ESP-IJCEET
ESP International Journal of Communication Engineering & Electronics Technology
ISSN: 2583-9217 / Volume 1 Issue 2 July 2023 / Page No: 12-19
Paper Id: IJCEET-V1I2P103 / Doi: 10.56472/25839217/ IJCEET-V1I2P103

Original Article

Design and Analysis of RADIX-8 Based 32-bit Pipelined Multiplier

Mummareddy Ramya Krishna[1], B. Ramana Kumar[2]

[1,2]*Department of Electronics and Communication Engineering. ISTS Womens Engineering College Andhra pradesh, East Gonagudem, India*

Received Date: 05 July 2023 Revised Date: 25 July 2023 Accepted Date: 31 July 2023

Abstract: *In microprocessors, microcontrollers, and signal processing applications, arithmetic operations are essential. Digital Signal Processing (DSP) modules such as Infinite Impulse Response (IIR) filters, Finite Impulse Response (FIR) filters, Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), and others frequently employ the multiplication operation. The main issue in multiplier design is to reduce power dissipation while enhancing performance. Different logic combinations are utilized in power reduction strategies like Recoding (RADIX-8 Modified Booth Encoding (MBE) structure) to generate Partial Product Rows (PPRs), which are employed in modified pipelined multipliers. For addition between produced PPRs, two addition algorithms are used: sequential based CLA and tree based carry Look-a-head Adder (CLA). When comparing the performance of the RADIX-8 pipelined multiplier with the above-mentioned methodologies to the RADIX-4 pipelined multiplier that is currently in use, the updated method produces a significant, somewhat crucial path latency, area, and power consumption reduction. FPGA technology is used in XILINX 14.7 to implement modified design.FC320-5 XC3S500E.*

Keywords: *Pipelined Multiplier, RADIX-8 MBE, Partial Products Generation, Sequential and Tree Based CLA Addition.*

I. INTRODUCTION

Digital Signal Processing (DSP) is widely used in advanced consumer electronics, offering accelerators for general-purpose, military, and communications systems. Numerous arithmetic processes, including the Finite Impulse Response (FIR), Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), and Infinite Impulse Response (IIR), are implemented in DSP applications using computationally demanding kernels. Arithmetic unit architecture and design allocation can be used to assess the performance of a DSP system. The subject of efficient arithmetic operations has expanded substantially in recent years. Arithmetic procedures (sub modules) are used to transport data between various digital modules. The multiplier submodule is one of the crucial ones. A variety of multiplier designs were included to improve DSP algorithm implementations that were more effective. Numerous strategies have been put out to maximize the multiplier operation's performance in terms of space and power usage. To effectively map flexible DSP crucial route fusion. One way to accomplish this is by using mathematical units like multipliers. Multiplier operation may be used to build a wide range of DSP applications.

In terms of hardware, multipliers can be implemented as pipelined or combinational multipliers. The application of combinational multipliers increases the critical path in two steps and the amount of hardware resources. N*N number of inputs resulted in N number of PPRs at stage 1. The combinational approach is inefficient for big multiplier design in stage 2, as the number of adders grows for the design. Pipelined multiplier uses recoding structure to get over the aforementioned drawbacks. Section 2 discusses addition strategies and partial product reduction techniques with an emphasis on pipelined multiplier unit optimization. Below is a discussion of the PPRs generation and adding procedures that are involved in designing various multipliers with restrictions.

Verilog HDL is used to create the CLA-based 32-Bit Signed Pipelined Multiplier, which achieves efficient critical path latency and area but lacks information on power usage. Comparing this pipelined multiplier to a standard multiplication procedure, half (16) Partial Product Rows (PPRs) are produced. To add PPRs, fifteen CLA adders are needed [1]. An alternative

method involves implementing a 32-bit high-speed pipelined multiplier based on FSM using Verilog HDL. This method also has efficient critical path latency, but it is larger than [1] and yields 64-bit addition of 32 partial products, which results in higher power consumption [2]. Another design is an 180nm CMOS version of a 32-bit pipelined multiplier, which is also delay efficient but less than [1], [2], and has no information regarding power consumption [3]. In this study, MBE is used to produce partial products, while the CSA-CCA architecture is used to add final PPRs. The next approach has been suggested to create a partial product array utilizing MBE and a post-truncated technique that use Wallace tree structure to shrink the array. CLA is used to add the partial products array in its final form [4]. Another strategy examines the power at various voltage levels and applies array-based multipliers in various ways [5]. In this article, created partial products addition and 32-bit multiplier PPRs are generated using RADIX-8 MBE. Finished using two methods CLA sequentially based and CLA tree based.

