Higher Order Radix Algorithm Based VLSI Architecture for Parallel MAC

[a] K.Sandeep Kumar, [a] R.Sravanthi, [a]Abdul Raheem
[a]Associate Professor, Electronics and Communication Engineering, Aurora's Scientific, Technological and Research Academy, Hyderabad, India-500005.

Abstract

In this paper a novel scheme of multiplication with accumulation for advanced arithmetic applications where higher speeds are demanded is presented.In this paper,when compared to conventional approach we merge accumulation with multiplication to derive a novel type hybrid carry save adder, hence the overall operational performance will be improved as the accumulator is being merged in carry save adders. The scheme presented in this paper uses a one's complement based radix-2 modified booth's algorithm , radix-4 algorithm ,radix-8 algorithm,radix-16 and radix-32 algorithms which has the altered array for of sign extension to adjust the bit density. The specialty of carry save adder presented in this paper is,carry propagation to LSB of the partial products and generation of LSB's well in advance to decrease the number of inputs to the final adder array module. And the presented model of MAC accumulates the intermediate data in the form of sum and carry bits instead of the output of the final address.Thus pipelining is done and the overall operational speed is improved in multi fold. Instead of lower radix schemes radix 8,radix 16,radix 32 are used to reduce partial products for enhancing the performance.The proposed scheme is useful for the digital data processing applications like DSPs,digital filters ,multimedia .

KEYWORDS- Booth's multiplier, Carry save adders, Mutiply and Accumulate block, Radix-2, 4,8,16 and 32 schemes.

I. Introduction

With the recent advancements in the area of multimedia, communication systems, advanced signal processing, video and image processing etc. there is a requirement to handle more number of input bits at higher speeds. The proposed multiplier with accumulator (Multiplier and Accumulator (MAC)) module [1] is essential in the field of digital signal processing operations like digital filters, convolution, and intermediate products. Many DSP's uses non linear methods such as discrete cosine and wavelet transforms which are observed and developed by repeated multiplication and additions. The speed of DSP's depends majorly on addition and multiplication operation determines execution speed.In this Radix-2[13], Radix-4, Radix-8 and Radix-16 schemes are used to enhance the performance of multiplication by reducing the critical path thus increasing the speed[2][3][4][13]. In the conventional multiplication we use shifters, full adders, half adders to compute final results. This paper is implemented with higher order booth's algorithms [5] with novel compressors instead of conventional scheme which was developed based on Wallace tree adders [6].

The proposed multiplier consists of following modules.

I. Booth encoders.
II. To reduce the partial product depth compressors are used.
III. Final addition through a chain of full adders.

Wallace tree is used to add the partial products from encoder as speed as possible and its operation time is proportional to $O(\log_2 N)$, where 'N' is the number of inputs and the logic behind compression is to count number of 1's among the 'N' inputs producing $(\log_2 N)$ number of outputs, in real time the most effective way to enhance performance is to diminish the number of partial products. One of efficient mechanism for designing MAC for general purpose DSP is proposed by elguibaly [8]. It is the schema where the accumulation has been combined with carry save adders so that the partial products will be compressed. Hence the performance will be enhanced as the number of inputs to the final adder is reduced and critical path also adjusted accordingly. In this paper, a novel schema for high end MAC is proposed in this MAC, the operations such as multiplication and addition are merged to form a new hybrid type carry save adder for better critical path analysis.In the proposed carry save adder tree [7][13], a carry look ahead adder also presented to reduce the number of inputs to the final adder stage. And pipelining [13] approach chosen to enhance throughput.

I.All about MAC

In this section the basic multiply and accumulate (MAC) operation is analyzed [8, 9, 13]. A basic multiplier scheme comprises of three operational steps

I. Radix encoding: where partial products are generated from the multiplier (Y) and multiplicand (X).
II. Array of adders for partial product compression which will be helpful for the suppression of partial products and convert them into the sum and carry form.
III. Final Addition stage where results are produced by adding sum and carry bits which are available from the above stage.

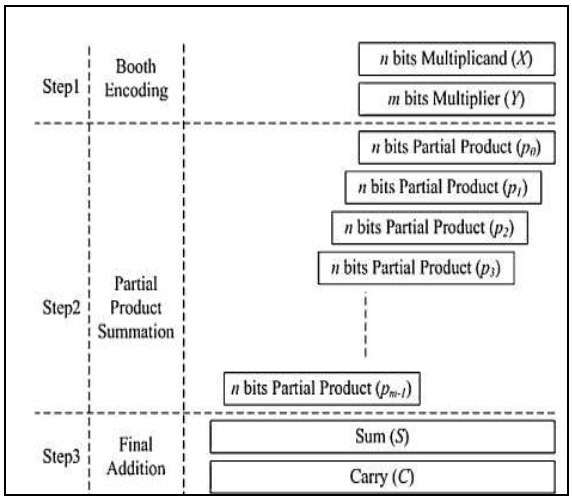


Figure.1. Basic arithmetic steps of multiplication without accumulation.

