PERFORMANCE AND POWER COMPARISON OF GENERIC MULTIPLIERS IMPLEMENTATION IN FPGA

Mr. S Vasu Krishna,
Associate Professor,
Department of ECE,
Geethanjali College of Engineering and Technology,
Hyderabad.

Dr. K Lal Kishore,
Professor & Dean, R & D,
Department of ECE,
CVR College of Engineering,
Hyderabad.

Abstract - A trade-off between low power consumption and the performance are most important criteria for DSP systems and portable/handheld systems. A typical central processing unit devotes a considerable amount of processing time in performing arithmetic operations, particularly multiplication operations. Multipliers are one of the widely used component in signal processing applications. This paper aims at an efficient implementation of high speed multiplier using the Shift and Add method, Radix-2, Radix-4 modified Booth multiplier algorithm .The low power consumption of booth multiplier makes it a preferred choice in designing different circuits with clock synchronization and clock gating. The proposed Synchronous Dynamic Clock Controlled booth multiplier makes it a preferred choice in designing different circuits to achieve low power. The result of this paper helps to choose a better option between serial and parallel multiplier in different systems. In this paper a comparison is made with the three multipliers by implementing each of them by targeting to a FPGA.

Index Terms: Multipliers, Radix-2, Radix-4 Booth Multiplier, DSP Processors

I. INTRODUCTION

FPGA is one of the most used ICs in low volume embedded system sector. The power consumption of any logic inside the FPGA chip can be classified as static and dynamic. The actual effort of the circuit to switch is the dominant component. The dynamic power consumption of CMOS circuitry is as shown below.

$$P = F*C*V^2$$

Where P is the power, C is the effective capacitance of the logic, V is the supply voltage, and F is the frequency of switching operation . The clock signal is a main component of all synchronous blocks, and since it switches every cycle, it has an activity factor. The dominant power in FPGAs is dynamic power, which is consumed by this clock network. But clocked circuits are preferred over combinational designs because of various advantages like stability, robustness, reliability and user defined clocks to satisfy various applications. Clock gating has been heavily used in reducing the power consumption of the clock network by limiting its activity factor. Fundamentally, clock gating reduces the dynamic power dissipation by disconnecting the clock from an unused circuit block. Traditionally, the system clock is connected to the clock pin on every flip-flop in the design. This results in three major components of power consumption. One is Power consumed by combinatorial logic whose values are changing on each event on inputs or noise. Second is Power consumed by flip-flops – this has a non-zero value even if the flip-flops inputs are not varying, and the state of the flip-flops is constant. Third is Power consumed by the clock buffer tree in the design. Clock gating has the potential of reducing both the power consumed by flip-flops and the power consumed by the clock distribution network. The synchronous circuits along with clock gating can save significant power, typically reducing switching activity by 15–25%.

As integrated circuit technology has improved, to allow more and more components on a chip, digital systems have continued to grow in complexity. As digital systems have become more complex, detailed design of the systems at the gate and flip-flop level has become very tedious and time consuming. For this reason, the use of hardware description languages in the digital design process continues to grow in importance. A hardware description language allows a digital system to be designed and debugged at a higher level before implementation at the gate and flip-flop level. The use of computer-aided design tools to do this conversion is becoming more wide-spread. This is analogous to writing software programs in a high-level language such as C and then using a compiler to convert the programs to machine language. Verilog is one such language, which is used for implementing multiplier in this paper. After the coding, they are targeted on the FPGA and by using Xilinx Synthesis Report, the performance of these multipliers has been observed and power consumption is observed using power analyzer in Xilinx 14.2 package.

II. SIMULATION IN TERMS OF TIME AND POWER CONSTRAINTS

Multipliers are the important fundamental blocks in arithmetic operations. In fact, multiplication based

operations such as Multiply and Accumulate (MAC) and inner product are among some of the frequently used computations-intensive arithmetic functions currently implemented in many Digital Signal Processors (DSP) applications such as convolution, Fast Fourier Transform (FFT), filtering and others.

Two ways to speed up the multiplication is either by reducing number of partial products or by accelerating accumulation. The three types of high-speed multipliers are Array multiplier, Radix-2 Booth multiplier and Radix-4. In this work, 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 with clock and clock enable is optimized in terms of power consumption.

A. Methods and Performance

In all the practical systems such as FIR filters, microprocessors, digital signal processors, etc the multipliers are key components to achieve high performance. A system's performance is generally determined by the performance of the multiplier because the multiplier is generally the slowest element in the system. Furthermore, it is generally the most area consuming. Hence, optimizing the speed and area of the multiplier is a major design issue. However, the conflicting constraints are area and speed, so that improving speed results mostly in larger areas.