II. MODIFIED PIPELINED MULTIPLIER ARCHITECTURE

Every multiplier design should carry out two processes: the creation of partial products and the addition of partial products. However, the improved architecture performs addition operations between the Partial Product Rows (PPRs) and creates them directly. When the hardware of the design is increased in a regular multiplier, as was covered in Section 1, both the logic and the routing size should rise. The critical route latency and power consumption of the design will likewise grow with an increase in the design hardware [8] [9].Using the RADIX technique lessens this effect [1]. Maximum PPRs are decreased with modified multiplier design and the RADIX algorithm. An alternative strategy, the pipeline approach, which is shown in [1] [2] [7], solves delay and area issues. For the creation of partial products, many methods are used [1–10]. For instance, the Wallace Tree, Bough Wooley, BOOTH, and Modified Booth Encoding (MBE) algorithms enhance the area, power, and delay performance of multipliers. Shifting (left or right shift) [5] of PPRs has an additional significance in the construction of these algorithms. The modified design generates PPRs using the MBE algorithm. The addition of PPRs is the next phase, and it may be completed with a variety of adders, including the Carry Look-ahead Adder (CLA), Carry Select (CSLA), Carry Save (CSA), and Carry Skip Adders (CSPA). Moreover, adders improve the multiplier response. Different structures are created for the addition of PPRs since the advanced multiplier requires a number of adders [1]. There were two structures included in the modified design. 1. CLA addition based on TREE. 2. CLA addition based on Sequential. Figure 1 shows the design process step-by-step.

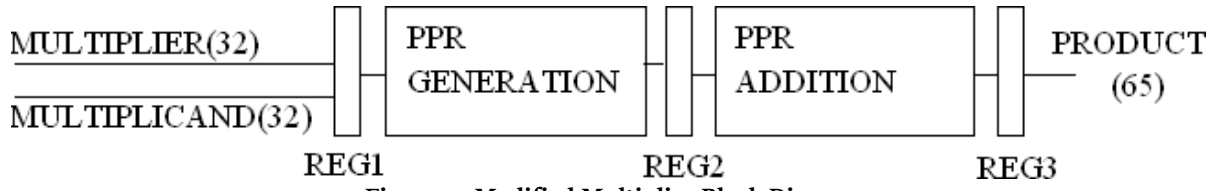

Figure 1: Modified Multiplier Block Diagram

First block is register1 which is implemented by using D-flip flops. This register stores the inputs multiplicand (32-bit) and multiplier (32-bit). Total 64- bits are stored in this register with the help of cascaded 64 D-flip flops. Next important block is PPRs Generation.

A. PPRs Generation:

This block is important for modified design. Different partial products generations are observed in [1-10]. Modified Booth Encoding [MBE] is adopted from [1] and its extension is RADIX-8 method. RADIX-8 MBE accesses the input data (multiplier [Y31-Y0] and multiplicand [X31-X0]) from register1. Complement and shift operation involved in RADIX-8 MBE. RADIX-8 has 16 different operations to generate PPRs. here 4 bits group is taken as a control of each operation so valueofthe4thbitvalueis8.SimilarlyforRADIX-4structure 3 Bits group is taken as a control of each operation so value ofthe 3rd bit is 4. Large multiplication process requires advanced algorithms to reduce PPRs. Dot diagram representation of PPRs generation is shown in figure 2 with the help of RADIX- 8structure.

