International Journal of Engineering Technology and Applied Science

(ISSN: 2395 3853), Vol. 4 Issue 3 March 2018

Area Delay and Power Efficient 8-bit booth multiplier Design using pre encoded 4-bit radix of Design

[1]Pooja Jain, [2]Dilip Ahirwar, [3]Anupreksha Jain
[1]M-Tech Scholar, [23]Assistant Professor,
Department of ECE, Adina Institute of Science & Technology, Sagar, India

Abstract: - The Booth multiplier has been widely used for high performance signed multiplication by encoding and thereby reducing the number of partial products. A multiplier using the radix-4 (or modified Booth) algorithm is very efficient due to the ease of partial product generation, whereas the radix-8 Booth multiplier is slow due to the complexity of generating the odd multiples of the multiplicand. In this paper, this issue is alleviated by the application of approximate designs. An approximate 2-bit adder is deliberately designed for calculating the sum of 1× and 2× of a binary number. This adder requires a small area, a low power and a short critical path delay. Subsequently, the 2-bit adder is employed to implement the less significant section of a recoding adder for generating the triple multiplicand with no carry propagation. In the pursuit of a trade-off between accuracy and power consumption, two signed 16×16 bit approximate radix-8 Booth multipliers are designed using the approximate recoding adder with and without the truncation of a number of less significant bits in the partial products. The proposed approximate multipliers are faster and more power efficient than the accurate Booth multiplier, moreover, the multiplier with 15-bit truncation achieves the best overall performance in terms of hardware and accuracy when compared to other approximate Booth multiplier designs.

Keywords: Radix-8, Radix-4, Booth Multiplier, Wallace Tree, CSA, RCA.

I INTRODUCTION

Day by day IC technology is getting more complex in terms of design and its performance analysis. A faster design with lower power consumption and smaller area is implicit to the modern electronic designs. Multipliers generally have extended latency, huge area and consume substantial amount of power. Hence low-power multiplier factor style has become a very important region in VLSI system style. Everyday new approaches are being developed to style low-power multipliers at technological, physical, circuit and logic levels. Since the multiplier is usually the slowest component in an exceedingly system, the system's performance is decided by performance of the multiplier. Also multipliers are the most area consuming entity in a design. Therefore, optimizing speed and area of a multiplier is a major design issue nowadays. However, area and speed are usually conflicting constraints so that improving speed results in larger areas and vice-versa. Also area and power consumption of a circuit are linearly correlated. So a compromise has to be done in speed of the circuit for a greater improvement in reduction of area and power.

A higher representation radix effectively indicates to fewer digits. Thus, a single-digit multiplication formula necessitates fewer cycles as we have a tendency to begin moving to a lot of higher radices that mechanically results in a lesser range of partial product. Many algorithms are developed for this purpose like Booth's formula, Wallace Tree technique etc. For the summation method many adder architectures are accessible viz. Ripple Carry Addition, Carry Look-ahead Addition, Carry Save Addition etc. But to reduce the power consumption the summation architecture of the multiplier should be carefully chosen.

Rest of paper organized as follows: In section 2 discusses the system model and related work discuss in the section 3. In section 4 discusses the problem statement and section 5 discusses the proposed methodology and simulation results discusses in the section 6, last but not the least conclusion of this paper discuss in the section 7.

II SYSTEM MODEL

Multiplying a variable by a group of identified constant coefficients may be a common operation in several digital signal process (DSP) algorithms. Compared to alternative common operations in DSP algorithms, like addition, subtraction, exploiting delay components, etc., multiplication is usually the foremost costly. There is a trade-off between the amount of logic resources used (i.e. the amount of silicon in the integrated circuit) and how fast the computation can be done. Compared to most of the other operations, multiplication requires more time given the same amount of logic resources and it requires more logic resources under the constraint that each operation must be completed within the same amount of time.

A general multiplier is needed if one performs multiplication between two arbitrary variables. However, when multiplying by a known constant, we can exploit the properties of binary multiplication in order to obtain a less expensive logic circuit that is functionally equivalent to simply affirming the constant on one input of a standard multiplier. In many cases, using a cheaper implementation

for only multiplication still results in significant savings when considering the entire logic circuit because multiplication is relatively expensive. Furthermore, multiplication might be the vital operation, based on the application present.