As a result, a whole spectrum of multipliers with different area-speed constraints has been designed with fully parallel multipliers at one end of the spectrum and fully serial multipliers at the other end. In between are digit serial multipliers where single digits consisting of several bits are operated on. These multipliers have moderate performance in both speed and area. Multiplication is implemented by adding a series of shifted multiplicands based on the number of digits of the multiplier.

B. 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. If the multiplier digit is 1 the multiplicand is simply copied down and represents the product. 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 .It should be capable of identifying whether a bit is 0 or a 1. It should be capable of shifting left partial products, it should also be able to add all the partial products to give the products as sum of partial products.

It should examine the sign bits. If they are alike, the sign of the product will be 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 the observation has been done 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.

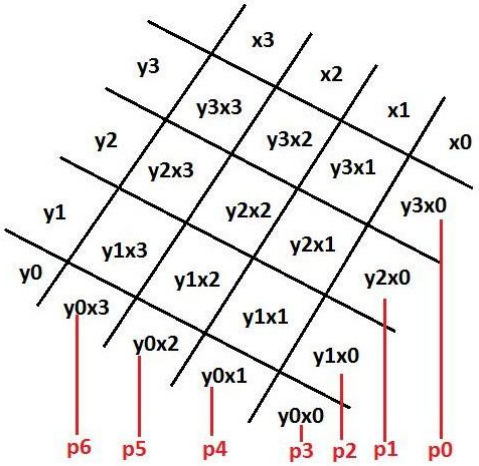

Figure 1: Array Multiplier Architecture

C. Booth Multiplication Algorithm

Booth algorithm gives a procedure for multiplying binary integers in signed 2's complement representation. The booth algorithm is illustrated with the following example.

D. Making the Booth Table

From the two numbers, pick the number with the smallest difference between a series of consecutive numbers, and make it a multiplier .i.e., In 0010, from 0 to 0 no change, 0 to 1 one change, 1 to 0 another change, so there are two changes in this one. In 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. Steps for multiplication of 2 x (– 4) : I. $(2)_{10}$ i.e $(0010)_2$ is the multiplicand and $(-4)_{10}$ i.e $(1100)_2$ 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. I. 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 1: Step1 in Radix-2 Booth Multiplication

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

III. 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 from using two registers for the X value. VII. Repeat the steps until four cycles are completed.

Table 2: Step 3 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 3: Step 4 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

Before designing a MBE (Modified Booth Encoder), 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 later. Prior to converting 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: 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 & shift	Beginning of string of ones	1
0	1	Add & shift	End of string of ones	1

Table 4. 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).

After finishing four cycles, the result is shown, in the last rows of U and V which is: $(11111000)_2$. 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. Recoding multiplier is $x_{n-1}\ x_{n-2}...x_1\ x_0$ in SD (Signed Digit) code. Recoded multiplicand is $y_{n-1}\ y_{n-2}\ ...\ y_1\ y_0$. x_i, x_{i-1} of multiplier is examined to generate y_i. Simple recoding is $y_i = x_{i-1} - x_i$. Drawback of radix-2 booth multiplier is a problem of isolated one's. To overcome this drawback, instead of comparing two bits at a time, the comparison of three bits at a time should be done.

IV. DYNAMIC CLOCK MODIFIED BOOTH ENCODER

Dynamic clock controlled Modified Booth encoding can be used to have variable low power factors and avoid variable size partial product arrays. A clock frequency of 50MHz is chosen for calculation. Modified Booth encoding is most often used to avoid variable size partial product arrays.

Table 5: 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. The truth table shows the function of booth encoder.

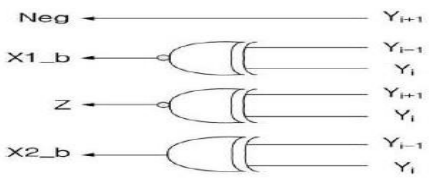

Figure 2: Booth Encoder

If a 3-bit binary input sequence is given at the input, and the operation performed as mentioned, the partial products will be generated.

V. SIMULATION RESULTS

The different multipliers are simulated using Xilinx 14.2 ISE simulator after writing the code.

