# High Speed and Area Efficient VLSI Architecture for Radix-4 Complex Booth Multiplier

ShaluTurker [1], Prof. Amrita Khera[2]

[1]*Research Scholar, Department of Electronics and Communication, Trinity Institute of Technology & Research, Bhopal*

[2]*Assistant Professor, Department of Electronics and Communication, Trinity Institute of Technology & Research, Bhopal*

***Abstract—The main objective of this research paper is to design architecture for radix-4 complex multiplier by rectifying the problems in the existing method and to improve the speed by using the common Boolean logic (CBL). The multiplier algorithm is normally used for higher bit length applications and ordinary multiplier is good for lower order bits. These two methods are combined to produce the high speed multiplier for higher bit length applications. The problem of existing architecture is reduced by removing bits from the remainders. The proposed algorithm is implementation Xilinx software with Vertex-7 device family.***

***Keywords: -Vedic Multiplier, Complex Multiplier, Common Boolean Logic Adder, Xilinx Software.***

## I. INTRODUCTION

Multipliers are widely used with constant growth of computer applications. Multiplication is an important fundamental function in arithmetic operations. The performance of computer applications are mostly depends on the performance of multiplication. The major factors in the design or implementation of multipliers are chip area and speed of multiplication [1]. There is a high demand of high speed multiplications which requires less hardware. Various algorithms and the architectures have been proposed to design high-speed and low-power multipliers. A multiplier which uses Modified Booth Algorithm are designed taking into account the less area consumption of booth algorithm because of less number of partial products and speeder accumulation of partial products and less power consumption of partial products addition using adder. The Booth's algorithm performs the encoding process serially. Hence the modified booth algorithm, which is proposed performs encoding in parallel and is implemented to design fast multiplier. Ripple carry adder and carry look ahead adder are employed for high speed accumulation [2].

The generation of each of these odd multiplies a two term addition or subtractions, yielding a total of carry propagate additions. However, the advantage of the high radix is that the partial product is further reduced. For instance, for radix-16and n-bit operands, about n/4 partial products are generated. Although less popular than radix-4, there industrial instances of radix-8 [3] and radix-16 multipliers [4] in microprocessors implementations. The choice of these radices is related to area/delay/power of pipelined multipliers (or fused multiplier address in the case of a Intel Itanium microprocessor [5]), for balancing delay between stages and/or reduce the number of pipelining flip-flops. Today highly energy-delay optimized, while partial product reductions trees suffer the increasingly serious problems related a complex wiring and glitching due to unbalanced signal. Optimal pipelining in fact, is a key in current and future multiplier (or multiplier-add) units: 1) the of the pipelined unit is very important, even for throughput oriented applications, as it impacts the energy of the whole core [5]; and 2) the placement of the pipelining flip-flops should at the same time minimize total power, due to the number of flip-flops required and the signal propagation paths. Two's complement radix-4 Booth multipliers, thus leaving open the research and extension to higher radices and unsigned multiplications unsigned integer arithmetic or mantissa times mantissa in a floating-point unit). Unsigned multiplication produces a positive carry out during recoding (this depends to one additional row, increasing the maximum height of the partial product array by one row, not just in one but in several columns. For all these reasons, we need to extend techniques in [6].

The Booth algorithm was invented by A. D. Booth, forms the base of Signed number multiplication algorithms that are simple to implement at the hardware level, and that have the potential to speed up signed multiplication Considerably. Booth's algorithm is based upon recoding the multiplier, y, to a recoded, value, z, leaving the multiplicand, x, unchanged. In Booth recoding, each digit of the multiplier can assume negative as well as positive and zero values. There is a special notation, called signed digit (SD) encoding, to express these signed digits. In SD encoding +1 and 0 are expressed as 1 and 0, but -1 is expressed as 1 [7].

Among them just two sutras are pertinent for increased activity. They are UrdhavaTriyakbhyam sutra (truly implies vertically and across) and Nikhilam Sutra (truly implies All from 9 and last from 10). Urdhava-Triyakbhyam is a non-specific technique for augmentation. The rationale behind UrdhavaTriyakbhyam sutra is especially like the conventional cluster multiplier.