A) Basic binary multiplier

The operation of multiplication is rather simple in digital electronics.

10×8 = 80	-6×4 = -24
1 0 1 0	1 0 1 0
1 0 0 0	0 1 0 0
0 0 0 0	0 0 0 0
0 0 0 0	0 0 0 0
0 0 0 0	1 1 1 0 1 0
1 0 1 0	0 0 0 0 0
1 0 1 0 0 0 0	1 1 1 0 1 0 0 0

It has its origin from the classical algorithm for the product of two binary numbers. This algorithm uses addition and shift left operations to calculate the product of two numbers. Booth's Multiplication Algorithm Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. The rule was fabricated by Andrew Donald Booth in 1950 during doing analysis on crystallography at Birkbeck faculty in Bloomsbury, London. Booth used table calculators that were quicker at shifting than adding and created the rule to extend their speed.

B) A Typical Implementation

Booth's algorithmic program is enforced by repeatedly adding (with standard unsigned binary addition) one in every of 2 preset values A and S to a product P, then employing a rightward arithmetic shift on P. Let m and r be the number and multiplier, respectively; and let x and y represent the quantity of bits in m and r.

1) Determine the values of A and S, and the initial value of P. All of these numbers should have a length equal to (x + y + 1).
   a) A: Fill the most significant (leftmost) bits with the value of m. Fill the remaining (y + 1) bits with zeros.
   b) S: S: Fill the foremost vital bits with the worth of (-m) in two's complement notation. Fill the remaining (y + 1) bits with zeros.
   c) P: Fill the foremost vital x bits with zeros. To the proper of this, append the worth of r. Fill the smallest amount vital (rightmost) bit with a zero.
2) Verify the 2 least vital (rightmost) bits of P.
   a) If they're 01, notice the worth of P + A. Ignore any overflow.
   b) If they're 10, notice the worth of P + S. Ignore any overflow.
   c) If they're 00, do nothing. Use P directly in the next step.
   d) If they are 11, do nothing. Use P directly in the next step.
3) Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let P now equal this new value.
4) Repeat steps two and three till they are been done y times.
5) Drop the smallest amount important (rightmost) bit from P. This is often the product of m and r.

The representations of the number and products don't seem to be specified; usually, these are both additionally in two's complement illustration, just like the multiplier factor, however any mathematical notation that supports addition and subtraction can work still. As explicit here, the order of the steps isn't determined. Typically, it yield from LSB to MSB, beginning at i = 0; the multiplication by 2i is then usually replaced by progressive shifting of the P accumulator to the right between steps; low bits are often shifted out, and consequent additions and subtractions will then be done simply on the greatest N bits.

III RELATED WORK

[1] K. Tsoumanis and K. Pekmestzi,et al. In this work, we introduce an architecture of pre-encoded multipliers for digital signal processing applications based on off-line encoding of coefficients. To this extend, the Non-Redundant radix-4 Signed-Digit (NR4SD) encoding technique, which uses the digit values $lbrace -1, 0,+1,+2 rbrace$ or $lbrace -2,-1,0,+1 rbrace$, is proposed leading to a multiplier design with less complex partial products implementation. Extensive experimental analysis verifies that the planned pre-encoded NR4SD multipliers, as well as the coefficients memory, are additional region and power efficiency than the traditional altered Booth theme.

[2] Dinesh, V and M. Kathirvelu et al. Multipliers are key parts of the many high performance systems like FIR filters, microprocessors, digital signal processors, etc. A system's performance is generally determined by the performance of the multiplier as the multiplier is generally the slowest element in the system. The analysis of performance parameters of different multiplier logics is essential for design of a system intended for a specific function with constraints on Power, Area and Delay. The work presents a detailed analysis of all the serial-parallel and parallel architectures. The multipliers are designed for 4 bit multiplication using DSCH tool and the corresponding layouts are obtained using Micro wind 3.5

tool using 45nm technology. From the analysis it's discovered that the array multipliers give a general routing structure which can be optimum for FPGA based mostly systems. Among the tree based mostly multipliers Dadda multipliers have a small advantage over Wallace tree multipliers in terms of performance. The altered booth multiplier factor is relatively inefficient for bits lesser than or up to four, because of the enhanced space concerned for realization of the booth encoder and booth selector blocks. The analysis shows that for lower order bits Dadda reduction is the most efficient.