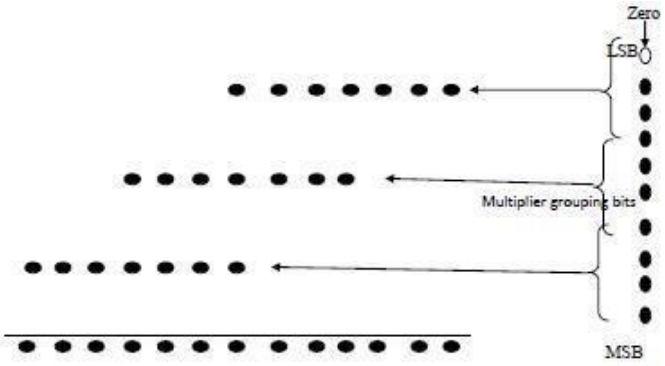

Figure 2: Dot Diagram Representation of PPR Generation

a) Algorithm of RADIX-8 structure: Steps to generate PPRsusing MBE Radix-8:

- Multiplier 'Y' LSB is taken as 'o' and remaining inputis same.
- The bit group T, where T = 4 of the multiplier "Y", different combinations of each group Zk, where n-1 k o. The rule for each Zk group is such that (Yi+2 Yi+1 Yi Yi-1), as shown in table 1.
- Now depending on K, where n-1 ¡= k ¡= o, where Skis the value PPR in the Table for all possible combinations of torque values Zk.
- Add these PPRs based on multiplicand value to get the final product. Mathematical expression of RADIX-8 MBE

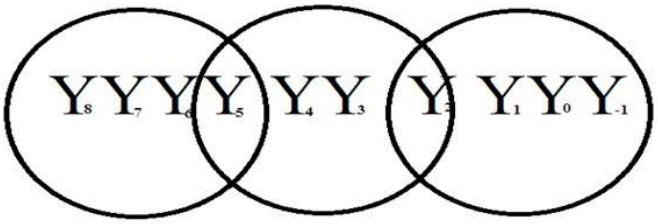

Figure 3: Multiplier Grouping Bits Using RADIX-8

Expressed as

$$X*Y=PP_0 2^0+PP_1 2^3+PP_2 2^6+PP_3 2^9+PP_4 2^{12}+....(1)$$

- Figure 2 shows the bit operands for the 16 MBE operations.
- Using the procedure described as shown in table 1, it operates dispersed. The four bit group multiplier bit is formed in the respective multiplication operations to generate PPR. Number of PPRs generated by using RADIX-8 encoding is n/3. Here n means number of bits in multiplier and we perform 32-bit multiplication process So 11 PPRS are generated. Reduce five of the partial products compared with RADIX-4 MBE multiplication process. Different operations selected based on the multiplier grouping bits like addition, subtraction (2s complement) shown in table1.
- RADIX-8 MBE generates PPRS Example: Take inputs multiplicand and multiplier 32-bit each shown in below Multiplicand: 00000000000000000000000000001010
- Multiplier: 00000000000000000000000000001010 (Groupingfour bits of the 32-bit multiplier)
- The partial product length is two bits longer than the multiplicand length, giving 35-bit length partial products. Below 1 to 11 represents multiplier 'Y' groups like as shown in figure
- 3. In multiplier we take LSB as 0 then multiplier bits are11 9 7 5 3 1
- 00000000000000000000000000010100
- 10 8 6 4 2
- We use 11 16*1 multiplexers are used for modified booth encoding mechanism. From the above example first four digitsof multiplier group represents 1 left shift of multiple and where LSB is 'o' then
- 00000000000000000000000000010100 (decimal value 20)

- Partial Product Row1 (PPR1) Second four digits of multiplier represent same as multiplicand operation then 00000000000000000000000000001010 (decimal value 10)
- Partial Product Row2 (PPR2)
- PPR2 is shifted into 3 places left then value is 80 (00000000000000000000000001010000)
- Here shifting means two PPRs are reduced by using RADIX- 8.Remaining PPRs are neglected because of multiplier next grouping bits are zeros then generated PPRs also zeros. PPRs generated by using 16*1MUX with different operations based on selection lines shown in figure 4. A multiplexer is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexerof 2n inputs has n select lines. A high-radix Booth encoding technique can reduce the number of PPRs with the help of MUX. Finally 11 partial product rows each 35-bit totally 385 partial product bits are generated then these partial products are kept in check by use of a 385 D-Flip-Flop Register.