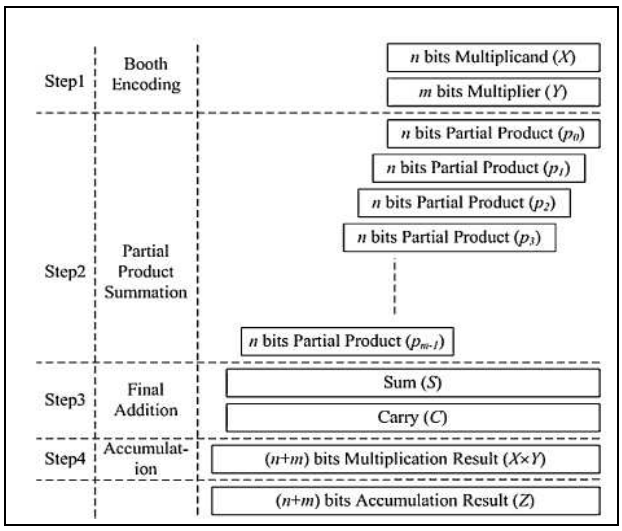


Figure.1a. Basic arithmetic steps of multiplication and accumulation.

If the accumulation module is to be inserted then the MAC has four stages [13] as shown in figure1a.

A generalized hardware schema is presented in the figure 2. performs multiplication operation by multiplying the input multiplexer 'X' and multiplicand 'Y' and the new result is added with old multiplication result 'Z' as the means of accumulation step.

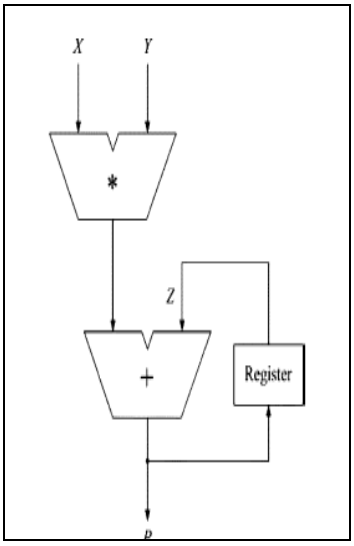


Figure2. Hardware architecture of general MAC

Let the size of input be 'N' bits wide then the 2's complement of binary number 'X' can be expressed as

$$X = 2^{N-1}x_{N-1} + \sum_{i=0}^{N-2} x_i . 2^i , x_i \in 0,1 \qquad (1)$$

If equitation (1) is expressed in base-4 type redundant sign digit format then in order to apply the radix-2 booth's algorithm, it will be [7] expressed as

$$X = \sum_{i=0}^{\frac{N}{2}-1} di . 4i \qquad (2)$$

$$di = -2.x_{2i+1} + x_{2i} + x_{2i-1} \qquad (3)$$

If equation (2) is used, the multiplication can be expressed as

$$X * Y = \sum_{i=0}^{N-1} di.2^{2i}.Y \quad (4)$$

If these equations are used then the MAC output 'P' expressed as

$$P = (X * Y) + Z \quad (4a)$$

Where X*Y is multiplication, Z is accumulation

$$P = \sum_{i=0}^{N-1} di.2^{i}.Y + \sum_{i=0}^{2N-1} Zi.2^{i} \quad (5)$$

Right side of equation (5) is calculated independently and final sum is observed by adding the two results. The MAC hardware designed by equation (5) is called the standard design [4].

If N-bit data are multiplied, then the numbers of partial products generated are proportional to 'N' and in order to add them serially the execution time is also proportional to 'N'. The fastest multiplier uses radix-2 booth encoding that generates the partial products and are compressed by a Wallace tree by using CSA's.[13]

If radix-2[5] booth encoding is used the partial products are reduced and the number of inputs to Wallace adder tree is reduced to half. If radix-4, radix-8, radix-16 are used the partial products are reduced proportionately.

Novel MAC:

In this section we describe the new method of MAC based on standard design [4]. In addition to this new ways of partial product generations and a hybrid CSA are presented [13].

