International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

Performance Evaluation of Different Digital Multipliers in VLSI using VHDL

Bhawna Singroul
M. Tech Student
Department of Electronics & Communication Engineering
Dr. C.V. Raman University Kota, Bilaspur
C.G., India

Pallavee Jaiswal
Assistant Professor
Department of Electronics & Communication Engineering
Dr. C.V. Raman University Kota, Bilaspur
C.G., India

***Abstract*-This project presents an efficient implementation of high speed multiplier using the shift and add method, Radix 2, Radix 4 modified Booth multiplier algorithm by comparing the working of the three multipliers by implementing each of them separately in FIR filter. The parallel multipliers like radix 2 and radix 4 modified booth multiplier does the computations using lesser adders and lesser iterative steps. As a result of which they occupy lesser space as compared to the serial multiplier. We first designed three different type of multipliers using shift and add method, radix 2 and radix 4 modified booth multiplier algorithm. We used different type of adders like sixteen bit full adder in designing that multiplier. Then we designed a 4 tap delay FIR filter and in place of the multiplication and additions we implemented the components of different multipliers and adders. Then we compared the working of different multipliers by comparing the power consumption by each of them.So by analyzing the working of different multipliers helps to frame a better system with less power consumption and lesser area.**

Keywords: Digital multipliers, comparison, delay, power, area, vedic, wallce, booth.

1 INTRODUCTION

Multipliers are key components of many high performance systems such as FIR filters, microprocessors, digital signal processors, etc. 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, area and speed are usually conflicting constraints so that improving speed results mostly in larger areas. As a result, whole spectrums of multipliers with different area-speed constraints have 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. However, existing digit serial multipliers have been Plagued by complicated switching systems and/or irregularities in design. Radix 2^n multipliers which operate on digits in a parallel fashion instead of bits bring the pipelining to the digit level and avoid most of the above problems. They were introduced by M. K. Ibrahim in 1993. These structures are iterative and modular. The pipelining done at the digit level brings the benefit of constant operation speed irrespective of the size of the multiplier. The clock speed is only determined by the digit size which is already fixed before the design is implemented.

2 METHODOLOGY

2.1 Overview

In this project we first designed three different types of multipliers using shift and add method, radix 2 and radix 4 modified booth multiplier algorithm. We used different type of adders like sixteen bit full adder in designing that multiplier. Then we designed a 4 tap delay FIR filter and in place of the multiplication and additions we implemented the components of different multipliers and adders. Then we compared the working of different multipliers by comparing the power consumption by each of them.

2.2 VHDL: The language

An entity declaration, or entity, combined with architecture or body constitutes a VHDL model. VHDL calls the entity-architecture pair a design entity. By describing alternative architectures for an entity, we can configure a VHDL model for a specific level of investigation. The entity contains the interface description common to the alternative architectures. It communicates with other entities and the environment through ports and generics. Generic information particularizes an entity by specifying environment constants such as register size or delay value
Power consumption in VLSI DSPs has gained special attention due to the proliferation of high-performance portable battery-powered electronic devices such as cellular phones, laptop computers, etc. DSP applications require high computational speed and, at the same time, suffer from stringent power dissipation constraints. Multiplier modules are common to many DSP applications. The fastest types of multipliers are parallel multipliers. Among these, the Wallace multiplier is among the fastest. However, they suffer from a bad regularity. Hence, when regularity, high performance and low power are primary concerns, Booth multipliers tend to be the primary choice. Booth multipliers allow the operation on signed operands in 2's complement. They derive from array multipliers where, for each bit in a partial product line, an encoding scheme is used to determine if this bit is positive, negative or zero. The Modified Booth algorithm achieves a major performance improvement through radix-4 encoding. In this algorithm each partial product line operates on 2 bits at a time, thereby reducing the total number of the partial

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

products. This is particularly true for operands using 16 bits or more.

2.3 Shift and add multiplier

Shift-and-add multiplication is similar to the multiplication performed by paper and pencil. This method adds the multiplicand X to itself Y times, where Y denotes the multiplier. To multiply two numbers by paper and pencil, the algorithm is to take the digits of the multiplier one at a time from right to left, multiplying the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

2.4 Booth multiplier (Radix2)

Booth's algorithm is based upon recoding the multiplier, y, to a recoded, value, z, leaving the multiplicand, x, unchanged. In Booth recoding, each digit of the multiplier can assume negative as well as positive and zero values. There is a special notation, called signed digit (SD) encoding, to express these signed digits. In SD encoding +1 and 0 are expressed as 1and 0, but -1 is expressed as 1

The value of a 2s complement integer was defined a by equation.

