# Design and Analysis of RADIX-8 Based32-bit pipelined multiplier

**Mummareddy. Ramya Krishna**
**Department of electronics and Communication Engineering.**
**ISTS Women's Engineering College**
**Andhrapradesh, east gonagudem**
**mummareddyramyakrishna777@gmail.com**

**Prof. B. Ramana Kumar,**
**Associate professor,**
**Department of electronics and Communication Engineering.**
**ISTS Women's engineering college**
**Andhrapradesh, east gonagudem**

***Abstract*—Arithmetic operations play a crucial role in signal processing applications, micro processors, and micro controllers. Multiplication operation is widely used in Digital Signal Processing (DSP) modules like Infinite Impulse Response (IIR) filters, Finite Impulse Response (FIR) filters, Discrete Fourier Transform (DFT),Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT) and etc. Reducing the power dissipation and increasing the performance is the most important challenge for multiplier design. Power reduction techniques such as Recoding (RADIX- 8 Modified Booth Encoding (MBE)) structure has different logic combinations to generating Partial Product Rows (PPRs) which is used in modified pipelined multiplier. Two additionapproaches like tree based Carry Look-a-head Adder (CLA) and sequential based CLA are used for addition between generated PPRs. Performance comparison between existed RADIX-4 pipelined multiplier and RADIX-8 pipelined multiplier with aboveaddition approaches, the modified method yieldsconsiderablemoderatecritical path delay, area and reduced the power consumption. Modified design is implemented in XILINX 14.7withFPGAtechnologyXC3S500E-FG320-5.***

***Index Terms*—Pipelined multiplier, RADIX-8 MBE, partial products generation, sequential and tree based CLA addition.**

## I. INTRODUCTION

Advanced consumer electronics make wide use of Digital Signal Processing (DSP) providing accelerators for the do-mains of communications, general, military purpose systems. In DSP applications carry out a large number of arithmetic operations as their implementation based on computationally intensive kernels, such as Discrete Fourier Transform (DFT), Fast Fourier Transform (FFT), Infinite Impulse Response(IIR) and Finite Impulse Response (FIR). DSP system performance can be evaluated by the design allocation and architecture of arithmetic units. Recent research activities in the field of optimized arithmetic operations are grown up. Data transfer in different digital modules is done by the arithmetic operations (sub modules). One of the important sub modules is multiplier. Different multiplier designs were introduced to enhancing more efficient implementations of DSPalgorithms.Severalapproacheshavebeenproposedtooptimizetheperformanceofthemultiplieroperationintermsofarea, power consumption. For efficiently mapping of flexible DSP RADIX-8 MBE and generated partial products addition critical path synthesis.This can be done by placing of arithmetic units like multiplier Many DSP applications can be implemented based on multiplier operation.

Multiplier can be implemented as either combinational or pipelined multiplier in terms of hardware. Combinational multiplier implementation increases both hardware resources and critical path in two stages. At stage1, N*N number of inputs produced N number of PPRs. At stage2, number of adders increases for the design so combinational method is not efficient for large multiplier design. Pipelined multiplier overcomes the above disadvantages with recoding structure. Targeting the optimized design of pipelined multiplier unit, partial product reduction techniques and addition approaches are discussed in section2. PPRs generation and addition processes involved to design different multipliers with constraints are discussed below.

CLA based 32-Bit Signed Pipelined Multiplier is implemented using Verilog HDL and achieve efficient critical path delay and area but power consumption information is not given. This pipelined multiplier generates half (16) Partial Product Rows (PPRs) compare with normal multiplication process. It requires 15 CLA adders for PPRs addition [1]. Another approach is FSM based 32-bit high speed pipelined multiplier is implemented using verilog HDL which is also having efficient critical path delay but more than [1] and it generates 64-bit 32 partial products addition then it consumes more power [2]. Another design is 32-bit pipelined multiplier implemented using 180nm CMOS implementation it is also delay efficient but less than [1], [2] and not described about power consumption [3]. In this paper partial products are generated by using MBE and final PPRs addition is done by using CSA-CCA architecture. Next one has been proposed to generate partial product array USING MBE and post truncated method which is reduced by using Wallace tree structure. Final addition of the partial products array is done by using CLA [4]. Another approach implements array based multiplier in different methods and analyzes the power at different voltage

Completed by two approaches CLA tree based and CLA sequential based.