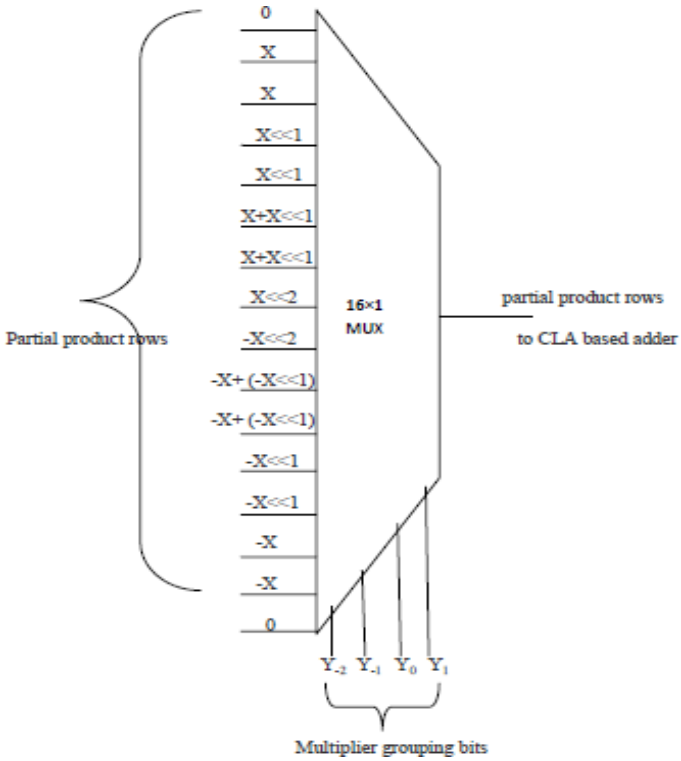

Figure 4: 16X1 MUX Selecting Operation For PPR

Create N/2 partial product rows for the N*N multiplier in RADIX-4 MBE. Reduce the partial products in RADIX-8 MBE to less than N /2. In light of this, RADIX-8 MBE produces fewer partial products than RADIX-4 MBE. Various complement and normal left shift operations were carried out using the information supplied in RADIX-8 table 1.

B. PPRs Addition:

Implement the design analysis comparison of partial products addition for best approach using RADIX-4 and RADIX-8 MBEs with CLA sequential and CLA tree based addition process. RADIX-8 got efficient critical path and power consumption when compared with RADIX-4 MBE. Power is optimized in radix-8 based addition over RADIX-4. Final functional verification value in decimal and binary forms of generated partial products is

From above example PP0+PP1=20+80=100000pp1+pp2000=00000000000000000000000000010100 +00000000000000000000000001010000000000000000000000000001100100 (decimal value-100) Above addition can be done by using CLA Adder with two different approaches 1.CLA Sequential based approach 2.CLA Tree based approach

a) CLA Sequential Based Approach:

In this method, the input of the next adder receives the output of the previous adder, keeping the adder size constant. This adder input is currently waiting for previous adder output because of increased design power limits and crucial path latency. Figures 10 through 13 display the values of those comparative results. As seen in figure 5, we have a total of 11 PPRs, and we add between them by employing ten CLA adders.

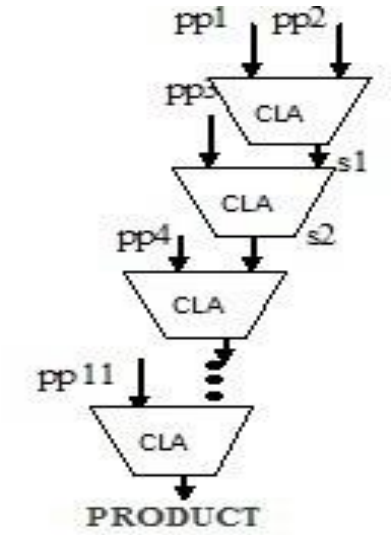

Figure 5: RADIX-8 Sequential based CLA Addition

b) CLA Tree-based Method:

The Tree-based approach overcomes the drawbacks of the First approach. Prior to the addition procedure, each PPR bit is moved three times to the left. The first PPR is 35 bits, and the second is left shifted to 3 times (38 bits) in relation to the first. The following one is similarly left-shifted to three times in relation to the preceding one. This method is repeated until the final PPR is reached. The same amounts of adders were used in this addition procedure, but the figure illustrates how the connections between the adders are structured differently. In this technique, five adders with ten PPRs are added at a time. With one of the outputs from the prior CLA adder, the remaining PPR is added. In this method, the adder's size is altered in accordance with the shifted operation. At last, the output reaches the last adder. When comparing this strategy to the pipelined multipliers above, there is a reduction in both critical path time and power usage using pipe lined multipliers.