[3] N. G. NikDaud and M. S. Badruddin et al. One of the effective ways to speed up multiplication are by reducing the number of partial products and accelerating the accumulation. In this work, a new architecture of hybrid Modified Booth Encoded Algorithm (MBE) and Carry Save Adder (CSA) is developed as fast multiplier architecture. Altera Quartus II platform is used to run the simulation. The architecture design is programmed into FPGA using Altera DE2 board to verify the synthesizability on physical hardware. This hybrid fast multiplier delivers good performance in term of higher speed as well as in term of less usage of logic elements.

[4] S. Nithya and M. N. V. Nithya et al. They implement a high speed and low power FIR digital filter style employing the constant breadth booth multiplier. To scale back the miscalculation in constant breadth multiplier reconciling probability calculator is employed (ACPE). To realize higher speed, the altered Booth encryption has been used and conjointly to fast up the addition the carry look ahead adder is employed as a carry propagate adder. The multiplier circuit is intended employing VERILOG and synthesized exploitation Xilinx ISE9.2i machine. The area, power and delay of the designed filter is analyzed exploitation cadence tool.

[5] C. Xiaoping and W. Shumin et al. The radix-4 Booth encryption or altered Booth encryption (MBE) has been very much adopted in partial merchandise generator to style high-speed redundant binary (RB) multipliers. Attributable to the existence of the error-correcting word (ECW) generated by MBE and RB encryption, the RB multiplier generates a further RB partial product rows. An additional RB partial product accumulator (RBPPA) stage is required for 2n-b RB MBE multiplier. The upper radix Booth formula than radix-4 is adopted to scale back the amount of partial merchandise. However, the Booth encryption isn't economical due to the problem in generating laborious multiples. The laborious multiples drawback in RB multiplier is resolved by distinction of 2 easy power-of-two multiples. This work presents a brand new radix-16 RB Booth encryption (RBBE-4) to avoid the laborious multiple of high-radix Booth encryption while not acquisition any ECW. The planned technique results in build high-speed and low-power RB multipliers. The experimental results show that the planned RBBE-4 multiplier achieves vital improvement in delay and power consumption compared with the RB MBE multiplier and also the current accorded best RBBE-4 multipliers.

[6] R. P. Rajput and M. N. S. Swamy et al. This work presents the planning and implementation of signed-unsigned altered Booth encryption (SUMBE) multiplier factor. This altered Booth encryption (MBE) multiplier and therefore the Baugh-Wooley multiplier performs multiplication operation on signed numbers solely. The array multiplier and Braun array multipliers perform multiplication operation on unsigned numbers solely. Thus, the necessity of the new automatic data processing system may be a dedicated and really high speed novel multiplier unit for signed and unsigned numbers. Therefore, this work presents the planning and implementation of SUMBE multiplier. The altered Booth Encoder circuit generates half the partial merchandise in parallel. By extending sign bit of the operands and generating an extra partial product the SUMBE multiplier is obtained. The Carry Save Adder (CSA) tree and therefore the final Carry Look ahead (CLA) adder accustomed speed up the multiplier operation. Since signed and unsigned multiplication operation is performed by constant multiplier unit the desired hardware and therefore the chip space reduces and this successively reduces power dissipation and price of a system.

[7] B. C. Paul and M. Okajima et al. They tend to provide a ROM-based sixteen times sixteen multiplier for low-power applications. the planning uses sixteen four times four ROM-based multiplier blocks followed by carry-save adders and a final carry-select adder (all ROM-based) to get the thirty two bit output. All memory blocks are enforced employing single semiconductor memory cells and eliminating identical rows and columns for optimizing the facility and performance. Measuring ends up in 0.18 mum CMOS method show a 400th reduction in power over the standard carry-save array multiplier once operated at its highest frequency. The ROM-based style additionally provides four hundred and forty yards less delay than the array multiplier with a minimal increase (7.7%) in power. This demonstrates the low-power operation of the ROM-based multiplier additionally at higher frequencies.