## II. MODIFIED PIPELINED MULTIPLIER ARCHITECTURE

Every multiplier design should be perform two operations one is partial products generation and second one is partial products addition. But modified design directly generates Partial Product Rows (PPRs) and Perform addition operation between them. In normal multiplier increases hardware of the design is discussed in section 1 then it should be increases both logic and routing size. If increases the design hardware then also increases critical path delay and power consumption of the design [8] [9].This affect is reduced by using RADIX approach [1]. Modified multiplier design reduces maximum PPRs with the help of RADIX algorithm. Delay and areaproblems overcome by another method which is pipeline approach represented in [1] [2] [7]. Different algorithms are involvedfor generation of partial products [1-10]. For example, BOOTH, Modified Booth Encoding (MBE), Wallace Tree, Bough Wooley algorithms improve the performance of multiplier in terms of area, power and delay. In these algorithms shifting (left or right shift) [5] operation of PPRs plays a withal role in the design. Modified design utilizes MBE algorithm forgeneration of PPRs. Next step is addition of PPRs which can be done by using different adders like Carry Look-a-head Adder (CLA), Ripple Carry Adder (RCA), Carry Save (CSA), Carry Select (CSLA) and Carry Skip Adders (CSPA). Adders also enhance the multiplier response. Advanced multiplier uses number of adders so different structures introduced for addition of PPRs [1]. Modified design also involved with two structures. They are 1. TREE based CLA addition 2. Sequential Based CLA addition Figure 1 represents step by step process of thedesign.

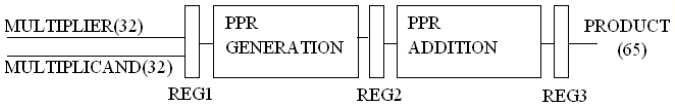


Fig. 1. Modified Multiplier Block Diagram

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 registerwith the help of cascaded 64 D-flip flops. Next important block is PPRsGeneration.

### A. PPRsGeneration

This block is important formed field 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 of the 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 ofRADIX- 8structure.

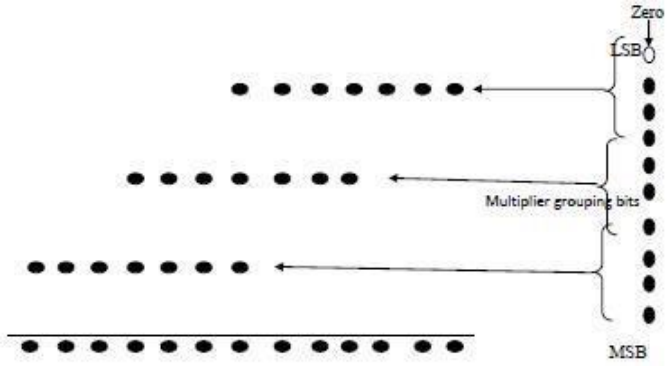


Fig.2. Dot Diagram Representation ofPPR Generation

*1) Algorithm of RADIX-8 structure:* Steps to generate PPRs using MBE Radix-8: -

Step 1: Multiplier 'Y' LSB is taken as '0' and remaining input is same.

Step 2: The bit group T, where T = 4 of the multiplier"Y", different combinations of each group Zk, where n-1 k 0. The rule for each Zk group is such that (Yi+2 Yi+1 Yi Yi-1), as shown in table 1.

Step 3: Now depending on K, where n-1 ¡= k ¡= 0, where Skis the value PPR in the Table for all possible combinations of torque values Zk.

step4: Add these PPRs based on multiplicand value to get thefinal product. Mathematicalexpression of RADIX-8 MBE IS

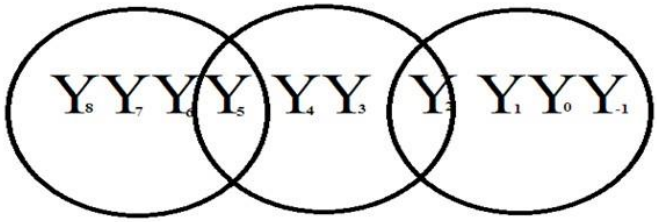


Fig.3. Multiplier Grouping Bits Using RADIX-8

Expressed as

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

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

TABLE I
RADIX-8 MODIFIED BOOTH ENCODING (MBE)