Here the paired usage of this calculation is determined in light of a similar rationale utilized for decimal numbers. The double usage of Nikhilam Sutra isn't yet effective.

Multiplier furthermore, basic Boolean rationale snake can contrast and regular strategy which is processed by Vedic multiplier, XOR entryway and half viper. Proposed procedure gives less way delay and less territory. Information grouping of Conventional strategy is significantly more than to proposed technique; however proposed technique has less spread postponement. Region and engendering postponement can be decreased by the guide of basic Boolean rationale viper. This viper will be composed like as swell convey snake.

## II. COMPLEX MULTIPLIER

Suppose two numbers are complex then

$$A = A_r + jA_i \quad (1)$$

$$B = B_r + jB_i \quad (2)$$

The product of A and B then

$$P = A \times B \quad (3)$$

$$P = A_r \times B_r - A_i \times B_i + j(A_r \times B_i + A_i \times B_r) \quad (4)$$

$$P_r = A_r \times B_r - A_i \times B_i \quad (5)$$

$$P_i = A_r \times B_i + A_i \times B_r \quad (6)$$

Where $P_r$and $P_i$ is speaks to the genuine and nonexistent piece of the yield of the mind boggling multiplier.

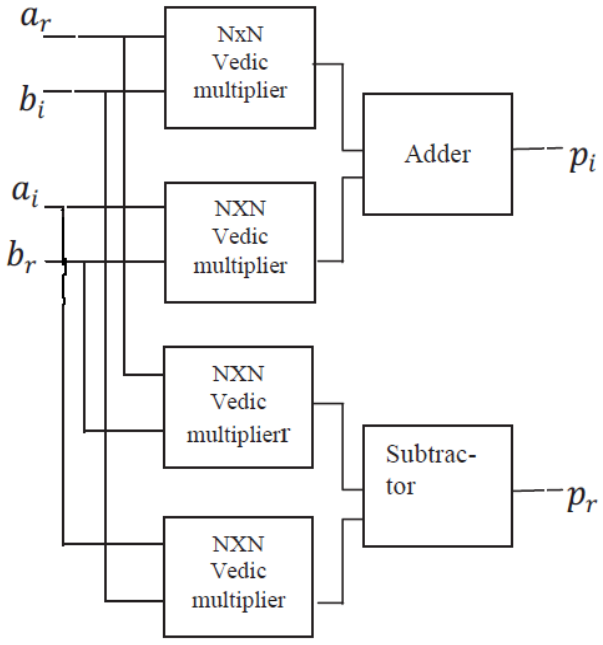


**Figure 1: Block Diagram of Complex Multiplier for four Vedic Multiplier**

Ar and Ai is speaks to the genuine and fanciful piece of the principal contribution of the unpredictable multiplier. Br and Bi is speaks to the genuine and nonexistent piece of the second contribution of the unpredictable multiplier.

Complex multiplier for four Vedic multipliers is shown in figure 1. In this block diagram reduce four Vedic multipliers to three Vedic multipliers is shown in below:

$$P_r = A_r \times B_r - A_i \times B_i = A_r(B_r + B_i) - B_i(A_r + A_i) \quad (7)$$

$$P_i = A_r \times B_i + A_i \times B_r = A_r(B_r + B_i) + B_r(A_i - A_r) \quad (8)$$