A. Basic MAC architecture hardware

If there is a need to multiply two N-bit numbers and accumulate into a 2N-bit number is considered, the critical path is determined by the 2N-bit accumulation operation. If a pipeline scheme is introduced for each step in standard design of fig.1 the delay of the last stage i.e., accumulation must be reduced in order to enhance the performance. The overall performance can be enhanced as the accumulation is a inbuilt operation in the CSA tree and critical path is determined by the final adder tree in the multiplier module.To enhance the speed of operation the partial products and the depth of the partial product should be decreased. A 2-bit CLA is used to add the final bits in the CSA and in addition to increase the throughput pipelining is used, the sums and carries from the CSA are accumulated instead of outputs from the final adder as sane as the sum and carry from the CSA in the previous cycle are feeded back to CSA. Due to this feedback style of feeding sum and carry the overall inputs to CSA are increased when compared with the standard design [8][13].

Expression for novel MAC:

First the multiplication equation (4) is expressed by elaborating as

$$X * Y = d_0.2Y + d_1.2^2.Y + d_2.2^4Y + \cdots + d_{(N/2)-1}.2^{N-2}.Y \quad (6)$$

If equation (6) is divided into the first partial product, sum of the intermediate partial products, and final partial products, it can be expressed as

$$X * Y = d_0.2Y + \sum_{i=1}^{N-2} di.2^{2i}.Y + d_{(N/2)-1}.2^{N-2}.Y \quad (7)$$

The equation (7) contains three types of data which are feedback for accumulation, which are sum, carry and pre accumulated results of the sum and carry from lower order bits. Now the proposed scheme is applied to Z in (5). If 'Z' is first divided into upper and lower bits and rearranged (8) will be observed.

$$Z = \sum_{i=0}^{N-1} Zi.2^i + \sum_{i=N}^{2N-1} Zi.2^i \quad (8)$$

In expression (8), the right hand side first term signifies upper bits and is the value that was fed back from previous stage as sum and carry.

$$Z = \sum_{i=0}^{N-1} Zi.2^i + \sum_{i=N}^{2N-1} Zi.2^i \quad (9)$$

The second term signifies to the lower bits and is the value that was the result of addition for the sum and carry. The second term can be expressed as

$$\sum_{i=N}^{2N-1} Zi.2^i = \sum_{i=0}^{N-1} Z_{N+i}.2^i.2^N = \sum_{i=0}^{N-2} (c_i + s_i).2^i.2^N \quad (10)$$

$$P = (d_0.2.Y + \sum_{i=0}^{N-1} Zi.2^i) + (\sum_{i=1}^{\frac{N}{2}-1} d_{2i}.2^{2i}.Y + \sum_{i=0}^{N-2} c_i.2^i.2^N) + (d_{(\frac{N}{2})-1}.2^{N-2}.Y + \sum_{i=0}^{N-2} s_i.2^i.2^N) \quad (11)$$

In equation (11) right side first term is the accumulation operation to accumulate the first partial product with the added result of the sum bit and the carry bit.The second term is to accumulate the intermediate partial products with sum of the CSA that uses feedback. And the third term represents the operation to accumulate the last partial product with carry bit of the CSA.

B. Proposed novel MAC architecture

If the MAC architecture discussed in the previous section is rearranged it would be as figure 3 in which the MAC has divided into three steps.

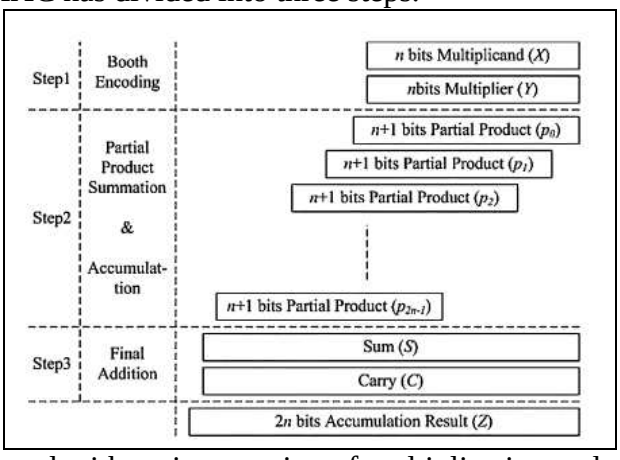


Figure3.Proposed arithmetic operation of multiplication and accumulation.

In comparison with figure.1, we observe that the accumulation has been merged into the process of adding the partial products and accumulation carried on the result step 2 not on step 3 as stated in the previous architecture and hence step 3 will not run until final sum is computed. The hardware equivalent of figure 3 is presented in figure 4[13].