| RADIX-8 MODIFIED BOOTH ENCODING | | | | |
|---|---|---|---|---|
| INPUT BITS OF MULTIPLIER | | | | PARTIAL PRODUCT S |
| Y-1 | Y-2 | Y-3 | Y-4 | PPR |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | Same as multiplicand (X) |
| 0 | 0 | 1 | 0 | Same as multiplicand (X) |
| 0 | 0 | 1 | 1 | 1 Left shift of X |
| 0 | 1 | 0 | 0 | 1 Left shift of X |
| 0 | 1 | 0 | 1 | X+1 Left shift of X |
| 0 | 1 | 1 | 0 | X+1 Left shift of X |
| 0 | 1 | 1 | 1 | 2 Times left shift of X |
| 1 | 0 | 0 | 0 | 2 Left shifts X (2's complement of |
| 1 | 0 | 0 | 1 | X+(1 LEFT SHIFT OF ( X)) |
| 1 | 0 | 1 | 0 | X+(1 LEFT SHIFT OF ( X)) |
| 1 | 0 | 1 | 1 | 1 LEFT SHIFT OF ( X) |
| 1 | 1 | 0 | 0 | 1 LEFT SHIFT OF ( X) |
| 1 | 1 | 0 | 1 | X |
| 1 | 1 | 1 | 0 | X |
| 1 | 1 | 1 | 1 | 0 |

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 (Grouping four 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 are
11 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 fourdigits ofmultipliergrouprepresents1leftshiftofmultiplicandwhere LSB is '0' 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 multiplexer of 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 partialproductbitsaregeneratedthenthesepartialproducts

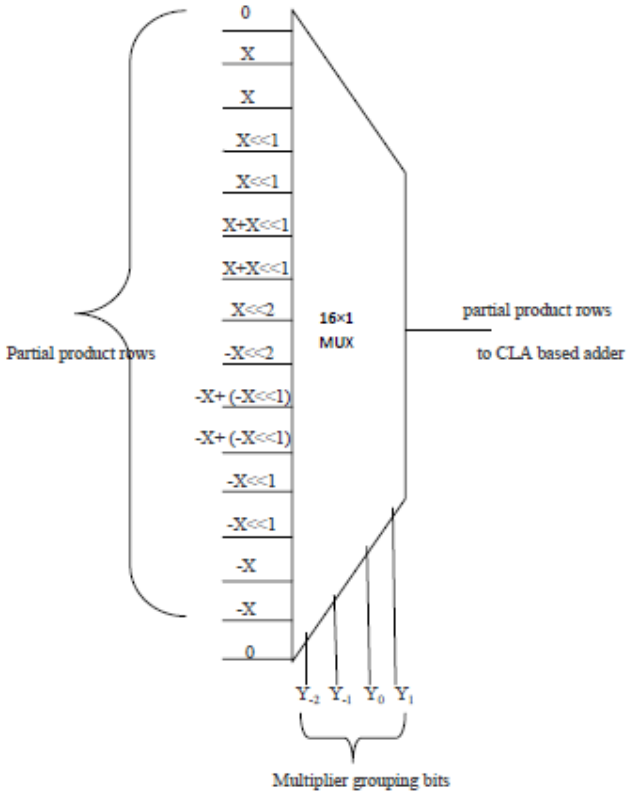


Fig. 4 : 16X1 MUX Selecting Operation For PPR

are stored with the help of 385 d-flip flop register. In RADIX-4 MBE, generate N/2 partial product rows for N*N multiplier. In RADIX-8 MBE reduce the partial products less than N /2 . so RADIX-8 MBE reduces the number of partial products compared with RADIX-4 MBE. Different complement and normal left shift operations performed based on represent in the RADIX-8 table 1.

*B. PPRsAddition*

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 i s
From above example PP0+PP1=20+80=100
000pp1+pp2000=00000000000000000000000000010100 + 00000000000000000000000001010000
00000000000000000000000001100100 (decimal value-100)
Above addition can be done by using CLA Adder with two different approaches 1.CLA Sequential based approach 2.CLA Tree basedapproach

*1) CLA Sequential based approach:* In this approach, adder size is constant because of output of one adder is given to theinput of next adder. Here present adder input is waiting for

Previous adder output due to increase the critical path delay and power constraints of the design. Those compared result values are shown in figures 10 to 13. . Total we have 11 PPRs and addition between them which is done by using 10 CLA adders as shown in figure 5.