III. MODIFIED MULTIPLIER PIPELINING PROCESS

In terms of communication, the pipe lining technology stores data in registers, or pipes. The pipe lining process is depicted in Figure 1 as the output of one block connected to the input of the subsequent block distinct tiers of blocks independently validated and recorded the outcome using a register. The findings are kept for instant access, which lowers the design's power consumption and latency. This method is used in the construction of multiplier circuits. This 32-bit pipelined multiplier uses a linear pipeline pipe lining approach, which implies that no feedback loop is present in the design. Multiplicand and multiplier values are kept through the pipeline lining process for the adjusted multiplier at Stage 1. Utilized register for reduced partial product row values at stage 2. Store the partial product added values at the end. Here, the three clock cycles needed for the aforementioned three registers.

Previous pipelined multiplications A greater number of clock cycles results from the addition of registers at various stages. Thus, the routing and logic involved in this hardware implementation were complicated. We are now using higher frequencies, and the procedure requires three clock cycles. Functional verification was completed after a modified booth encoding multiplication method using a pipeline technique was created.

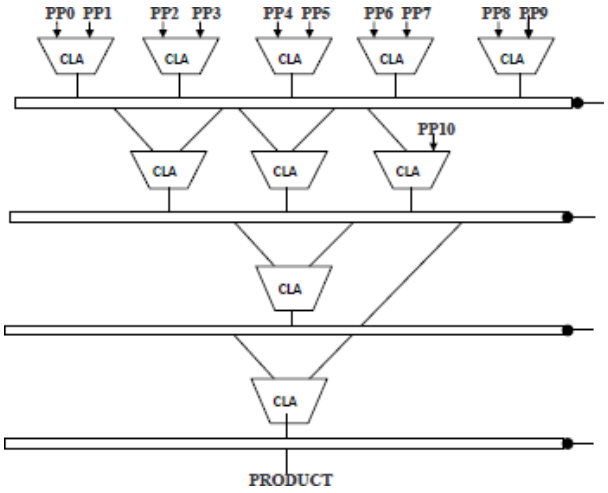

Figure 6: RADIX-8 Tree Based CLA Addition

IV. PERFORMANCE EVALUTION

Giving the modules to the tool, which synthesizes them into an RTL net list during the Design Synthesis process, is known as the multiplier design entry. If necessary, we can verify the net list's behavioral simulation; if not, we may put it into practice right away. The resulting net list is translated, mapped, placed, and routed onto the chosen FPGA device during the implementation phase. Static timing analysis and functional simulation are available after implementation. Timing simulation may be obtained by doing back annotation. We proceed with FPGA programming and use the Chip Scope Pro Analyzer to confirm that the FPGA is installed on the circuit if the results are as anticipated and achievable. Based on that, Figures 7, 8, and 9 depict the simulation, synthesis, and chip scope internal analysis of the FPGA (SPARTAN3E-FG320) output verification, respectively.

A. Design and Implementation Results:

Here totally 4 types of approaches of modified and existed delay results shown in below figures 10 with FPGA device technology report. The approaches are

- RADIX-4 sequential based pipelined multiplier
- RADIX-4 tree based pipelined multiplier
- RADIX-8 sequential based pipelined multiplier
- RADIX-8 tree based pipelined multiplier

Second one power consumption comparison between same 4 types of approaches reports are shown in below figures