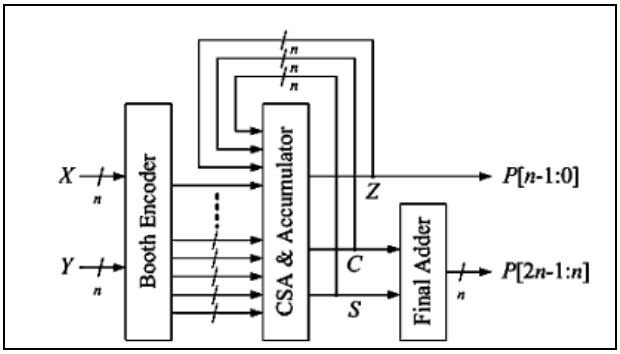


Figure4. Hardware equivalent of the proposed MAC.

The N-bit MAC inputs, X and Y are covered into an (N+1) bit partial product by passing through the booth encoder. In the CSA and accumulator is carried out along with addition of partial products. As a result a N-bit sum 'S', carry 'C' and 'Z' previous stage data are generated. These three bits are fed as input to for the next stage accumulation. If the final sum of MAC is needed then P[2N-1:N] is generated by adding the 'S' and 'C' in the final adder and should be combined with P[N-1:0] that were already generated[13].

C. Novel CSA scheme

CSA is developed as per the proposed MAC and is shown in figure 5.

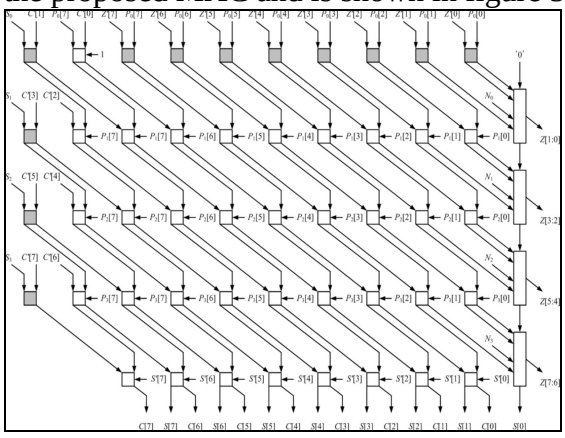


Fig. 5. Architecture of the proposed CSA tree.

It is based on equation (11). In the figure5, 'Si' is to minimize the sign expansion and Ni is to convert 1's complement number into 2's complement number. S[i] and C[i] correspond to the i^{th} bit of the feedback sum and carry. Z[i] is the i^{th} bit of the sum of the lower bits for each partial product that were added in advance and Z[i] is the previous result.Figure 5 shows example of a 8 bit multiplier, totally four partial products P0[7:0] to P3[7:0] are generated from booth's encoder module. In equation (11) $d_0.Y$ and $d_{(N/2)-1}.2^{N-2}.Y$ correspond to these terms respectively. This CSA require at least four rows of FF's for the four partial products. Thus totally '5' FA rows are needed since one more level of rows are needed for accumulation bits.

For an n×n bit MAC architecture, the level of CSA needed is (n/2+1). In the above figure where square represents a FA, gray represents a HA and rectangular module with 6 inputs represents a CLA of 2-bit with a carry input. The critical path of the MAC is determined by 2-bit CLA we can use FA but the overall performance will be degraded. Table1 shows characteristics of proposed CSA followed by the hardware resource calculation in table2[13].

	[4]	[8]	Proposed
Number system	2's Complement	1's Complement	1's Complement
Sign Extension	Used	Used	Not used
Accumu-lation	Result of final addition	Result of final addition	CSA sum and carry
CSA Tree	FA,HA	FA,CLA 2 bits	FA,HA, CLA 2 bits
Final Adder	2n bit wide	(n+2) bit	N bit

Table1.Charecteristics of CSA

II.A.Radix-2 Encoding.

In order to reduce the number of partial products radix multiplication schemes are proposed. One among the radix scheme is radix-2 encoding. In this algorithm only two bits of multiplier are considered at a time. Table 3 shows the recording by radix-2 method. In radix-2 only 1, 0, -1 is to be considered.