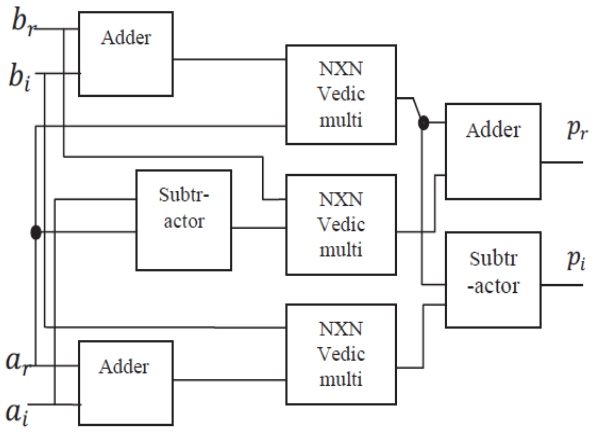


**Figure 2: Block Diagram of Complex Multiplier for three Vedic Multiplier**

## III. PARTITION MULTIPLIER USING CBL

In our proposed method the high speed Vedic multiplier method is replaced by the partition multiplier method which claims to provide a better speed and less propagation delay. Here we have used four multipliers $M_0$, $M_1$, $N_0$ and $N_1$ of 4-bit to perform 8-bit multiplication. The method used is the addition of all partial product formed by the cross multiplication of one bit with another. The LSB bits of first multiplier $M_0$ (3-0) add with LSB bits $N_0$ (3-0) of the final output $t_1$. Padding n/2-bit zero add with final output t1. Another bits of first multiplier $M_0$ (7-4) are added in series with LSB 4-bits of second multiplier $N_0$ (3-0) to form the 8-bits, which in turn get added padding (n/4) zero with $t_2$ and Padding (n/4) zero of the final output (15-0).

### Common Boolean logic Adder

In a zone effective and low power half snake based Carry select viper (CSLA) utilizing normal Boolean rationale is outlined so as to upgrade the general framework execution as far as territory and power as contrast with other existing designs. Half viper is utilized to produce the incomplete entirety for cin=0 and basic Boolean rationale (CBL) is utilized for figuring halfway total for cin=1.

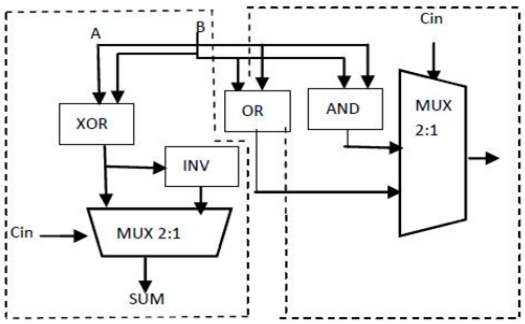


**Figure 3: Block Diagram of CBL Adder**

This engineering is utilized to expel the recreated viper cells in the traditional CSLA, spare number of entryway tallies and accomplish a low power. Through investigating reality table of a solitary piece full snake we recommend that for producing yield summation and convey motion for cin=0, require just a single XOR door and one AND entryway individually, the yield summation motion for cin=1 is simply the opposite as cin=0. Summed up figure of regular Boolean rationale Adder is appeared in figure 3.

## IV. RADIX-4 ALGORITHM

To further decrease the number of partial products, algorithms with higher radix value are used. In radix-4 algorithm grouping of multiplier bits is done in such a way that each group consists of 3 bits as mentioned in table 1. Similarly the next pair is the overlapping of the first pair in which MSB of the first pair will be the LSB of the second pair and other two bits. Number of groups formed is dependent on number of multiplier bits. By applying this algorithm, the number of partial product rows to be accumulated is reduced from n in radix-2 algorithm to n/2 in radix-4 algorithm. The grouping of multiplier bits for 8-bit of multiplication is shown in figure 4.