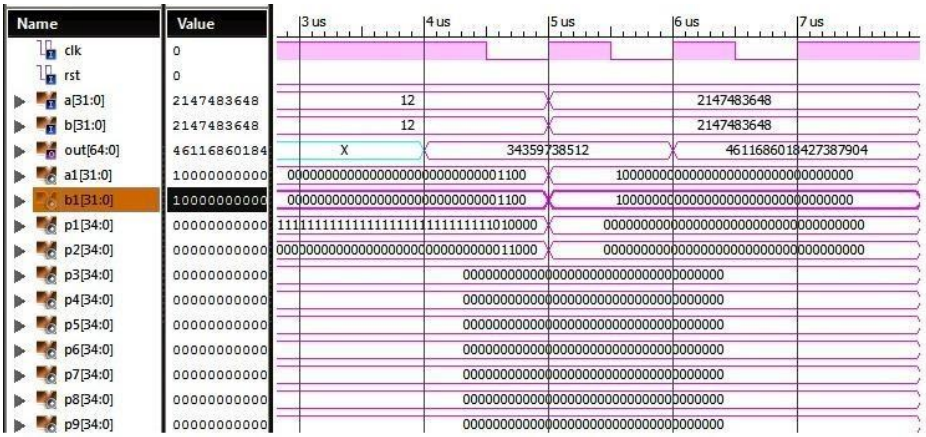

Figure 7: RADIX-8 32-bit Pipelined Multiplier Simulation Report

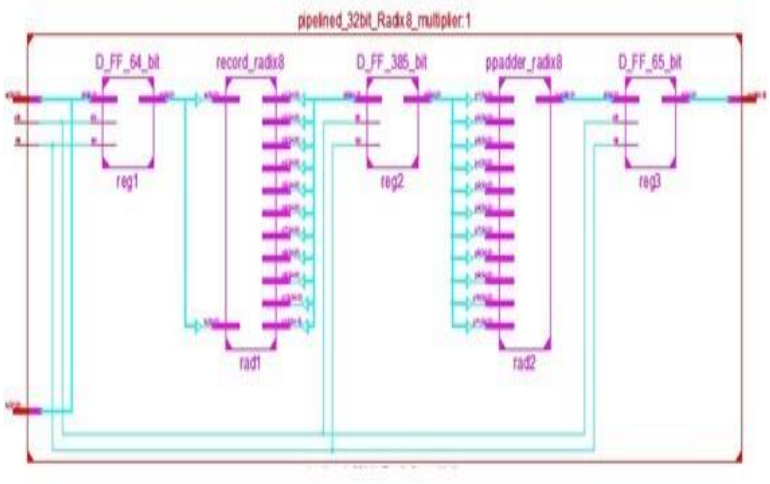

Figure 8: RTL Internal Diagram of RADIX-8 Pipelined Multiplier

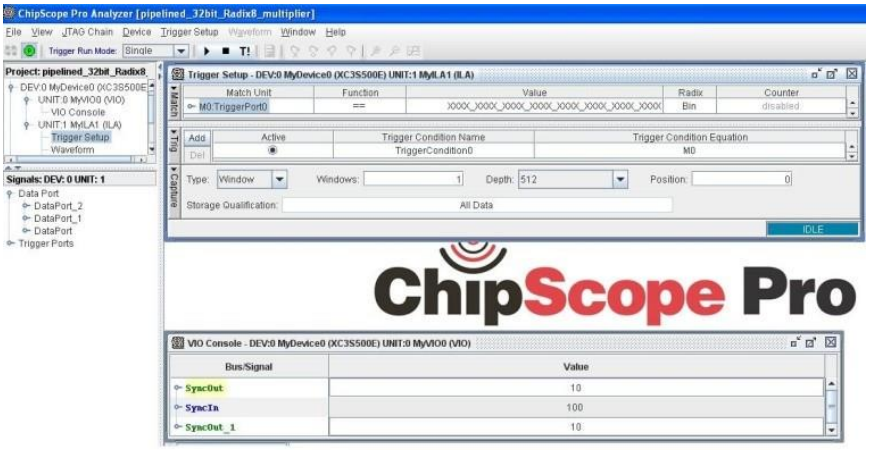

Figure 9: Chip Scope Pro Analysis for RADIX-8 Pipelined Multiplier

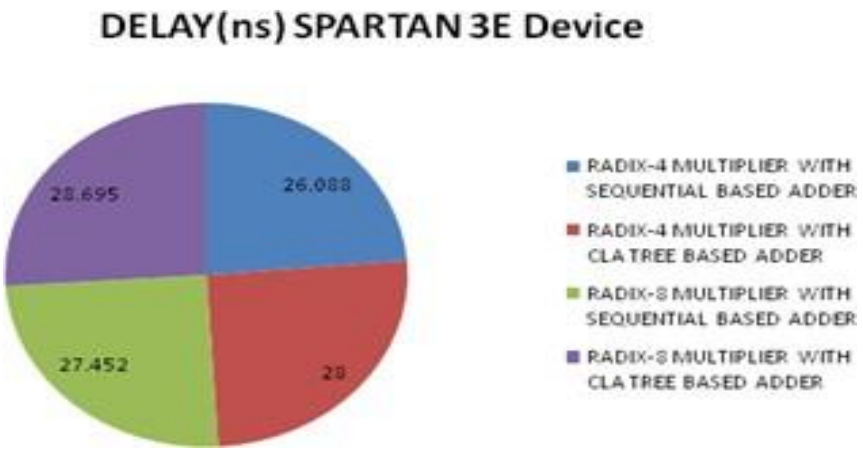

Figure 10: Delay Report of 4 Pipe Lined Multipliers

11, 12, 13Power consumption of multiplier design is analyzed by using below equation

$$P_d = \propto_t f_{clk} C_{load} Vdd^2 \qquad (2)$$

Power optimization is basic interest to implement modified design.

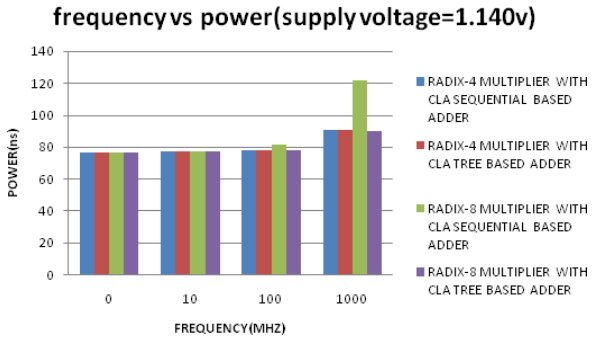

Figure 11: Comparison of Power Report between 4 Pipelined Multipliers (at1.140V)

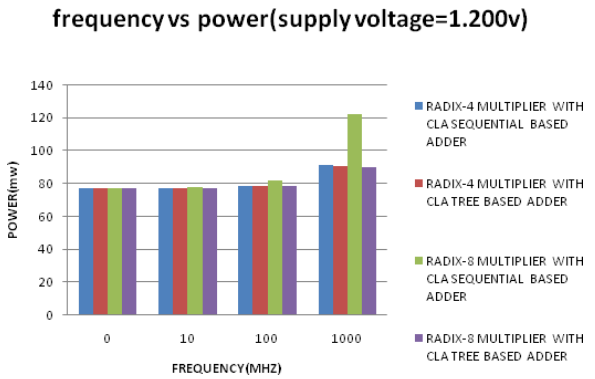

Figure 12: Comparison of Power Report between 4 Pipelined Multipliers (at1.200V)

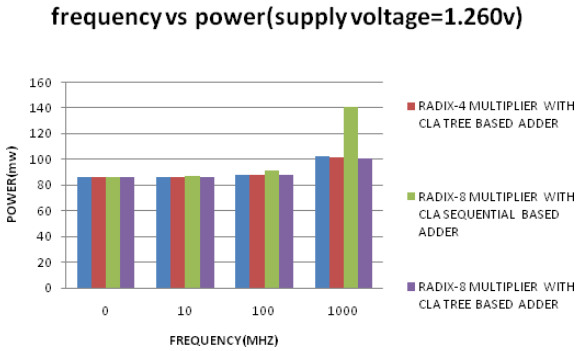

Figure 13: Comparison of Power Report between 4 Pipelined Multipliers (at1.260V)

Routing and logic decides the how much power RADIX-4 MBE generates 16 PPRs for 32-bit multiplier. From these results we finally concluded that RADIX-8 MBE CLA tree based adder is the best technique for large bits multiplication when power and delays are main required for the particular design. Final design power consumption is equal to power consumed by routing plus power consumed by logic. Tables 2,3,4,5 represent total power.

V. CONCLUSION

In this particular work, a 32-bit multiplier design implemented in RADIX-4 MBE with the help of [1] and RADIX-8 MBE PPRs generating methodology with sequential and tree carry look ahead adder based PPRs addition in a XILINX FPGA Device and also corresponding power analysis was performed. As per power and timing analysis reports at high frequency range example at 1000 MHz and supply voltage vccint=1.140v RADIX-4 MBE CLA TREE consumes 90.61mW and RADIX-8 MBE CLA consumes 89.72W, at high supply voltage=1.260v RADIX-4 MBE CLA TREE consumes 101.61mW, RADIX-8 MBE CLA consumes 100.54mW. RADIX-8 MBE generates 11 partial products and constraints.