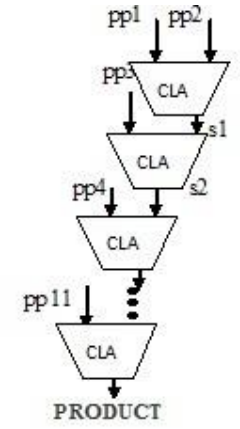


Fig.5. RADIX-8 Sequential based CLA addition

*2) CLA Tree based approach:* First approach disadvantage is overcome by the Tree based approach. Before going to addition process each PPR bits are shifted left to 3 times. First PPR is 35-bit and second one is left shift to 3 times (38-bit) with respect to previous one. Similarly next one is left shifted to 3 times with respect to previous one. This process repeats again and again up to final PPR.This addition process also utilized the same number of adders but the way of structure connection between adders is different shown in thefigure.At a time five adders with 10 PPRs are added in this process. Remaining PPR is added with one of the output of previous CLA adder. Size of the adder is changed based upon shifted operation in this approach. Finally output gets at the final adder.Critical path delay and power consumption is reduced in this approach pipe lined multipliers which is compared with above approach pipelinedmultipliers.

## III. MODIFIED MULTIPLIER PIPELININGPROCESS

Pipe lining technique is a process which already stored data in registers (pipes)in terms of communication. Process of pipe lining represents output of the on block is connected as the input of next block shown in figure 1.different levels of blocks individually verified and stored that result with the help of register. The stored results are access directly then it reduced the power consumption and delay of the design. This process is applied to multiplier design circuit. Linear pipeline pipe lining technique (that means no feedback path is available in the design) used in this particular 32-bit pipelined multiplier. Pipelining process for modified multiplier at Stage1, multiplicand and multiplier values stored. At stage2 used register for reducedpartialproductrowvalues.Infinalstage,storethepartial product added values. Here we required three clock cycles for above three registers. Previous pipelinedmultiplications

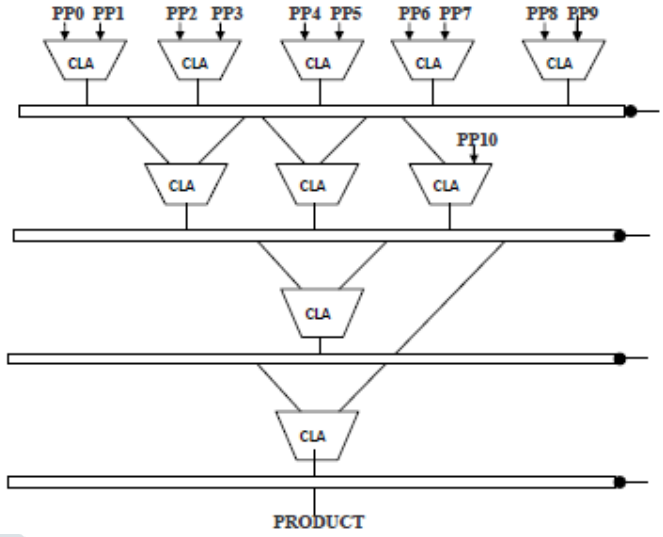


Fig.6. RADIX-8 Tree Based CLA Addition

Registers are added at different stages then it increased number of clock cycles. So here hardware implementation in terms of routing and logic were complex. Now we implemented higher frequencies and 3 clock cycles are needed for the operation. Modified booth encoding multiplication process with pipeline approach was synthesized and functional verification is done.

## IV. PERFORMANCEEVALUTION

The multiplier design entry means giving the modules to the tool, those modules are synthesized into RTL net list in Design Synthesis process. If it needed we can check the behavioral Simulation of net list if not we can directly implement it. Implementation is a process of Translating, Mapping and Placing and Routing the generated net list on to the FPGA selected device. After implementation we can get theFunctionalsimulation,Statictiminganalysis.Ifwedoback Annotation we can get Timing simulation. If the results we get are expected and can be realized then we go for Programming of FPGA and we can verify that FPGA on the circuit by Chip Scope Pro Analyzer. Based on that Simulation, synthesis and chip scope internal analysis of FPGA (SPARTAN3E-FG320) output verification is shown in the figure 7, 8 and 9 respectively.