IV PROBLEM STATEMENT

One of the numerous solutions of realizing high speed multipliers is enhancing similarity that helps in decreasing the amount of consequent calculation levels. The initial version of Booth algorithmic program (Radix-4) had 2 explicit drawbacks. They were:

- The range of add-subtract operations and shift operations become variable and causes inconvenience in realizing parallel multipliers.

- The rule becomes inefficient once there are isolated 1's.

These issues are overpowered by employing altered Radix4 Booth algorithmic program that scans strings of 3 bits employing the algorithmic program given below:

1. Lengthen the sign bit 1 position if necessary to ensure that n is even.
2. Add a 0 to the right of the LSB of the multiplier.

Corresponding to the value of each vector, each Partial Product will be 0, +M, -M, +2M or -2M.

V PROPOSED METHODOLOGY

We used XILINX 14.1 platform to write our programs. All the RTL simulations have been done using this software only. Also for delay report the synthesis tool embedded in Xilinx was used. It takes HDL designs and synthesizes them to gate-level net-lists. Also it supports both Verilog and VHDL.

The basic building block for multiplication circuit that consist of four process module i.e. booth encoding, partial product summation, final addition and their accumulation shown in the figure 1.

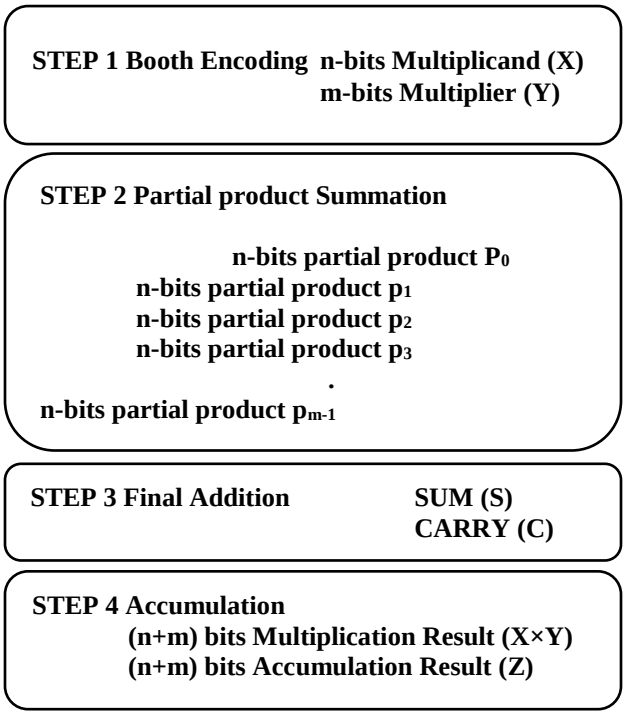

Figure 1: Basic Building Blocks for Multiplication Circuit

In booth encoding define the number of bits of multiplicands and number of bits of multiplier and then partial product has been done. The summation of these partial product has done in very next process and then summation of these partial product has been done in process of final addition process and at the last result will store in the accumulator.