This equation says that in order to get the value of a signed 2's complement number, multiply the m – ith digit by -2`-1, and multiply each remaining digit i by +2g.

y_i	y_{i-1}	z_{i-i}	Multiplier Value	Situation
0	0	0	0	String of 0s
0	1	1	+1	End of string of 1s
1	0	1	-1	Begin string of 1s
1	1	0	0	String of 1s

Table 2.1: Booth recoding table for adix_2.

2.5 Booth Multiplication Algorithm for radix 2

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 two* 1100 two

Step 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.e., 0010 -- From 0 to 0 no change, 0 to 1 one change, 1 to 0 another change, and so there are two changes on this one

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. Therefore, multiplication of 2 x (– 4), where 2 ten (0010 two) is the multiplicand and (– 4) ten (1100 two) 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.

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

Table 2.2: Making the Booth table

Step 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

1 1 Shift only.

0 1 Add Y to U, and shift

1 0 Subtract Y from U, and shift or add (-Y) to U and shift

II 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.

III Shift X circular right shift because this will prevent us from using two registers for the X value.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

2.6 Modified booth multiplier (Radix 4)

The Booth multiplier makes use of Booth encoding algorithm in order to reduce the number of partial products by processing three bits at a time during recoding. This recoding method is widely used to generate the partial products for implementation of large parallel multipliers, which adopts the parallel encoding scheme.

2.7 Booth multiplication algorithm for radix 4

One of the solutions of realizing high speed multipliers is to enhance parallelism which helps to decrease the number of subsequent calculation stages. The original version of the Booth algorithm (Radix-2) had two drawbacks. They are: (i) the number of add subtract operations and the number of shift operations become variable and become inconvenient in designing parallel multipliers. (ii) The algorithm becomes inefficient when there are isolated 1's. These problems are overcome by using modified Radix4 Booth algorithm which scans strings of three bits with the algorithm given below:

1) Extend the sign bit 1 position if necessary to ensure that n is even.

2) Append a 0 to the right of the LSB of the multiplier.

3) According to the value of each vector, each Partial Product will he 0, +y , -y, +2y or -2y.

The negative values of y are made by taking the 2's complement and in this paper Carry-look-ahead (CLA) fast adders are used. The multiplication of y is done by shifting y by one bit to the left. Thus, in any case, in designing a n-bit parallel multipliers, only n/2 partial products are generated.

X(i)	X(i-1)	X(i-2)	Y
0	0	0	+0
0	0	1	+y
0	1	0	+y
0	1	1	+2y
1	0	0	-2y
1	0	1	-y
1	1	0	-y
1	1	1	+0

Table 2.3: Radix4 Modified Booth algorithm scheme for odd values of i.

3 RESULTS OF DIFFERENT MULTIPLIERS

After analysing the three multipliers, and compare their characteristics in terms of multiplication speed, no of computations required, no of hardware, we come on finding that radix 4 booth multipliers is better than Array multiplier and Radix-2 booth multipliers. By implementing Array, Radix-2 & Radix -4 multiplier we analysis that Radix -4 multiplier computation speed is higher than Array and Radix-2 multipliers . We have done the coding of all multipliers separately in VHDL & simulate it to get the accurate waveforms as output each multiplier shown in Figure. Also get the device utilization summary, where we get the exact no of i/p, o/p/ no of slices requirement etc for the multiplier. In Radix-4 design simulation result is same as Radix-2 scheme. Only difference between these two schemes is synthesis report. These results are given in tables. Comparison of all three multipliers are shown in graph.