*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

1. RADIX-4 sequential based pipelined multiplier
2.RADIX-4 tree based pipelined multiplier
3.RADIX-8 sequential based pipelined multiplier
4.RADIX-8 tree based pipelined multiplier

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

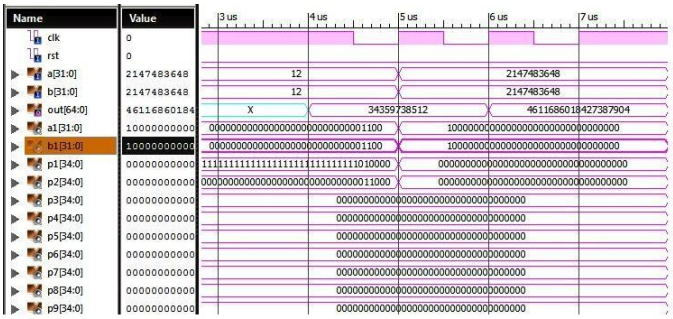


Fig.7. RADIX-8 32-bit Pipelined Multiplier simulation report

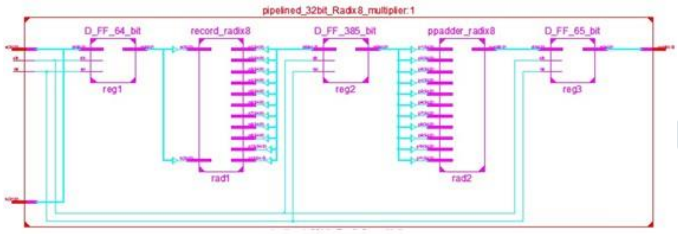


Fig.8. RTL Internal Diagram of RADIX-8 Pipelined Multiplier

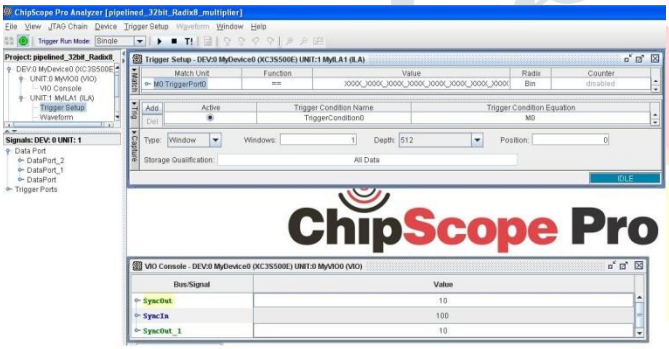


Fig.9. CHIP SCOPE PRO ANALYSIS For RADIX-8 Pipe lined Multiplier

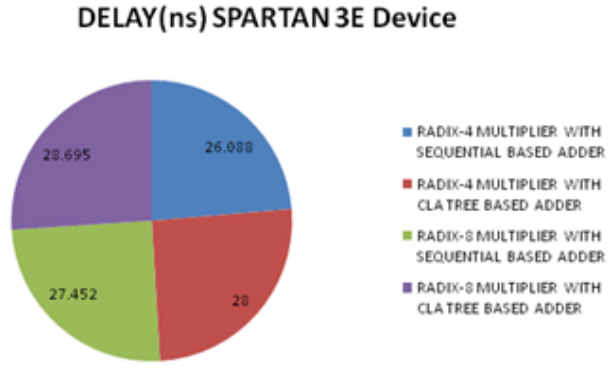


Fig.10. Delay Report of 4 Pipe lined Multipliers

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

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

Power optimization is basic interest to implement modified

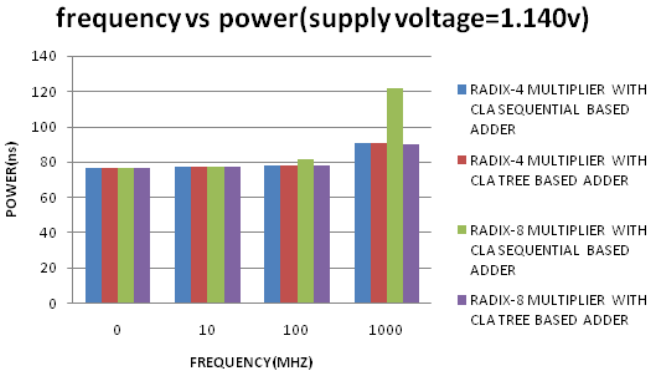


Fig.11. Comparison of power Report between 4 Pipe lined Multipliers (at 1.140V)

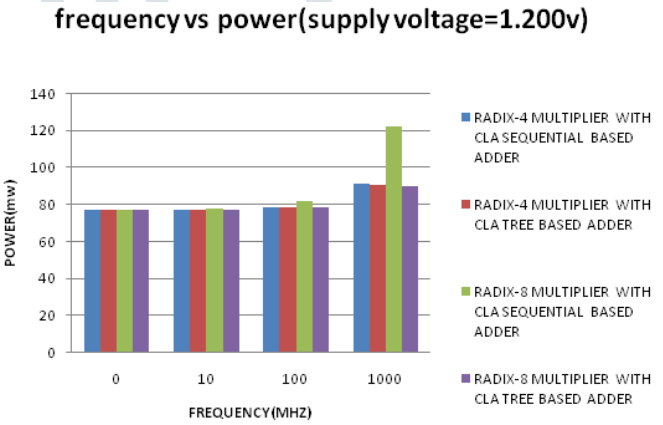


Fig.12. Comparison of power Report between 4 Pipe lined Multipliers (at 1.200V)