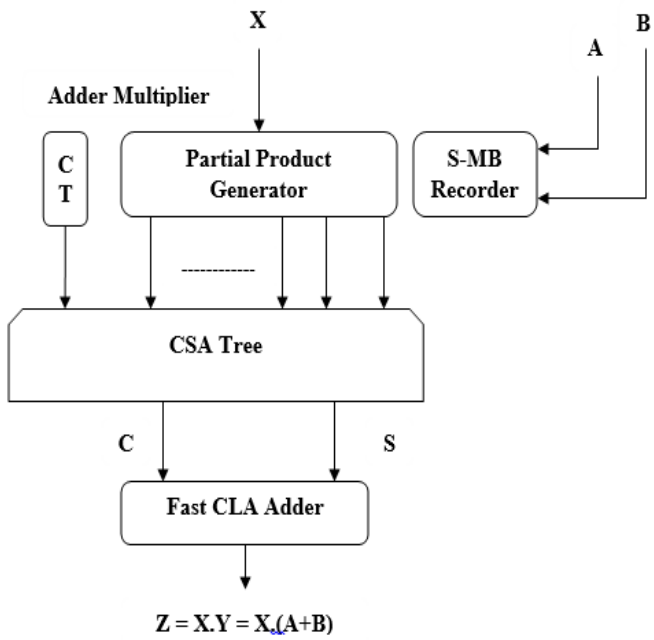

Figure 2: Fused design with direct recoding of the sum of A and B in its MB representation

The step by step process of basic building block for multiplication circuit shown in the figure 2.

Step 1: Process of booth encoding in which n-bits of multiplicand (X) and m-bits of multiplier (Y) are involved. Product has done in this process.

Step 2: Process of partial product summation, in this process partial product summation has done by in shifting sequence.

Step 3: Process of final addition, in this process addition has done for all sequences which have generated through partial product summation process.

Step 4: Process of accumulation, in this process shows the results of multiplication and store into the memory.

These steps of basic building block for multiplication circuit describe that the basic process multiplication.

VI SIMULATION RESULTS

In this section discuss about the plan and reproduction of proposed model of 8 bit both multiplier using pre encoded 4-bit radix of design for the handling of diminishment of power utilization. The outline information driven mimic in Xilinx programming. The Xilinx programming version is 14.1.

The numerous sections of the proposed designed has been simulated and the separate results has been shown on GUI.

Booth Multiplier Experimetal Analysis:

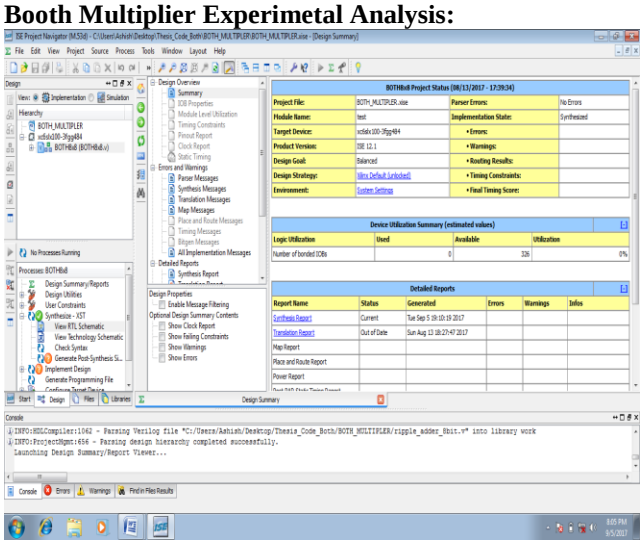

Figure 3: window show that the starting view of our implementation.

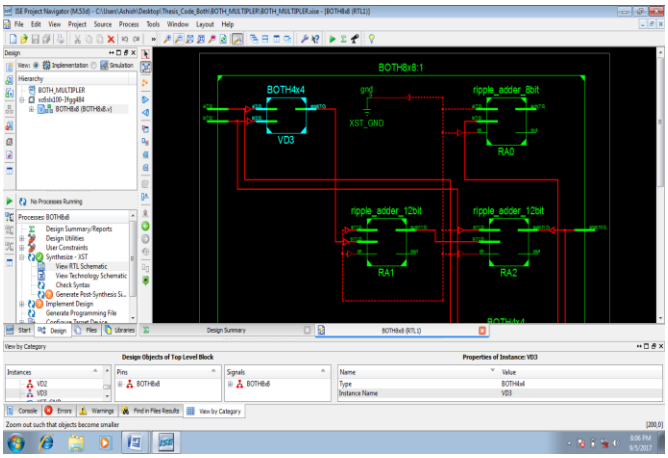

Figure 5: window show that the RTL schematic view with zoom in our implementation after the run the program.