3.1 ARRAY MULTIPLIER

Number of Slices	229
Number of 4 input LUTs	302
Number of bonded INPUT	16
Number of bonded OUTPUT	16
CLB Logic Power	104mW

Table3.1: Output of Array multiplier

3.2 RADIX 2 BOOTH MULTIPLIER

Number of Slices	130
Number of 4 input LUTs	249
Number of bonded INPUT	16
Number of bonded OUTPUT	17
CLB Logic Power	79mW

Table3.2: Output of Radix 2 booth multiplier

3.3 RADIX 4 BOOTH MULTIPLIER

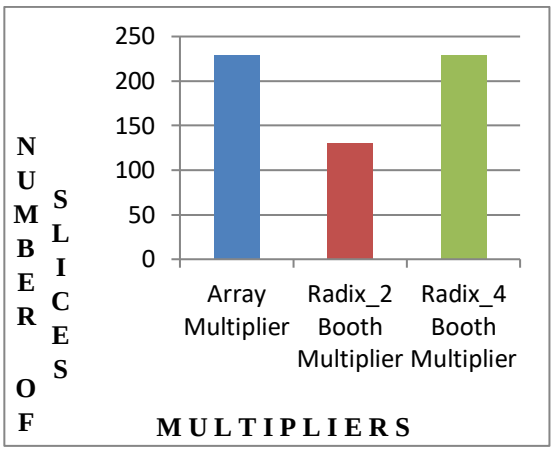
Number of Slices	229
Number of 4 input LUTs	302
Number of bonded INPUT	16
Number of bonded OUTPUT	16
CLB Logic Power	47mW

Table3.3 : Output of Radix 4 booth multiplier

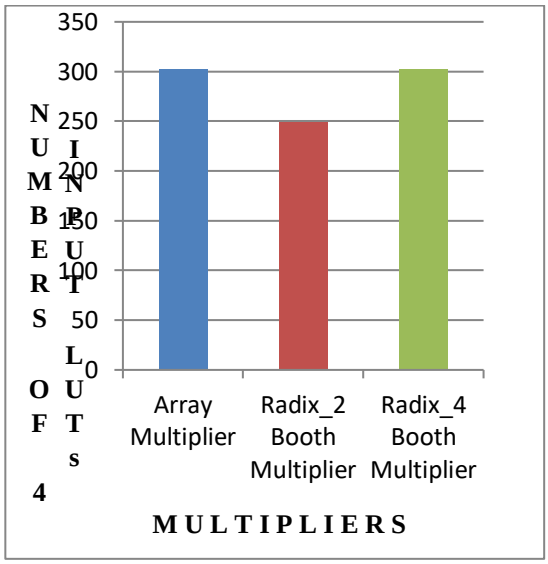
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

