-Speed and Low-Power Interconnect Technology for Sub-Quarter-Micron ASIC's", IEEE Transaction on Electron Devices, V. 44, No 2, 250, 1997.
- [5] A. Bellaouar and M. I. Elmasry, "Low-Power Digital VLSI Design", Kluwer Academic Publishers, 1995..
- [6] "Low voltage, low power VLSI subsystems" By Kiat Seng Yeo, Kaushik Roy
- [7] "Digital Design" By John.F.Wakerly.
- [8] "VHDL Premier and VHDL Synthesis" By V.J.Prasad.
- [9] Marc Hunger and Daniel Marienfeld, "New self checking booth multiplier", Int J. Appl. Math, Comput. Sci., Vol.18, No.3, 319-328, 2008
- [10] Fayed and M. Bayoumi, "A merged multiplier-accumulator for high speed signal processing application", Proc. ICASSP, vol. 3, pp.3212-3215, 2002
- [11] "FPGA implementation of different architectures" by Laxman S, Darshan, Dr Chirag Sharma., IJATAE, Volume2, Issue6.

B Naga Venkata Satya Durga Jahnvi was born in 1991. She did her Bachelors in Electronics and Communication Engineering from VNR Vignana Jyothi Institute of Engineering and Technology affiliated to JNTU, Hyderabad, India. Her interests include VLSI, Image Processing and Embedded Systems.

Shivani Mupparaju was born in 1992. She did her Bachelors in Electronics and Communication Engineering from VNR Vignana Jyothi Institute of Engineering and Technology affiliated to JNTU, Hyderabad, India. Her interests include VLSI, Image Processing and Embedded Systems.

Dr.L.Padmasree was born in 1966. She did her B.Tech from Bapatla Engineering College, India. She received PhD in 2010 from JNTU, India. She is working as a professor in Electronics and Communication Engineering department in VNR Vignana Jyothi Institute of Engineering and Technology affiliated to JNTU, Hyderabad, India. Her research areas include Image Processing, VLSI, Digital Systems and Neural Networks.