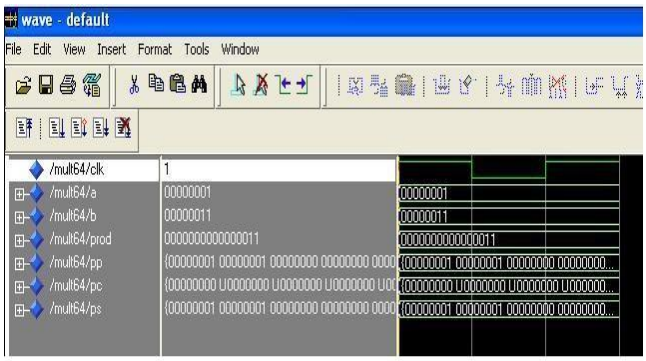

Figure 3: Simulation of Array Multiplier

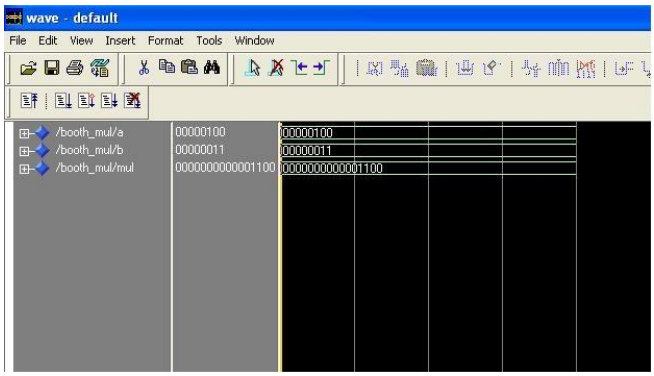

Figure 4: Simulation Of Radix-2 Booth Multiplier

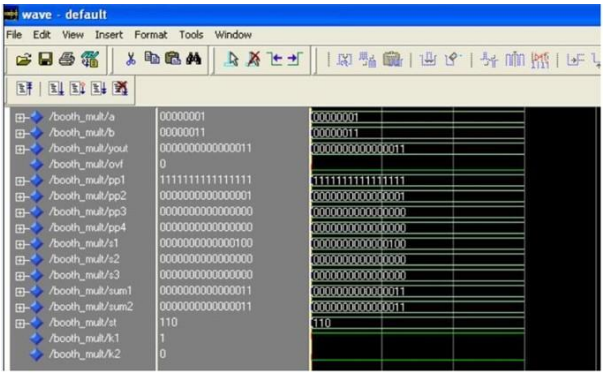

Figure 5: Simulation of Radix-4 booth multiplier

The timing report of the multiplier is generated by doing synthesis of code and is targeted on to the Field Programmable Gate logic Array (FPGA). Initially, the package pins are assigned to each port and after that the devices have to be configured in Slave/Serial mode and finally the code is dumped into the FPGA to verify the outputs.

main Project Status			
Project File:	main.ise	Current State:	Placed and Routed
Module Name:	mult64	• Errors:	No Errors
Target Device:	xc6vlx75t-2ff784	• Warnings:	13 Warnings
Product Version:	ISE 14.2 - 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
No partition information was found.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	123	2,816	4%	
Logic Distribution				
Number of occupied Slices	64	1,408	4%	
Number of Slices containing only related logic	64	64	100%	
Number of Slices containing unrelated logic	0	64	0%	
Total Number of 4 input LUTs	123	2,816	4%	
Number of bonded IOBs	32	140	22%	

Figure 6: Design Summary of Array Multiplier

Project File:	main.ise	Current State:	Placed and Routed
Module Name:	booth_mul	• Errors:	No Errors
Target Device:	xc6vlx75t-2ff784	• Warnings:	1 Warning
Product Version:	ISE 14.2 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
No partition information was found.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s)
Number of 4 input LUTs	234	2,816	8%	
Logic Distribution				
Number of occupied Slices	117	1,408	8%	
Number of Slices containing only related logic	117	117	100%	
Number of Slices containing unrelated logic	0	117	0%	
Total Number of 4 input LUTs	234	2,816	8%	
Number of bonded IOBs	32	140	22%	

Figure 7: Design Summary of Radix-2 Booth Multiplier

Project File:	main.ise	Current State:	Placed and Routed
Module Name:	booth_mult	• Errors:	No Errors
Target Device:	xc6vlx75t-2ff784	• Warnings:	8 Warnings
Product Version:	ISE 14.2 - 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
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 8: Design Summary of Radix-4 Booth Multiplier.

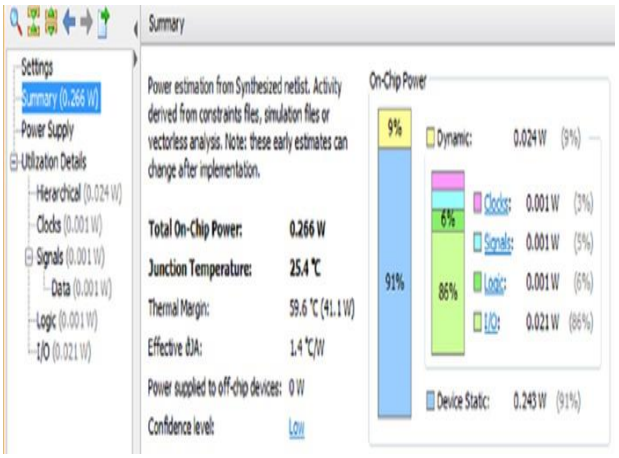

Figure 9: FPGA Power results

The above result states that the proposed 4x4 Dynamic Clock controlled Synchronous Booth Multiplier consumes only 1mW logic power, 1mW clock power and 1mW signal, a total of 3mW compared to generic Booth multiplier. Comparison table of different multipliers are shown below.