3.4 COMPARISION OF MULTIPLIERS CHARACTERISTICS

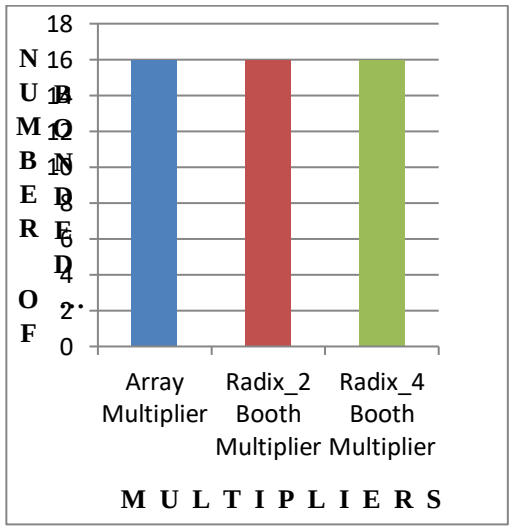
3.4..1 NUMBER OF SLICES



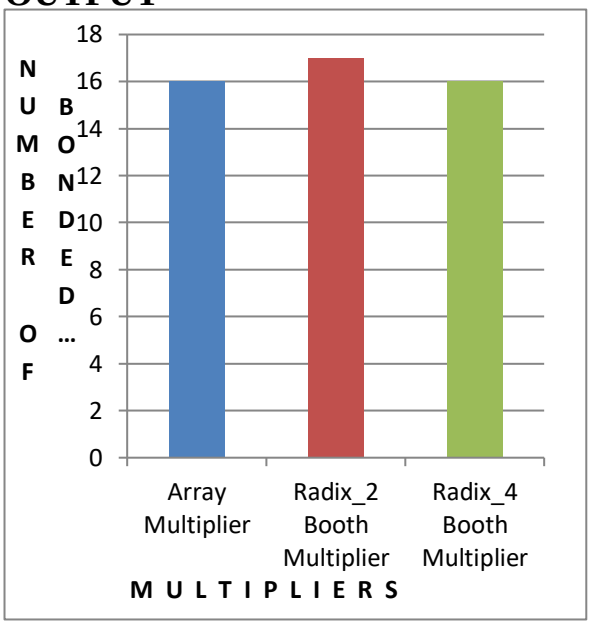
3.4.2 NUMBER OF 4 INPUT LUTs



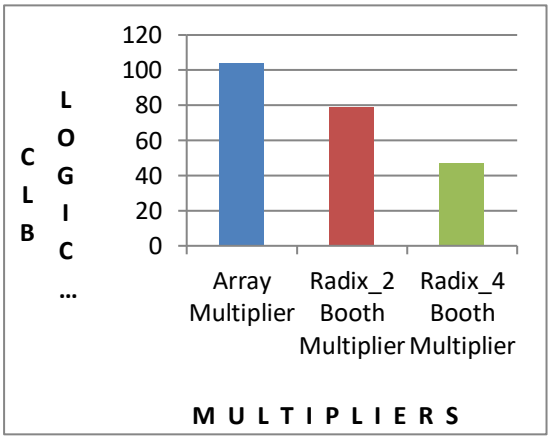
3.4.3 NUMBER OF BONDED INPUT



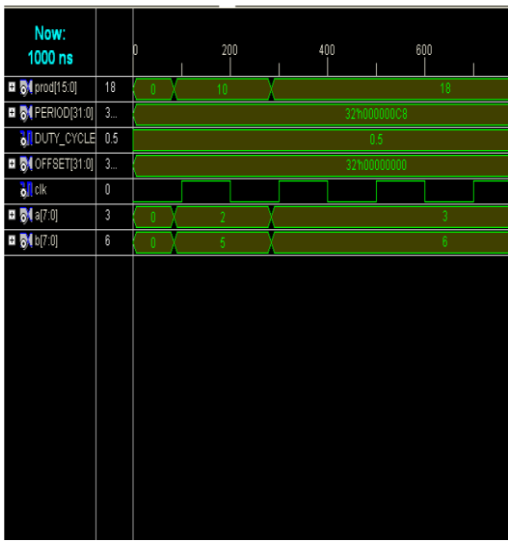
3.4.4 NUMBER OF BONDED OUTPUT



3.4.5 CLB LOGIC POWER



3.5 MULTIPLIER OUTPUT



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 7 Issue 05, May-2018

4 CONCLUSION

Our project gives a clear concept of different multiplier and their implementation in tapdelay FIR filter. We found that the parallel multipliers are much option than the serialmultiplier. We concluded this from the result of power consumption and the total area. Incase 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. Thisspeeds up the calculation and makes the system faster.

While comparing the radix 2 and the radix 4 booth multipliers we found that radix 4consumes lesser power than that of radix 2. This is because it uses almost half number ofiteration and adders when compared to radix 2.When all the three multipliers were compared we found that array multipliers are mostpower consuming and have the maximum area. This is because it uses a large number ofadders. As a result it slows down the system because now the system has to do a lot ofcalculation.

Multipliers are one the most important component of many systems. So we always needto find a better solution in case of multipliers. Our multipliers should always consumeless power and cover less power. So through our project we try to determine which of thethree algorithms works the best. In the end we determine that radix 4 modified boothalgorithm works the best.

5 FUTURE WORK

Multiplication is a most commonly used operation in many computing systems. Infact multiplication is nothing but addition since, multiplicand adds to itself multiplier no.of times gives the multiplication value between multiplier and multiplicand. But considering the fact that this kind of implementation really takes huge hardware resources and the circuit operates at utterly low speed. In order to address this so many ideas have been presented so far for the last three decades. Each one is aimed at particular improvement according to the requirement. One may be aimed at high clock speeds and another maybe aimed for low power or less area occupation. Either way ultimate job is to come up with an efficient architecture which can address three constraints of VLSI speed, area, and power. Among these three speeds is the one which requires special attention. If we observe closely multiplication operation involves two steps one is producing partial products and adding these partial products. Thus, the speed of a multiplier hardly depends on how fast generate the partial products and how fast we can add them together. If the number of partial products to be generated is of less than it indirectly means that we have achieved the speed in generating partial products. Booth's algorithms are meant for this only. To speed up the addition among the partial products we need fast adder architectures.