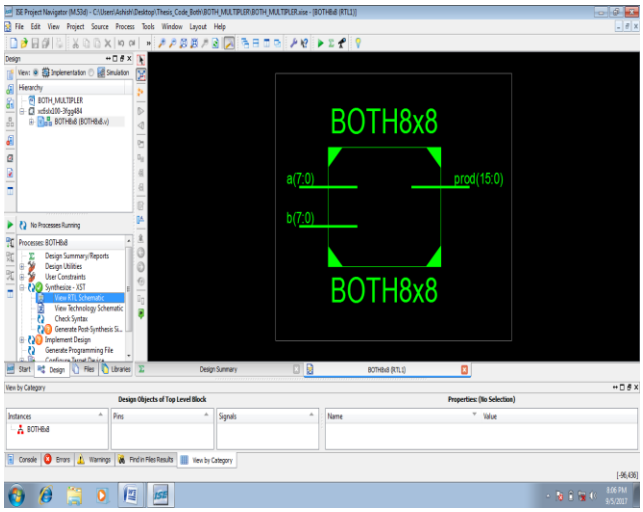

Figure 4: window show that the RTL schematic view in our implementation after the run the program.

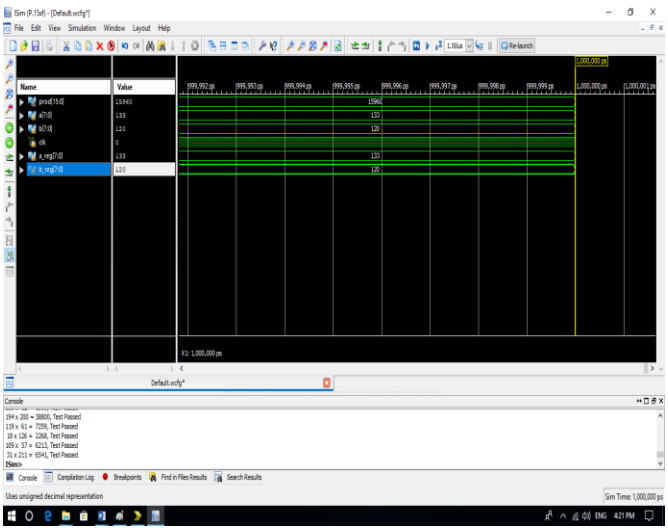

Figure 6: Test-bench Waveform generation using Xilinx-I-sim

Comparative Result Analysis

Table 1: Shows that the Data Driven input variance to the clock edge data-1110.

LOCAL UTILIZATION	Number Of 4 Input LUTS	Number of Occupied Files	Number of Slices Containing Only Related Logic	Number of Slices Containing Only Unrelated Logic	Number of Bonded
DFD (%)	17	3	97	2	11
RDFD (%)	20	5	98	4	14
USED	268	210	210	0	13
AVAILABLE	1536	768	210	210	124
UTILIZATION (%)	17	27	100	0	10

Above table 1 mainly based on Number of 4 Input LUTS. DFD (%), RDFD (%), Utilization value of used and available parameter for the experimental process in a simulation model.

Table 2: Shows that the Data Driven input variance to the clock edge data-1101.

LOCAL UTILIZATION	Number of 6 Input LUTs	Number of Occupied files	Number of Slices Containing only related logic	Number of Slices Containing only unrelated logic	Number of Bonded
DFD (%)	17	31	98	1	14
RDFD (%)	19	34	100	3	17
USED	324	260	260	0	19
Available	1864	838	260	260	138
Utilization (%)	21	33	100	0	16

Above table 2 mainly based on Number of 6 Input LUTS.DFD (%), RDFD (%), Utilization value of used and available parameter for the experimental process in a simulation model.

Table 3: Shows that the resultant of delay, area and power using NR4SD-, NR4SD+ and Booth method.

Parameter / Method	Delay	Area	Power
NR4SD-	1.95	18357	11.00
NR4SD+	1.95	18485	11.10
BOTH	1.70	18311	10.90

Table 4: Shows that the resultant of delay, area and power using NR4SD-, NR4SD+ and Booth method.