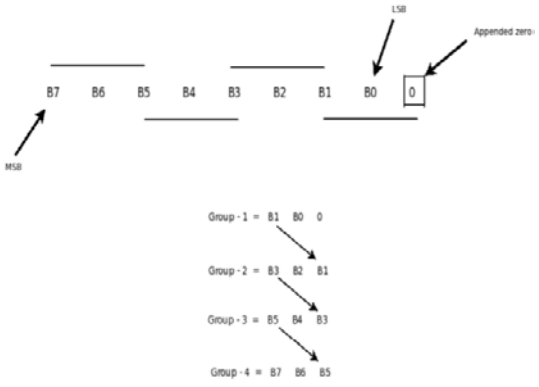


**Figure 4: Grouping of multiplier bits in Radix-4 Booth algorithm**

For 8-bit multiplier the number groups formed is four using radix-4 booth algorithm. Compared to radix-2 booth algorithm the number of partial products obtained in radix-4 booth algorithm is half because for 8-bit multiplier radix-2 algorithm produces eight partial products. The truth table and the respective operation are depicted in table 1. Similarly when radix-8 booth algorithm is applied to multiplier of 8-bits each group will consists of four bits and the number of groups formed is 3. For 8x8 multiplications, radix-4 uses four stages to compute the final product and radix-8 booth algorithm uses three stages to compute the product. In this thesis, radix-4 booth algorithm is used for 8x8 multiplications because number components used in radix-4 encoding style.

**Table 1: Truth Table for Radix-4 Booth algorithm**

| $B_{i+1}$ | $B_i$ | $B_{i-1}$ | Operation | $Y_{i+1}$ | $Y_i$ | $Y_{i-1}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | +0 | 0 | 0 | 0 |
| 0 | 0 | 1 | +A | 0 | 1 | 0 |
| 0 | 1 | 0 | +A | 0 | 1 | 0 |
| 0 | 1 | 1 | +2A | 0 | 0 | 1 |
| 1 | 0 | 0 | -2A | 1 | 0 | 1 |
| 1 | 0 | 1 | -A | 1 | 1 | 0 |
| 1 | 1 | 0 | -A | 1 | 1 | 0 |
| 1 | 1 | 1 | -0 | 1 | 0 | 0 |

## V. SIMULATION ANALYSIS

Simulation of these tests should be possible by utilizing Xilinx 14.1i VHDL instrument. In this paper we are concentrating on engendering delay. Spread postpone must be less for better execution of advanced circuit.

As appeared in table II the quantity of cut, number of LUTs, delay is acquired for the complex Vedic multiplier utilizing basic Boolean rationale viper and past calculation. From the investigation of the outcomes, it is discovered that the complex Vedic multiplier utilizing basic Boolean rationale snake gives a predominant execution as contrasted and past calculation for Xilinx programming.