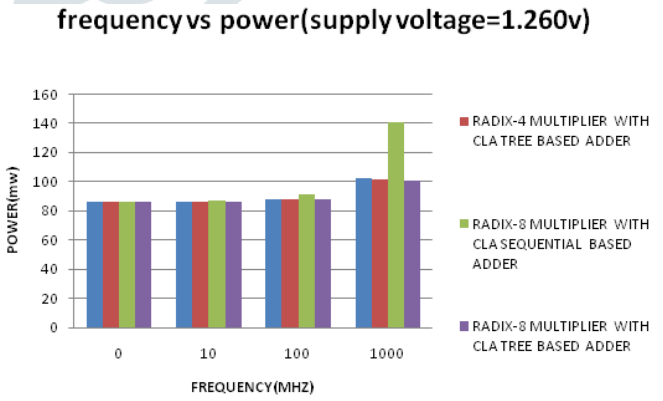


Fig.13. Comparison of power Report between 4 pipelined Multipliers (at 1.260V)

design. Routing and logic decides the how much power

TABLE II
32-BIT RADIX-4 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 | 0.15(0.06,0.09) | 77.15 |
| | 1.200 | 81 | 0.15(0.06,0.09) | 81.15 |
| | 1.260 | 86 | 0.16(0.06,0.10) | 86.16 |
| 100 | 1.140 | 77 | 1.39(0.52,0.87) | 78.39 |
| | 1.200 | 81 | 1.49(0.58,0.91) | 82.49 |
| | 1.260 | 86 | 1.60(0.64,0.96) | 87.60 |
| 1000 | 1.140 | 77 | 13.87(5.21,8.66) | 90.87 |
| | 1.200 | 81 | 14.90(5.78,9.12) | 95.90 |
| | 1.260 | 86 | 15.94(6.37,9.57) | 101.94 |

TABLE III
32-BIT RADIX-4 PIPELINED MULTIPLIER WITH TREE 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 | 0.14(0.05,0.09) | 77.14 |
| | 1.200 | 81 | 0.14(0.05,0.09) | 81.14 |
| | 1.260 | 86 | 0.16(0.06,0.10) | 86.16 |
| 100 | 1.140 | 77 | 1.36(0.49,0.87) | 78.36 |
| | 1.200 | 81 | 1.46(0.55,0.91) | 82.46 |
| | 1.260 | 86 | 1.56(0.60,0.96) | 87.56 |
| 1000 | 1.140 | 77 | 13.61(4.95,8.66) | 90.61 |
| | 1.200 | 81 | 14.60(5.48,9.12) | 95.60 |
| | 1.260 | 86 | 15.61(6.04,9.57) | 101.61 |

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

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 |

TABLE V
32-BIT RADIX-8 PIPELINED MULTIPLIER WITH TREE 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 | 0.13(0.04,0.09) | 77.13 |
| | 1.200 | 81 | 0.14(0.05,0.09) | 81.14 |
| | 1.260 | 86 | 0.15(0.05,0.10) | 86.15 |
| 100 | 1.140 | 77 | 1.28(0.41,0.87) | 78.28 |
| | 1.200 | 81 | 1.36(0.45,0.91) | 82.36 |
| | 1.260 | 86 | 1.46(0.50,0.96) | 87.46 |
| 1000 | 1.140 | 77 | 12.72(4.06,8.66) | 89.72 |
| | 1.200 | 81 | 13.62(4.50,9.12) | 94.62 |
| | 1.260 | 86 | 14.54(4.97,9.57) | 100.54 |

RADIX-4 MBE generates 16 PPRs for 32-bit multiplier. From these results We finally concluded that RADIX-8 MBE CLAtree based adder is the best technique for large bits multiplication when power and delays are main constraints.

## ACKNOWLEDGMENT

We would like to thank all the authors in the references for providing great knowledge and helpful advices when ever required.

## REFERENCES

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

[2] Abdullah-Al-Kafi, AtulRahman, BushraMahjabeen, MahmudurRahman,"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 Par- allel 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, Embed- ded 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 pur- pose," .
[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