Parameter / Method	Delay	Area	Power
NR4SD-	1.98	19556	13.05
NR4SD+	1.98	19772	12.11
BOTH	1.80	19300	11.05

Performance Result Evaluation:

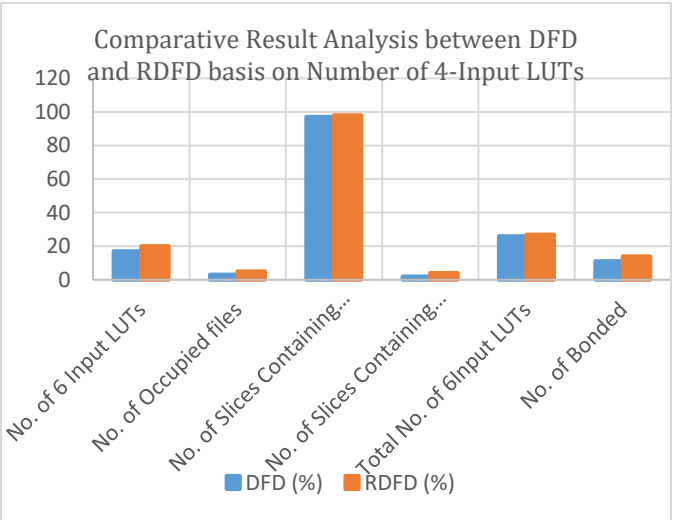

Figure 7: shows that comprative result analysis of Number of Occupied Files, Number of Slices Containing Only Related Logic, Number of Slices Containing Only Unrelated Logic,Total Number Of 4input LUTS, Number of Bonded on the basis of Number of 4-Input LUTs and clock edge data (1110) for DFD, RDFD, Used, Available and Utilization in Logic Circuit.

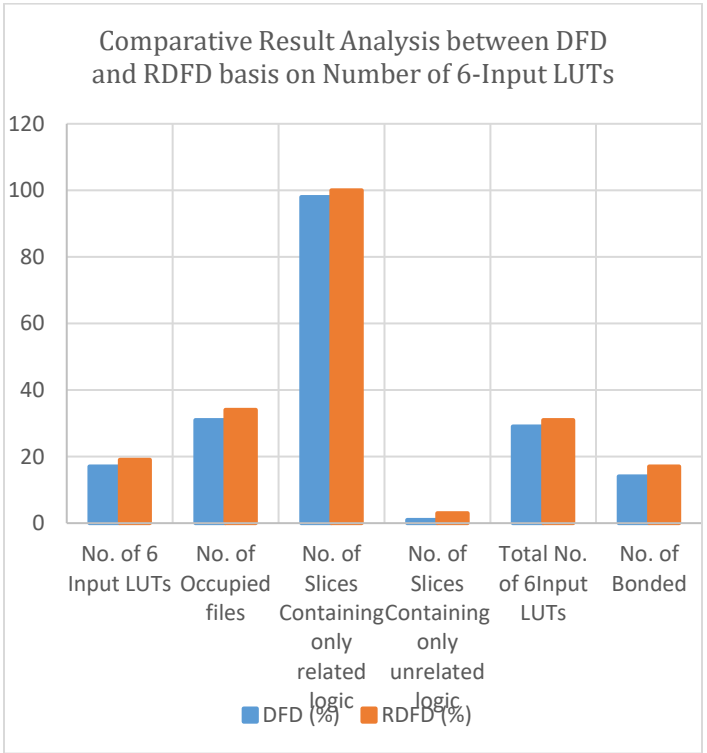

Figure 8: shows that comprative result analysis of Number of Occupied Files, Number of Slices Containing Only Related Logic, Number of Slices Containing Only Unrelated Logic,Total Number Of 4input LUTS, Number

of Bonded on the basis of Number of 6-Input LUTs and clock edge data (1101) for DFD, RDFD, Used, Available and Utilization in Logic Circuit.