VI. ACKNOWLEDGMENT

We would like to thank all the authors in the referencesfor providing great knowledge and helpful advices whenever required.

Table IV: 32-Bit Radix-8 Pipelined Multiplier with Sequential Adder Power Report

Frequency(Mhz)	Supply voltage(Vccint)	Power(mW)		
		Quiescent(logic)	Dynamic(Signals,I/O)	Total
0	1.140	77	0(0,0)	77
	1.200	81	0(0,0)	81
	1.260	86	0(0,0)	86
10	1.140	77	0.45(0.35,0.10)	77.45
	1.200	81	0.49(0.39,0.10)	81.49
	1.260	86	0.54(0.43,0.11)	86.54
100	1.140	77	4.46(3.51,0.95)	81.46
	1.200	81	4.89(3.89,1.00)	85.89
	1.260	86	5.34(4.29,1.05)	91.34
1000	1.140	77	44.61(35.10,9.51)	121.61
	1.200	81	48.90(38.89,10.01)	129.90
	1.260	86	53.39(42.88,10.51)	139.39

VII. REFERENCES

[1] SmrutiBokade, PravinDakhole, “CLA based 32-Bit Signed Pipelined Multiplier,” International Conference on Communication and Signal Processing, April 6-8, 2016, India.

[2] Abdullah-Al-Kafi, Atul Rahman, Bushra Mahjabeen, Mahmudur Rahman, “An Efficient Design Of FSM Based 32-Bit Unsigned High-Speed Pipelined Multiplier Using Verilog HDL” 8th International Conference on Electrical and Computer Engineering, DOI 10.1109/ICECE.2014.7027026, Dec2014.

[3] Qingzheng LI, Guixuan LIANG, Amine BERMAK, “A High Speed 32-Bit Signed/Unsigned Pipelined Multiplier,” Fifth IEEE International Symposium on Electronic Design, Test and Applications, DOI 10.1109/DELTA.2010.10, Jan2010

[4] Shiann-RongKuang, Jiun-Ping Wang, Cang-Yuan Guo, “Modified Booth Multipliers With a Regular Partial Product Array” IEEE Transactions on Circuits and Systems, DOI 10.1109/TCSII.2009.2019334, Vol56, Issue 5, May2009.

[5] Huang Z. J., Ercegovac M. D., Cater J, “High-performance low-power left-to-right array multiplier design” IEEE Transactions on Computers, DOI 10.1109/TC.2005.51, Vol 54, Issue 3, March 2005.

[6] Wen-Chang Yeh, Chein-Wei Jen, “High-Speed Booth Encoded Parallel Multiplier Design,” IEEE Transactions on Computers, DOI 10.1109/12.863039, Vol49, Issue 7, July 2000.

[7] Rahul D Kshirsagar, Aishwarya.E.V, AhireShashank Vishwanath, P Jayakrishnan, “Implementation of Pipelined Booth Encoded Wallace Tree Multiplier Architecture” 2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), DOI 10.1109/ICGCE.2013.6823428, Dec2013.

[8] Vijayalakshmi, R. Seshadri, Dr. S. Ramakrishnan, Design And Imple-mentation Of 32-Bit Unsigned Multiplier Using CLAA And CSLA International Conference on Emerging Trends in VLSI, Embedded System, Nano Electronics and Telecommunication System, DOI 10.1109/ICEVENT.2013.6496579, Jan2013.

[9] Soniya, Suresh Kumar, “A Review of Different Type of Multipliers and Muliplier Accumulator Unit”, International Journal of Emerging Trends and Technology in Computer Science, Vol. 2 No. 4, August 2013.

[10] J.A. Hidalgo, V. Moreno-Vergara, O. Oballe, A. Daza, M.J. Martn-Vzquez, A. Gago, “A RADIX-8 multiplier unit design for specific purpose,”.

[11] MULTIPLIERS, BOOTH MULTIPLIERS pp[13-21], http://users.encs.concordia.ca/asim/COEN 6501/Lecture Notes/L3 Notes.pdf.

[12] BOOTH MULTIPLIER, pp[3-8], http://vlsiip.com/download/booth.pdf