Since the multipliers have a significant impact on the performance of the entire system, many high performance algorithms and architectures have been proposed. The very high speed and dedicated multipliers are used in pipeline and vector computers. Thus, the requirement of the modern computer system is a dedicated and very high speed multiplier unit that can perform multiplication operation on signed as well as unsigned numbers.

6 REFERENCES

[1] [1] R. Jaikumar, P. Poongodi and R. Lavanya," Implementation of high speed arithmetic logic using vedic mathematics techniques" ictact journal on microelectronics, february 2015

[2] [2] M. Ramalatha, K. Deena, Dayalan ,Dharani "High Speed Energy Efficient ALU Design usingVedic Multiplication Techniques" ACTEA IEEE 2009.

[3] [3] N. Petra, D. D. Caro, V. Garofalo, E. Napoli, and A. G. M. Strollo, "Design of fixed width multipliers with linear compensation function", IEEETrans. Circuits Syst., vol. 58, no. 5, pp. 947960, May 2011.

[4] [4] Jiun-Ping Wang, Shiann-RongKuang, and Shish-Chang Liang, "High-Accuracy Fixed-Width Modied Booth Multipliers for Lossy Applications", IEEE Trans. Circuits Syst., vol. 19, no. 1, pp. 52-60, January 2011.

[5] [5] Yuan-Ho Chen, T.-Y. Chang, and C.-Y. Li, "Area-Effective and Power-Efficient Fixed-Width Booth Multipliers Using Generalized Probabilistic Estimation Bias", IEEE Trans. Circuits Syst., vol. 1, no. 3, pp. 277-287, September 2011.

[6] [6] Yuan-Ho Chen and T.-Y. Chang, "A High-Accuracy Adaptive Conditional-Probability Estimator for Fixed-Width Booth Multipliers", IEEE Trans. Circuits Syst., vol. 59, no. 3, pp. 594-603, March 2012.

[7] [7] Shin-Kai Chen, Chih-Wei Liu, "Design and Implementation of High-Speed and Energy-Efcient Variable-Latency Speculating Booth Multiplier (VLSBM)", IEEE Trans. Circuits Syst. I, vol. 60, no. 10, pp. 26312643, October 2013.

[8] [8] Shen-Fu Hsiao, Jun-Hong Zhang Jian, and Ming-Chih Chen, "Low-Cost FIR Filter Designs Based on Faithfully Rounded Truncated Multiple Constant Multiplication/Accumulation", IEEE Trans. Circuits Syst. II, Express Briefs, 2013.

[9] [9] O. L. MacSorley, "High speed arithmetic in binary computers", Proc.IRE, vol.49,pp. 67-91, 1961.

[10] [10] Parhami, Behrooz, "Computer Arithmetic: Algorithms and Hardware Designs", Oxford University Press 2000.

[11] [11] David H. K. Hoe, Chris Martinez and Sri JyothsnaVundavalli, "Design and Characterization of Parallel Prefix Adders using FPGAs", Proc. IEEE, pp. 168172, 2011.

[12] [12] Srinivasasamanoj R. ,M. Sri Hari and B. RatnaRaju, "High speed VLSI implementation of 256-bit Parallel Prefix Adders", International Journal of Wireless Communications and Networking Technologies, vol. 1, no. 1, 2012.

[13] [13] Shelja Jose, ShereenaMytheen, "Modified Booth Multiplier Based Low-Cost FIR Filter Design", International Journal of Engineering Scienceand Innovative Technology, vol. 3, September, 2014.

[14] [14] Phil E. Madrid, Brian Millar and Earl E. Swartzlander, Jr, "Modified Booth's algorithm for high radix fixed point multiplication", IEEE Transactions on Very Large Scale Integration (VLSI) Systems , Vol. 1, No. 2 June 1993.

[15] [15] Soniya, Suresh Kumar, "A Review of Different Type of Multipliers and Multiplier-Accumulator Unit", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 2, Issue 4, July – August 2013

[16] [16] GreeshmaHaridas,Dr. David Solomon George, "Area Efficient Low Power Modified Booth Multiplier for FIR", International Conference on Emerging Trends in Engineering, Science and Technology, Procedia Technology 24 (2016) 1163 – 1169