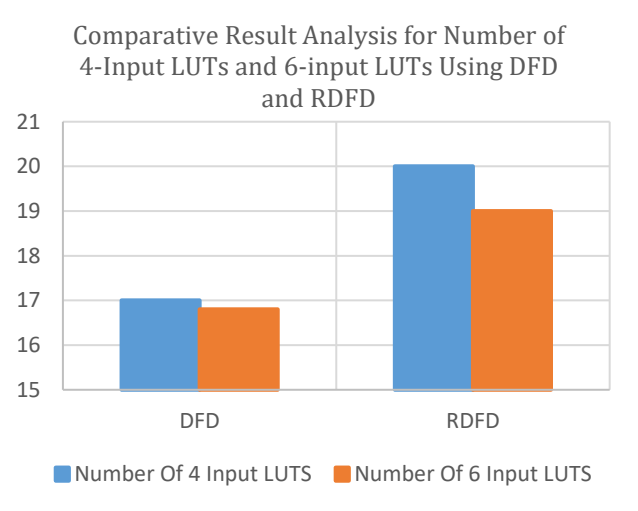

Figure 9: shows that comprative result analysis between DFD and RDFD forNumber of 4-Input LUTs and Number of 6-Input LUTs in Logic Circuit.

VII CONCLUSION

In this paper, new designs of pre-encoded multipliers are explored by off-line encoding the standard coefficients and storing them in system memory. We propose encoding these coefficients in the radix-8 booth multiplier using Non-Redundant radix-4 form. The proposed radix-8 booth multiplier using Non-Redundant radix-4 multiplier designs are more area and power efficient compared to the conventional and pre-encoded MB designs. Extensive experimental analysis verifies the gains of the proposed radix-8 modified booth multipliers in terms of area complexity and power consumption compared to the conventional MB multiplier.

REFERENCES

[1] K. Tsoumanis, N. Axelos, N. Moschopoulos, G. Zervakis and K. Pekmestzi, "Pre-Encoded Multipliers Based on Non-Redundant Radix-4 Signed-Digit Encoding," in IEEE Transactions on Computers, vol. 65, no. 2, pp. 670-676, Feb. 1 2016.

[2] B. Dinesh, V. Venkateshwaran, P. Kavinmalar and M. Kathirvelu, "Comparison of regular and tree based multiplier architectures with modified booth encoding for 4 bits on layout level using 45nm technology," 2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE), Coimbatore, 2014, pp. 1-6.

[3] N. G. NikDaud, F. R. Hashim, M. Mustapha and M. S. Badruddin, "Hybrid modified booth encoded algorithm-carry save adder fast multiplier," The 5th International Conference on Information and Communication Technology for The Muslim World (ICT4M), Kuching, 2014, pp. 1-6.

[4] S. Nithya and M. N. V. Nithya, "An efficient fixed width multiplier for digital filter," 2014 IEEE 8th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, 2014, pp. 96-102.

[5] C. Xiaoping, H. Wei, C. Xin and W. Shumin, "A New Redundant Binary Partial Product Generator for Fast 2n-Bit Multiplier Design," 2014 IEEE 17th International Conference on Computational Science and Engineering, Chengdu, 2014, pp. 840-844.

[6] R. P. Rajput and M. N. S. Swamy, "High Speed Modified Booth Encoder Multiplier for Signed and Unsigned Numbers," 2012 UKSim 14th International Conference on Computer Modelling and Simulation, Cambridge, 2012, pp. 649-654.

[7] B. C. Paul, S. Fujita and M. Okajima, "ROM-Based Logic (RBL) Design: A Low-Power 16 Bit Multiplier," in IEEE Journal of Solid-State Circuits, vol. 44, no. 11, pp. 2935-2942, Nov. 2009.

[8] G. W. Reitwiesner, "Binary arithmetic," Advances in Computers, vol. 1, pp. 231-308, 1960.

[9] K. K. Parhi, VLSI Digital Signal Processing Systems: Design and Implementation. John Wiley & Sons, 2007.

[10] K. Yong-Eun, C. Kyung-Ju, J.-G. Chung, and X. Huang, "Csd- based programmable multiplier design for predetermined coefficient groups," IEICE Trans. Fundam. Electron. Commun. Comput. Sci., vol. 93, no. 1, pp. 324-326, 2010.