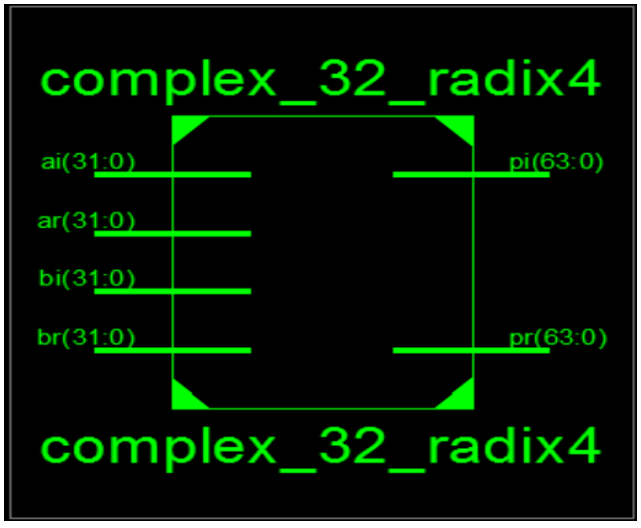


**Figure 5: View Technology Schematic of Complex Multiplier using Radix-4 Booth Multiplier**

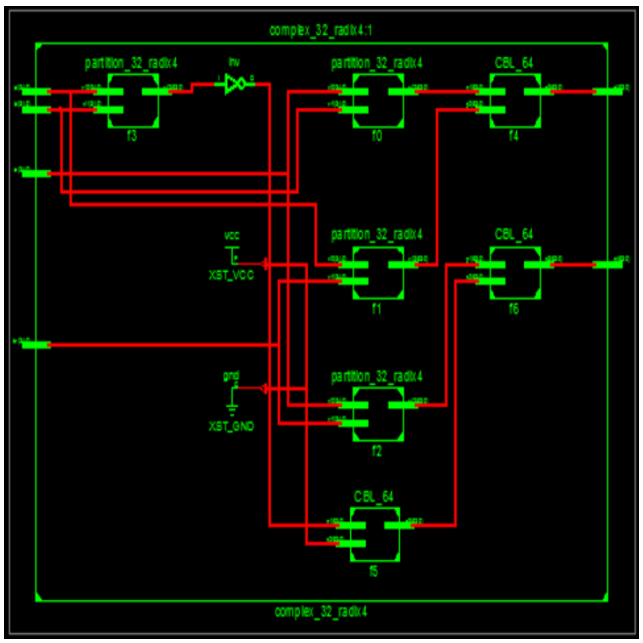


**Figure 6: RTL View of Complex Multiplier using Radix-4 Booth Multiplier**

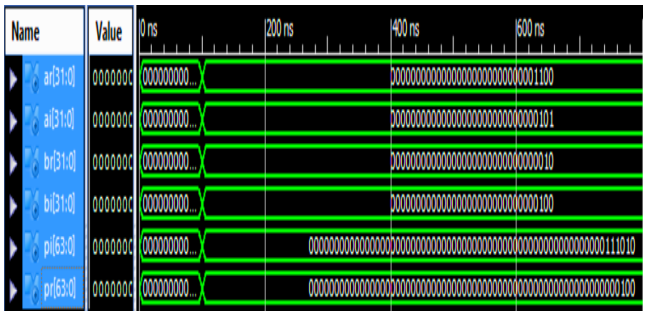


**Figure 7: Output Binary Waveform of Radix-4 Complex Multiplier using CBL Adder**

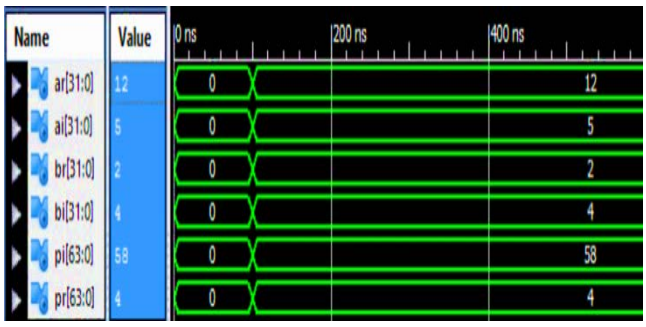


**Figure 8: Output Decimal Waveform of Radix-4 Complex Multiplier using CBL Adder**

From the analysis of the results, it is found that the complex multiplier using CBL adder gives a superior performance as compared with previous algorithm for Vertex-7 device family. The output waveform of the complex multiplier using CBL adder is shown in figure 7 and figure 8 respectively.

**Table II: Comparison Result for 32-bit Radix-4 Complex Multiplier for CBL Adder**

| Design | Number | Number | Delay |
|---|---|---|---|
| | of LUTs | of IOBs | |
| Previous Complex Multiplier [2] | 10416 | 256 | 25.979 ns |
| Complex Radix-4 Multiplier using Ripple Carry Adder | 10642 | 256 | 26.927 ns |
| Complex Radix-4 Multiplier using CBL Adder | 9892 | 256 | 24.006 ns |