Table 6: 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	20.078 ns

Table 7: Performance and Power Comparison of Various Multipliers

Specifications	Array Multiplier	Radix-2 Booth Multiplier	Radix-4 Booth Multiplier	Proposed RADIX-4 BOOTH Multiplier
No. of Slices	4%	8%	6%	7%
No. of LUTs	4%	8%	6%	7%
No. of bonded IOBs	22%	22%	23%	23%
Power	104 mW	79 mW	47 mW	3mW
Performance	66 MHz	71 MHz	50 MHz	50MHz

VI. CONCLUSION

This paper gives a clear concept of different multipliers, their comparison and a proposal of Dynamic Clock controlled Synchronous Booth Multiplier to achieve low power. It can be concluded that the parallel multipliers are better than the serial multiplier based on the result of power consumption and Performance. In case of parallel multipliers, the total area is much less than that of serial multipliers, hence the power consumption is also less. It also speeds up the calculation and makes the system faster. While comparing the radix 2 and the radix 4 booth multipliers it has been found that radix 4 consumes less power than that of radix 2. This is because it uses almost half number of iterations and adders when compared to radix 2.When all the three multipliers were compared, it has been 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. The proposed Dynamic Clock controlled Synchronous Booth Multiplier can be a potential choice for variable applications.

REFERENCES

[1] C. F. Law, S. S. Rofail , and K. S. Yeo "A Low-Power 16×16-Bit Parallel Multiplier Utilizing Pass-Transistor Logic," IEEE Journal of Solid State circuits, Vol.34, No.10, pp. 1395-1399,Oct. 1999.

[2] Jong Duk Lee, Yong Jin Yoony, Kyong Hwa Leez and Byung-Gook Park "Application of Dynamic Pass Transistor Logic to 8-Bit Multiplier," Journal of the Korean Physical Society, Vol.38, No. 3, pp.220-223,March, 2001.

[3] Oscal T. C. Chen, Sandy Wang, and Yi-Wen Wu "Minimization of Switching Activities of Partial Products for Designing Low-Power Multipliers," IEEE Transaction on VLSI System.Vol.11, No.3, pp. 418-433, June ,2003.

[4] Pouya Asadi and Keivan Navi, "A New Low Power 32×32-bit Multiplier," World Applied Sciences Journal IDOSI Publication,Vol.2, No.4,pp.341-347, 2007.

[5] C.N. Marimuthu and P. Thangaraj, "Low power high performance multiplier," Proc. ICGST-PDCS, Vol.8, pp.31–38, Dec. 2008.

[6] Young-Ho Seo and Dong-Wook Kim, "A new VLSI architecture of parallel multiplier-accumulator based on radix-2 modified Booth algorithm," in IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol.18, No.2, pp.201-208, Feb. 2010.

[7] Ning Zhu, Wang Ling Goh, Weija Zhang, Kiat Seng Yeo, and Zhi Hui Kong, "Design of Low-Power High-Speed Truncation-Error-Tolerant Adder and Its Application in Digital Signal Processing," IEEE Transactions on VLSI Systems, Vol. 18, No. 8, August ,2010.

[8] Padma Devi, Ashima Girdher and Balwinder Singh, "Improved Carry Select Adder with Reduced Area and Low Power Consumption," International Journal of Computer Application ,Vol 3.No.4, June ,2010 .

[9] B.Ramkumar, Harish M Kittur, P.Mahesh Kannan, "ASIC Implementation of Modified Faster Carry Save Adder", European Journal of Scientific Research Vol.42, No.1, pp.53-58,2010.

[10] Mariano Aguirre-Hernandez and Monico Linares-Aranda, "CMOS Full-Adders for Energy-Efficient Arithmetic

Applications," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.19, No.4, April ,2011.
[11] Y. Sunil Gavaskar Reddy and V.V.G.S. Rajendra Prasad, "Power Comparison of CMOS and Adiabatic Full Adder Circuits", International Journal of VLSI design & Communication Systems (VLSICS) Vol.2, No.3 ,Sept. 2011.
[12] R. Uma, "4-Bit Fast Adder Design Topology and Layout with Self-Resetting Logic for Low Power VLSI Circuits,", International Journal of Advanced Engineering Sciences and Technology, Vol .7, No. 2,pp. 197-205,2011.
[13] Rahul Shrestha and Utkarsh Rastogim , "Design and Implementation of Area-Efficient and Low-Power Configurable Booth-Multiplier," 2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems.