Design and Implementation of High-Speed Low Power and Error Tolerant Radix-4 Multiplier architecture on Spartan 3E FPGA

SK. Shakeela[1], Dr. E.V. Krishna Rao[2]
[1] Student of M. Tech, Dept. of ECE, LBRCE, L.B. Nagar, Mylavaram-521230, A.P, India
[2] Professor of ECE and Dean of Academics, LBRCE, L.B. Nagar, Mylavaram-521230, A.P, India

***Abstract*—**Multipliers has a dominant part in the digital world. It is a memoryless sub-block integrated into the microprocessor. It is the block with large delay and high power consumption.In this paper, the Booth multipliers are discussed.The main advantage of using Booth multipliers is reducing the partial products which reduce power consumption of the device. This methodology helps to achieve low power with high performance and less complex circuits. Here, we study Radix-4 8-bit and 16-bit Booth multipliers using approximate Booth encoders. This is Implemented on SPARTAN 3E FPGA. This entire design is structured, implemented in Verilog HDL language on Xilinx 14.7 design Tool.The error tolerant computing is analyzed by using two approximate Booth encoders with respect to approximate factor. In image processing the peak signal-to-noise ratio (PSNR) is found to assess the quality of image by using two approximate booth encoders in Booth multiplier.

Keywords—Radix-4 Multiplier, Booth Encoder, Approximate computing, High speed, low power , PSNR.

I. INTRODUCTION

Multipliers are used in many different places in microprocessor design. It is the non-memory sub-block of the microprocessor with the largest size and delay that has a big impact on the cycle time. Multipliers are also very important and frequently used in Digital Signal Processing applications to run complex high-speed calculations. [1]

Booth multiplication is used critically to increase the speed of the multiplier by encoding the multiplicands that are multiplied. This technique is standard and used in the design of a sub-block in the chip and provides major enhancements over the traditional technique. In the conventional multiplier, the number of partial products to be added is determined by the number of bits the multiplier or multiplicand being used. The bigger the number of bits the multiplicand or the multiplier contains, the longer time it takes to produce the product. The delay of the multiplier is determined largely by the number of partial products to be added. One of the most popular algorithms used to reduce the number of partial products is Booth encoding multiplier. Booth encoding multiplication can reduce the number of partial products being encoded to increase the speed of the binary multiplications. Radix-4 Booth encoding multiplier reduces the number of partial products by half, N/2. [2] This can increase the time of compression and contribute to an increase in speed. [3]

Multipliers typically consist of a partial product matrix (PPM), which accumulates the partial products and reduces them to just two operands, and a last stage Carry Propagate Adders (CPA).The speed of multiplication can be least significant bit (LSB) position of each partial product increased by reducing the number of partial products and/or accelerating the accumulation of partial products approaches namely booth algorithms using Wallace Tree 4-2 Compressors, Carry Propagate Adders (CPA).[5]Error-tolerant the technique used is Normalization of error distance. The approximate design of a radix-4 Booth multiplier is one of the most popular schemes for signed multiplication. A radix-4 Booth multiplier, a radix-4 modified Booth encoding (MBE) is used to generate the partial products.

The implementation of the MBE significantly affects the area, delay and power consumption of Booth multiplier. In the traditional MBE algorithm, an extra partial product bit is generated at the least significant of row due to the negative encoding. This leads to an irregular partial product array as requiring a complex reduction tree. A more efficient approximate radix-4 Booth encoder is proposed in this paper. The designs of both approximate radix-4 Booth encoders are presented and extensively analyzed .The multiplicand encoding process using radix -4 Booth algorithm is based on the multiplier bits. It will compare 3 bits at a time with overlapping technique. Grouping starts from the LSB, and the first block uses only two bits of the multiplier and assumes a zero for the third bit. It consists of eight different types of states as we are comparing 3bits at a time and during these states, we can obtain the outcomes, which are a multiplication of multiplicand with 0,-1 and -2 consecutively. The state diagram presents various logics to perform the Radix-4 Booth multiplication in different states as per the adopting encoding technique is shown in Figure.1.

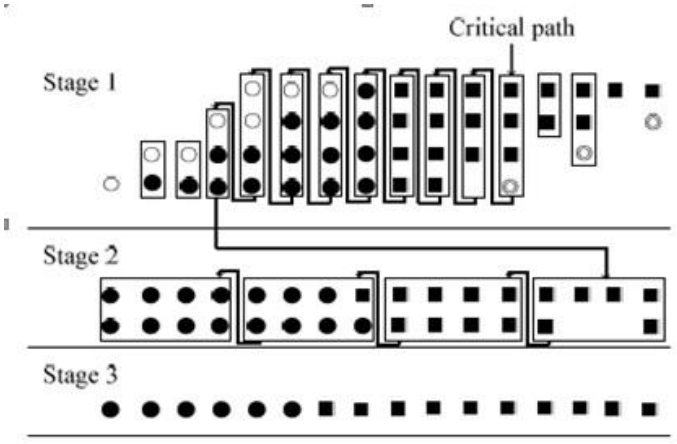

Figure.1: Radix-4 Booth multiplication

II. BACKGROUND STUDY

A. RADIX-4APPROXIMATE BOOTH MULTIPLIER

Radix-4 Booth multiplier in which they are two binary inputs as input, one is multiplicand and another one is a multiplier. If both binary numbers are positive then it will perform two booth encoders. Approximate Booth multipliers are designed based on the approximate Booth encoders and the approximate regular array of partial products.

Booth encoding has been anticipated for improving the performance of multiplication of twos complement binary numbers, it has been further improved by the MBE or radix-4 Booth encoding. The Booth encoder has a vital role in the Booth multiplier, which cuts the number of partial product rows by half. Approximate radix-4 booth multiplier designed based on two approximate booth encoders those are Radix4 encoder1 and Radix4 encoder2.

B. METHOD1: RADIX-4 BOOTH ENCODING APPROXIMATION

The benefit of the Radix4 encoder1 design is an error is less. In the approximate Booth multiplier, the proposed approximate Booth encoder Radix4 encoder1 shown in Figure.2 is used in the first part to generate the inexact partial products. The approximate Booth encoders can then be used in all or only part of the partial product generation process; therefore, an approximation factor p (p=1, 2, ..., 2N) is defined as the number of least significant partial product columns that are generated by the approximate Booth encoder.

This Booth encoder is the conventional design of modified booth encoder. The traditionally modified booth encoder has more delay but this Booth encoder has less delay. This encoder reduces the complexity. The conventional MBE, as only entries are modified, however, all modifications change a ′1′ to a ′0′. The absolute value of the approximate product is always larger than its exact.

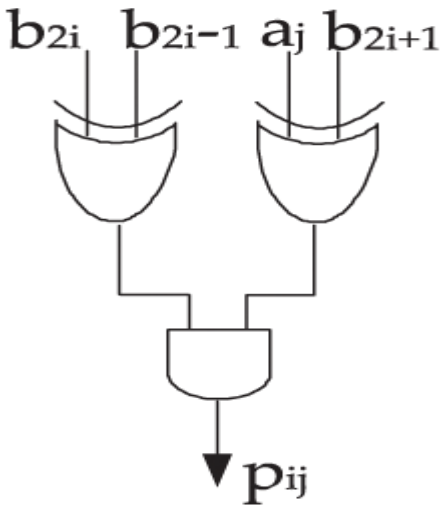

Figure.2: Radix4 encoder1

The approximate radix-4 Booth multiplier (Radix4 multiplier 1) uses radix4 encoder1 (to generate the p least significant partial product columns) and the regular approximate partial product array. The exact MBE is used for generating the 2N-p most significant partial product columns. The exact 4-2 compressors are used to accumulate both approximate and exact partial products.

Approximate radix-4 Booth multiplier (Radix4 multiplier2) uses Radix4 encoder2 shown in Figure.3 the regular approximate partial product array and the exact 4-2 compressors. The error is controlled by the approximation factor of a counterpart. Compared with the exact MBE, Radix4 encoder1 can significantly reduce the critical path delay of Booth encoding.

C. METHOD2: RADIX-4 BOOTH ENCODING APPROXIMATION

Radix4 encoder1, the modification is achieved by not only changing a ′1′ to a ′0′ but also changing a ′0′ to a ′1′. The approximate product can be either larger or smaller. Then the exact product and errors can complement each other in the partial product reduction process. Radix4 encoder2 in a Booth multiplier, the error may not be larger than for a Booth multiplier with Radix4 encoder1.

Figure.3: Radix4 encoder2

D. APPROXIMATE WALLACE TREE DESIGN

The reduction of partial products using full adders as 4-2 compressors became generally known as the "Wallace Tree". This architecture reduces the partial product rate. The tree reduction for an 8*8 bit partial product tree, the ovals around the dots represent either a full adder or a Half adder. This is reducing the need for carry propagation in the adder avoid the latency of one addition is equal to gate delay of the adder.
The chip/system designers add accuracy as a new constraint to optimize Latency-Power-Area metrics. In this paper, we present a new power and area-efficient Approximate Wallace Tree structure (AWTs).

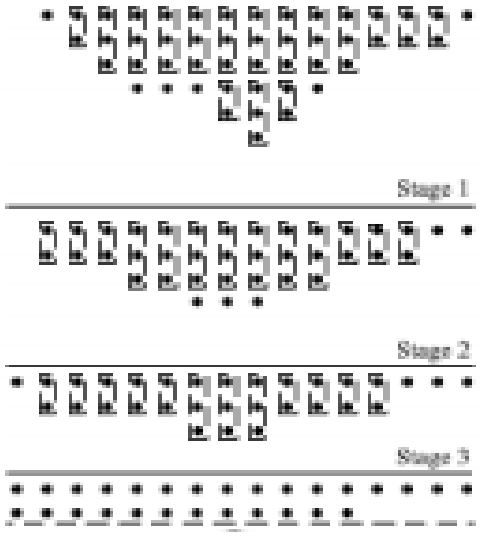

Figure.3: Approximate Wallace tree structure

E. COMPRESSOR 4-2:

[15]There are various ways to implement the 4-2 compressor. In Figure.4 the simplest of which is cascading two full adder cells together shown in Figure.5. This structure was a popularized by Santoro who built a 64*64 bit multiplier. The interconnection of these full adder must not make the carry out dependent on the carry in. It takes two full adders to build one 4-2 compressor. The speed of each 4-2 compressor is limited by the delay of 3 XOR gates in series. Thus, a 4-2 is slightly faster than the two-full adder. Thus, the tree compression is slightly faster using 4-2 with a trade off a little more hardware.

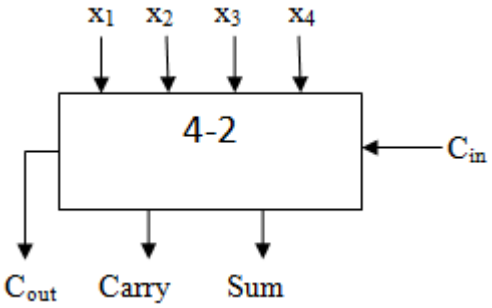

Figure.4: 4-2 Compressor

The compressor has 5 inputs x_1, x_2, x_3, x_4 and C_{in} to generate 3 outputs Sum, Carry and C_{out}. The 4 inputs x1, x2, x3 and x4 and the output Sum have the same weight. The input C_{in} is output from a previous lower significant compressor and the C_{out} output is for the compressor in the next significant stage.

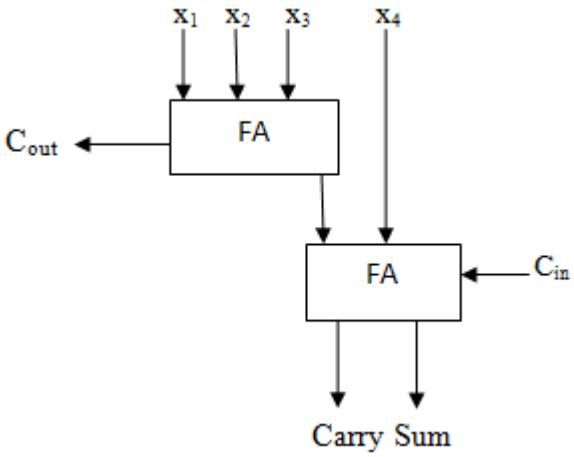

Figure.5: 4-2 Compressor using full adder

The main goal of either multi-operand carry-save addition or parallel multiplication is to reduce n numbers to two numbers; therefore, n-2 compressors (or n-2 counters) have been widely used in computer arithmetic. A 4-2 compressor is usually a slice of a circuit that reduces n numbers to two numbers when properly replicated. A widely used structure for compression is the 4-2 compressor.
This gives the outputs of the 4- 2 compressor, the common implementation of a 4-2 compressor is accomplished by utilizing two full-adder (FA) cells

$$Sum = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus C_{in}$$

$$C_{out} = (x_1 \oplus x_2)\, x_3 + \overline{(x_1 \oplus x_2)}\, x_1$$

$$Carry = (x_1 \oplus x_2 \oplus x_3 \oplus x_4)\, C_{in} + \overline{(x_1 \oplus x_2 \oplus x_3 \oplus x_4)}\, x_4$$

F. CARRY LOOK AHEAD ADDER(CLA):

[16] The ripple-carry adder, its limiting factor is the time it takes to propagate the carry. The carry look-ahead adder answers this problem by calculating the carry signals in advance, based on the input signals. The result is a reduced carry propagation time. To be able to understand how the carry look-ahead adder works we must manipulate the Boolean expression dealing with the full adder.

The Propagate P and generate G in a full-adder, is given as:

$$P_i = x_i \oplus y_i \quad \text{Carry propagate}$$

$$G_i = x_i y_i \quad \text{Carry generate}$$

Notice that both propagate and generate signals depend only on the input bits and thus will be valid after one gate delay.

The new expressions for the output sum and the carryout are given by:

$$S_i = P_i \oplus C_{i-1}$$

$$C_{i+1} = G_i + P_i C_i$$

Although it is impractical to have a single level of carry look – ahead logic for log adder

Hence equations show that a carry signal will be generated in two cases:

- if both bits x_i and y_i are 1
- if either x_i or y_i is 1 and the carry-in C_i is1.

It is shown in the Figure.6 Carry look ahead adder.

To make the carry generator from 4 bits to n bits, we need only add AND gates and inputs for the OR gate. The largest AND gate in the carry section has always n+1 inputs and the number of AND gates requirements is n.

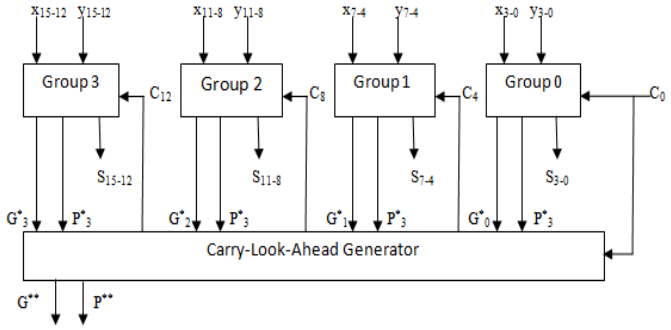

Figure.6: carry look ahead adder

G. ERROR ANALYSIS AND EVALUATION:

[17] The approximate designs, several metrics have been proposed to measure the error of approximate adders and multipliers including the mean error distance (MED), the relative error distance (RED) and the normalization of MED (NMED). Several error metrics are used to fairly compare different approximate designs of various sizes. Here the error distance (ED), relative error distance (RED)and Normalized error distance are found out for two approximate multipliers. Where ED is defined as the difference between approximate value and exact value. RED is defined as error distance by the output of exact value. Normalized error distance is defined as the ratio of error distance to the square root of relative error distance.

The error rate of approximate booth encoder1 will be significantly smaller than for approximate encoder2.

The below table 1 and 2 represents the error of approximate multipliers at different approximate factors which is nothing but an error distance, this is further demonstrated with the peak signal to noise ratio(PSNR) results for the image application.

Table1: Error analysis of Radix4 multiplier1

Approximate factor(p)	RED	NED(10^{-2})
2	0.0085	0.021
4	0.017	0.30
6	0.024	0.38
8	0.031	0.45
10	0.043	0.48
12	0.052	0.52

Table2: Error analysis of Radix4 multiplier2

Approximate factor(p)	RED	NED(10^{-2})
2	0.0087	0.021
4	0.0168	0.30
6	0.027	0.46
8	0.031	0.48
10	0.054	0.51
12	0.059	0.63

III. PROPOSED METHOD

In this proposed method, we implement Radix-4 8-bit and 16-bit Booth multipliers. In last part of Booth multipliers to compute final product result we use different adder which is parallel prefix adder. Parallel-prefix adders, also known as carry-tree adders, which pre-compute the propagate and generate signals. These signals are variously combined using the fundamental carry operator (fco). Due to associative property of the fco, these operators can be combined in different ways to form various adder structures. The parallel prefix adder used in this multiplier is spanning tree adder. It has been proved that it can be useful to apply a radix-4 architecture in high-speed multipliers. Figure.8 shows the 16-Bit Spanning tree adder.

Spanning tree adder is a very widespread and extensively used adder. It is one of the parallel prefix adders. These adders are the ultimate class of adders that are based on the use of generating and propagate signals.

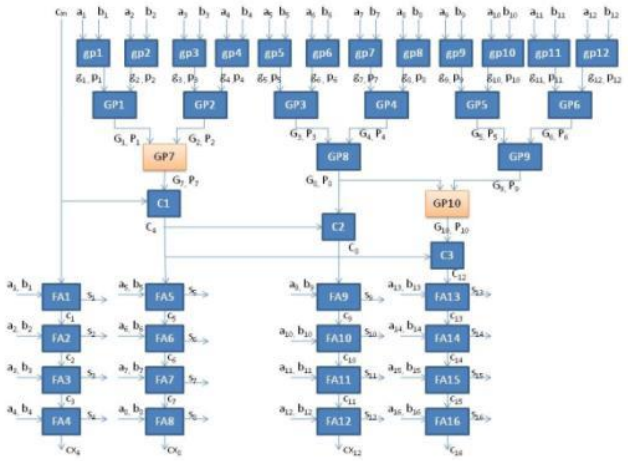

Figure.8: 16-Bit Spanning tree adder.

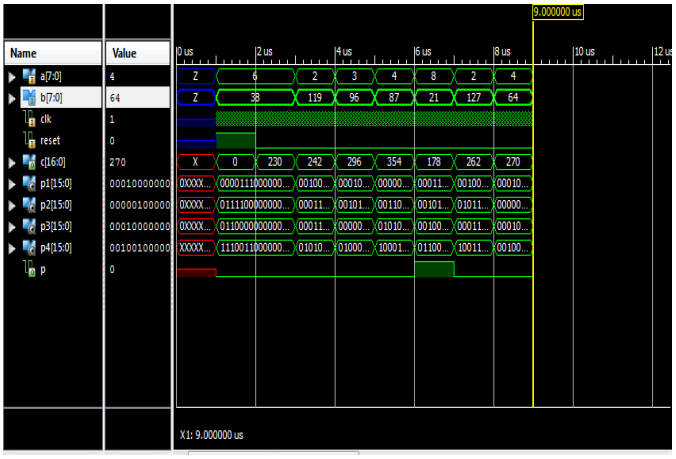

Figure.10: Radix4 multiplier2

IV. SIMULATION RESULTS AND DISCUSSION

In this Section, the simulation results of Radix4 multiplier1 ,2 is shown in Figure.9,10 and11,12 using spanning tree adder and Radix4 multiplier1,2 using 16 bit is shown in Figure.13,14. From these Simulation results, it is observed that error tolerance is greatly reduced in order from Radix-4 8-bit to 16-bit.

The Graphical representation of these multipliers in terms of Area, Power and Delay are shown in the respective Table.3. From these Figures.15,16,17, it is visualized that this proposed method helps to reduce power, delay but area increases gradually with the design due to the use of a compressor

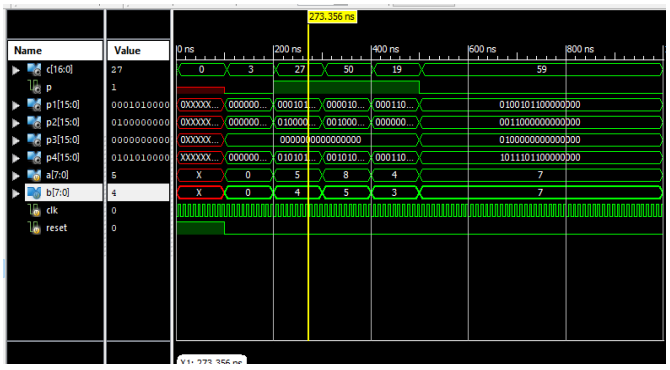

Figure.11: Radix4 multiplier1(using spanning tree adder)

Radix-4 8-Bit Approximate Booth multiplier

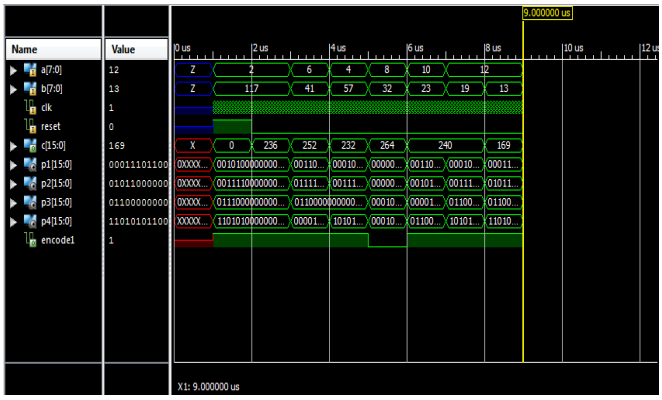

Figure.9: Radix4 multiplier1

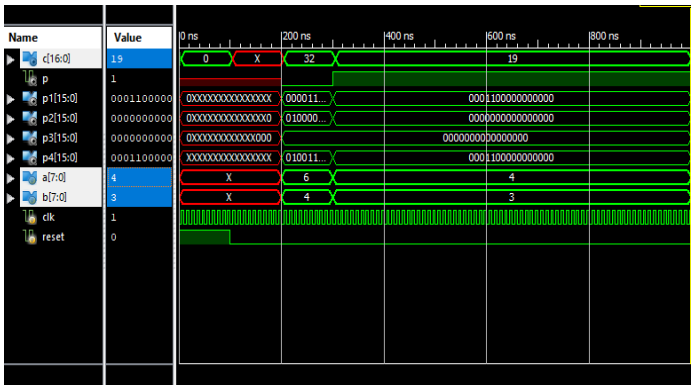

Figure.12: Radix4 multiplier2(using spanning tree adder)

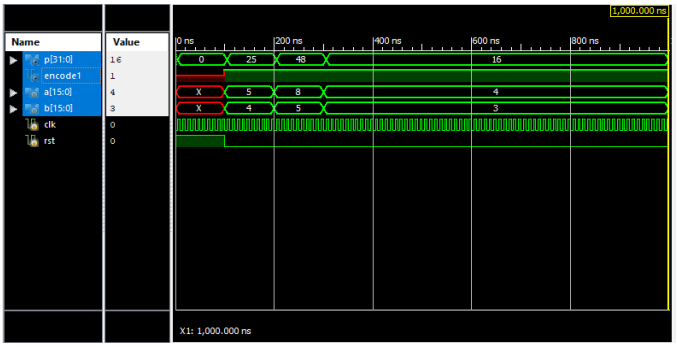

Figure.13: Radix4 16-bit Multiplier 1

Figure.14: Radix4 16-bit Multiplier2

Table.3: Radix4 8-bit multiplier vs 16-bit multiplier

Parameters	Radix -4 8-bit multiplier	Radix -4 8-bit Spanning tree	Radix -4 16-bit multiplier
Total Power(mw)	119.92	114.76	82
Total Delay(ns)	29.050	25.545	7.00
Total Area(slices)	307	276	617

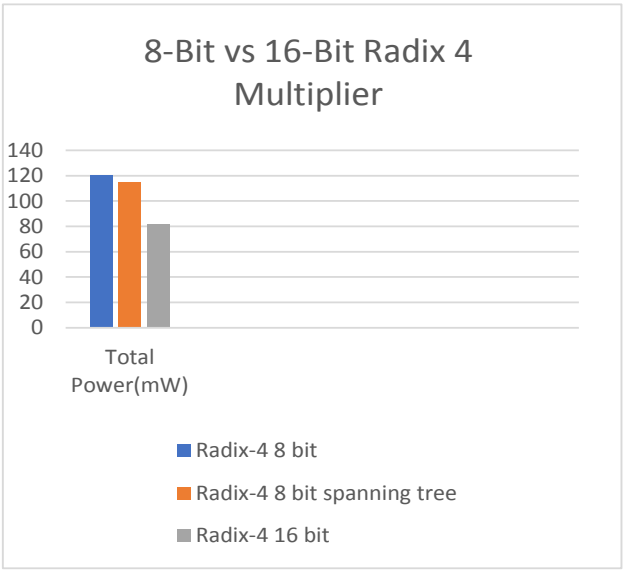

Figure.15: Total Power radix4 8-bit multiplier vs 16-bit multiplier

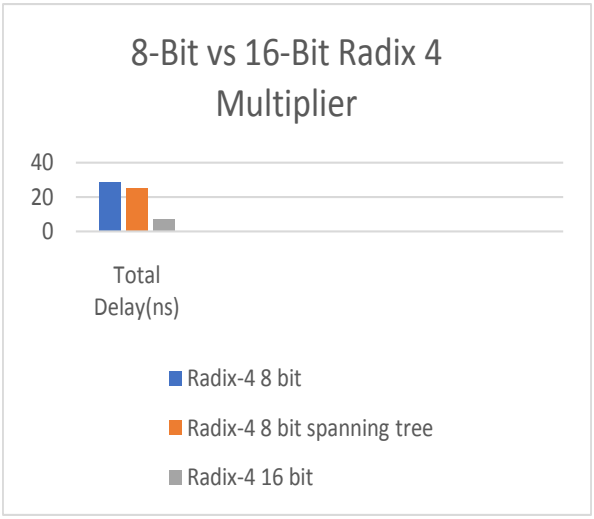

Figure.16: Total Delay radix4 8-bit multiplier vs 16-bit multiplier

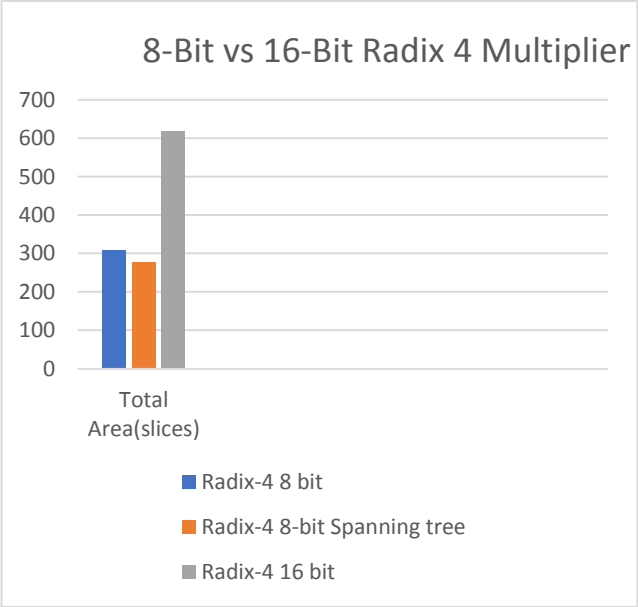

Figure.17: Total Area radix4 8-bit multiplier vs 16-bit multiplier

V. IMAGE PROCESSING:

In this section, the Radix4 multiplier1 and multiplier2 with various approximate factors are applied to image processing. The peak signal-to noise ratio (PSNR) is used to assess the quality of the output image.

The below Table.4 shows the PSNR value of Images using booth multipliers at different approximation factors. These are calculated in C using MATLAB .From this Table.4 the PSNR of Radix4 Multiplier2 is higher than that of Radix4 Multiplier1 .So, it has better quality of image. The images using Radix 4 multipliers1 and 2 at different approximate factors are shown in Fig.18 and Fig.19.

(a) (b) (c)

(d) (e) (f)

Fig.17:ImagesusingRadix4 multiplier1 atdifferent approximate factor.

(a) p=2,(b) p=4,(c) p=6,(d) p=8,(e) p=10,(f) p=12

(a) (b) (c)

(d) (e) (f)

Fig.18:Images usingRadix4multiplier2 atdifferent approximate factor.

(a) p=2,(b) p=4,(c) p=6,(d) p=8,(e) p=10,(f) p=12.

Table.4: PSNR (dB) of Images Radix4 multipliers at different approximation factors

Approximate Factor(p)	PSNR(dB)	
	Radix4 Multiplier1	Radix4 Multiplier2
2	13.351	13.35
4	13.22	13.54
6	13.84	15.12
8	14.21	16.15
10	13.46	14.157
12	11.57	12.36

VI. CONCLUSION

Multipliers are crucial and recurrently used in Digital Signal Processing applications to run complex high-speed calculations. This is Implemented on SPARTAN 3E FPGA. This proposed method shows different methods of radix4 and those parameters like power, delay and area. Error tolerance is increased for higher Radix states.This Approximate multiplier finds its application in space signal computing for retrieving and analysing the data from the received signal. It also helps in Digital image processing, and in DSP architectures to reduce noise ratio.This entire design is structured, implemented in Verilog HDL language on Xilinx 14.7 design Tool.The proposed multipliers have also been applied to image processing,resulting in very small loss of accuracy .

VII. APPLICATIONS

Radix Multipliers plays a large role in the digital era. Major data transmission is done in 0 and 1. There is a noise which occurs in transmission of the data. Hence approximation of data is vital for every transmission. Thus, error tolerant computing has its vast applications in Digital Image processing and Space signal processing and large scope in scientific applications.

VIII. REFERENCES

[1] Bryan C. Catanzaro, Department of Electrical and Computer Engineering Brigham Young University, "Higher Radix Floating-Point Representations for FPGA-Based Arithmetic", 2005.

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

[3] C. N.Marimuthu, P. Thangaraj, Anna University, India, "Low Power High Performance Multiplier", ICGST-PDCS International Conference on Computer Science and Engineering, 2008

[4] S. Venkataramani, S. T. Chakradhar, and K. Roy. "Computing approximately, and efficiently," Proc. Design, Automation & Test in Europe Conference& Exhibition (DATE), 2015, pp.748-751.

[5] Liu, W., Qian, L., Wang, C., Jiang, H., Han, J., and Lombardi, F. (2017). Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing. IEEE Transactions on Computers *66*, 1435–1441.

[6] J. Han and M. Orshansky, "Approximate computing: an emerging paradigm for energy-efficient design," Proc. 18th IEEE European Test Symposium,2013, pp.1-6.

[7] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," IEEE Trans. CircuitsSyst. I, Reg. Papers, vol. 57, pp. 850-862, 2010.

[8] V. Gupta, D. Mohapatra, S. Park, A. Raghunathan, and K. Roy,"IMPACT: IMPrecise Adders for Low-Power Approximate Computing," Proc. Int. Symp. Low Power Electronics and Design(ISLPED), pp. 1-3, 2011.

[9] W. Liu, L. Chen, C. Wang, M. O′Neill and F. Lombardi, "Design andanalysis of floating-point adders", IEEE Trans. Computers, vol. 65, pp.308-314, Jan. 2016.

[10] J. Liang, J. Han, and F. Lombardi, "New metrics for the reliability of approximateand probabilistic adders," IEEE Trans. Computers, vol. 63, pp.1760-1771, Sep. 2013.

[11] A. Booth, "A signed binary multiplication technique," Quarterly J. Mechanicsand Applied Mathematics, vol. 4, pp/236-240, June 1951.

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

[13] K.-J. Cho, K.-C. Lee, J.-G. Chung, and K. K. Parhi, "Design of low error fixed-width modified Booth multiplier," IEEE Trans. VLSI Systems, vol.12, no. 5, pp. 522–531, 2004.

[14] M. J. Schulte and E. E. Swartzlander Jr., "Truncated multiplication with correction constant," Proc. Workshop VLSI Signal Process. VI, 1993, pp.388–396.

[15] International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 4, Issue 11, November 2015 Design of Low Power Approximate Compressor For Dadda Multiplier S.Sudha, M.Revathy

[16]http://users.encs.concordia.ca/~asim/coe

[17] Liang, J., Han, J., and Lombardi, F. (2013). New metrics for the reliability of approximate and probabilistic adders. IEEE Transactions on Computers *62*,