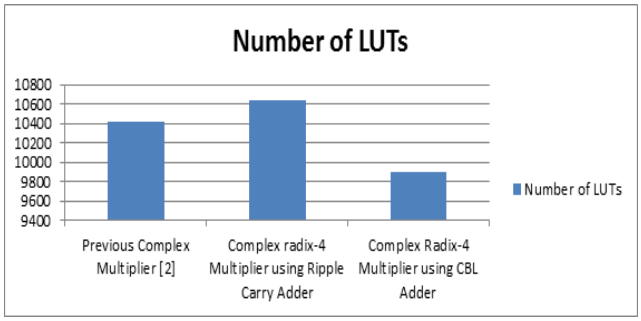


**Figure 9: Bar graph of the 32-bit Radix-4 Complex Vedic multiplier for LUTs**

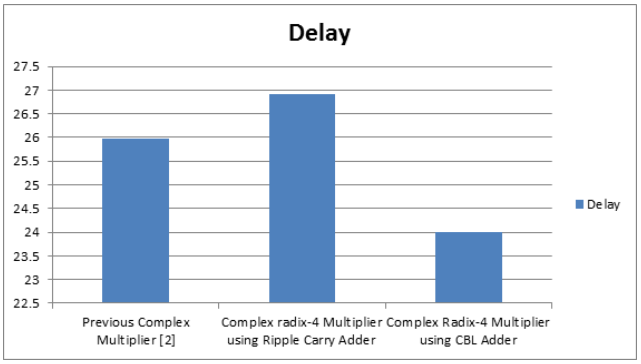


**Figure 10: Bar graph of the 32-bit Radix-4 Complex Vedic multiplier for Delay**

Figure 9 and figure 10 shows the graphical illustration of the performance of CM using CBL adder discussed in this research work in term of number of LUTs and delay. From the above graphical representation it can be inferred that the CM using CBL adder gives the best performance as compared with previous algorithm.

## VI. CONCLUSION

In this paper design of CBL adder, partition multiplier, radix-4 multiplierand complex multiplier is presented. From implementation results it is observed that the complex multiplier consumes less delay compare to previous design. The architecture designs of 32 x32-bit; Modified Radix-4 Booth Encoder Multiplier is done.

## REFERENCES

[1] D. Kalaiyarasi and M. Saraswathi, “Design of an Efficient High Speed Radix-4 Booth Multiplier for both Signed and Unsigned Numbers”, 4th International Conference on Advances in Electrical, Electronics, Information,

Communication and Bio-Informatics (AEEICB), IEEE 2018.

[2] Prof. S. B. Patil, Miss. Pritam H. Langade, “Design of Improved Systolic Array Multiplier and Its Implementation on FPGA”, International Journal of Engineering Research and General Science Volume 3, Issue 6, November-December, 2015

[3] ElisardoAntelo, Paolo Montuschi and Alberto Nannarelli, “Improved 64-bit Radix-16 Booth Multiplier Based on Partial Product Array Height Reduction”, IEEE Transactions On Circuits And Systems—I: Regular Papers, Vol. 64, No. 2, February 2017.

[4] Kavita and Jasbir Kaur, “Design and Implementation of an Efficient Modified Booth Multiplier using VHDL”, Special Issue: Proceedings of 2nd International Conference on Emerging Trends in Engineering and Management, ICETEM 2013.

[5] Shiann-RongKuang, Jiun-Ping Wang and Cang-Yuan Guo, “Modified Booth Multipliers With a Regular Partial Product Array”, IEEE Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 56, No. 5, May 2009.

[6] S. Vassiliadis, E. Schwarz, and D. Hanrahan, “A general proof for overlapped multiple-bit scanning multiplications,” IEEE Trans. Comput., vol. 38, no. 2, pp. 172–183, Feb. 1989.

[7] D. Dobberpuhl et al., “A 200-MHz 64-b dual-issue CMOS microprocessor,” IEEE J. Solid-State Circuits, vol. 27, no. 11, pp. 1555–1567, Nov. 1992.

[8] E. M. Schwarz, R. M. A. III, and L. J. Sigal, “A radix-8 CMOS S/390 multiplier,” in Proc. 13th IEEE Symp. Comput. Arithmetic (ARITH), Jul. 1997, pp. 2–9.

[9] J.Clouser etal.,“A600-MHz superscalar floating-point processor,” IEEE J. Solid-State Circuits, vol. 34, no. 7, pp. 1026–1029, Jul. 1999.

[10] S. Oberman, “Floating point division and square root algorithms and implementation in the AMD-K7 microprocessor,” in Proc. 14th IEEE Symp. Comput. Arithmetic (ARITH), Apr. 1999, pp. 106–115.

[11] R. Senthinathan et al., “A 650-MHz, IA-32 microprocessor with enhanced data streaming for graphics and video,” IEEE J. Solid-State Circuits, vol. 34, no. 11, pp. 1454–1465, Nov. 1999.