Table2.Hardware Resources

Block		Partial Product
0	0	0 *multiplicand
0	1	1*multiplicand
1	0	-1*multiplicand
1	1	0*multiplicand

Table3.Radix-2 Encoding

Component	**[4]**		**[8]**		**Proposed**	
	General	**16 bit**	**General**	**16 bit**	**General**	**16 bit**
FA	$(n^2/2+n)$	964.8	$(n^2/2+2n+3)$	1092.1	$(n^2/2+n/2)$	911.2
HA	0	0	0	0	3n/2	76.8
CLA 2-bit	0	0	(n/2-1)	49	n/2	56
Accumu-lator	(2n+1) bits CLA	214	-	-	-	-
Final Adder	2n bits	197	(n+2) bits	109.5	n bits	97
Total	-	1375.8		1250.6		1141

Example: Radix-2 can be applied by altering multiplicand 'Y' by appending '0' at LSB position. Now start pairing two bits at a time as shown below.
Example : Y= 01110 and Altered Y= **0**11100

Algorithm: 'B' converted as 011100
Modified : **0** 1 1 1 0 0
Pairing : (01) (11) (11) (10) (00)
1 0 0 -1 0
While pairing '2' bits are to be selected at

a time ,to form next pair one right bit from the previous pair is grouped to next bit as per data and the process should be continued to the last bit of multiplicand.

Multiplication of 'A' with coded data.

A: 0 0 1 0 0

Coded data: 1 0 0 -1 0

Bold terms are padded terms inorder to match partial product dimensions and discard carry data .The left most bits which are bolded are padded bits and right most bolded bits are for sign extension representation, the number of bits are based on the length of operands.

It is possible to reduce the number of partial products by half, by using the technique of radix-4 booth recoding. The basic idea is that, instead of shifting and adding for ever column of the multiplicated term and multiplying by 1 or 0 , we only take every second column and multiply by ±1,±2,0, to obtain the same results. So, to multiply by 7, we can multiply the partial product aligned against the least significant bit by -1 , and multiply the partial product aligned with the third column by 2.

Partial product 0 = Multiplicand * -1 , shifted left 0 bits (X -1)

Partial product 1 = Multiplicand * 2, shifted left 2 bits (X 8)

These results are same as the conventional shifts and add method results.

PP0 = Multiplicand *1, Shifted left '0' bits (X1)

PP1 = Multiplicand *1, Shifted left '1' bits (X2)

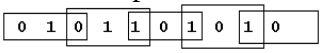
PP2 = Multiplicand *1, Shifted left '2' bits (X4)

PP3 = Multiplicand *1, Shifted left '3' bits (X0)

The advantage of this method is having the number of p.p. Thus the hardware complexity is reduced proportionally and leads to low power consumption. But a complexity penality will be observed for solving the results in 2's or 1's complement format.

B.Radix-4 Scheme

To radix-4 [16] booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouply starts from the LSB and the first block only uses two bits of the multipler,since threr is no previous block to overlap.



In the grouping of bits from the multiplier term, for use in booth recoding. The least significant block uses only two bits of the multiplier, and assumes a zero for the third bit. The overlap is necessary so that we know what happened in the last block, as the MSB of the block acts like a sign bit.Table shows the booth recoding strategy for each of the possible blocks values.

Block	Partial Products
000	0
001	1 * Multiplicand
010	1 * Multiplicand
011	2 * Multiplicand

100	-2 * Multiplicand
101	-1 * Multiplicand
110	-1 * Multiplicand
111	0

Table 4:Radix-4 Encoding

With the help of the above table 4 we will encode the data based on the pairing blocks.Since we use the LSB of each blocks tom know what the sign bit was on the previous block, and there are never any negative products before the LSB , the LSB of first block is always assumed to be '0'.

C.Radix-8 Booth Encoder

Radix-8 [15] booth recoding applies the same algorithm as that of radix-4, instead of grouping two (as per radix-2), three (as per radix-4) we group four bits at a time , hence the partial products reduced proportionally.Encoding scheme is presented in Table5.

Block	Partial Products
0000	0*Multiplicand
0001	+1*Multiplicand
0010	+2*Multiplicand
0011	+3*Multiplicand
0100	+4*Multiplicand
0101	+5*Multiplicand
0110	+6*Multiplicand
0100	+7*Multiplicand
1000	-7*Multiplicand
1001	-6*Multiplicand
1010	-5*Multiplicand
1011	-4*Multiplicand
1100	-3*Multiplicand
1101	-2*Multiplicand
1110	-1*Multiplicand
1111	0*Multiplicand

Table5.Radix-8 Encoding

D.Radix-16 and Radix-32 Booth Encoders

As per the advanced system design there is a huge requirement to model high speed designs, especially for signal processing applications higher order multipliers are needed.so radix-16 and radix-32 encoding schemes are proposed[14] .Following steps are involved in radix-16.

1) Sign extension by '1' position to ensure that the number of bits are even.
2) Append a'0' at the right side of the LSB of the multiplier

3) Group '5' bits at a time as mentioned in the previous schemes.

And inorder to design a 'n'bit parallel multiplier only'n/4' partial products are required in radix-16.If 'n' bit multipliers and multiplicand are chosen then radix-32 reduces the number of partial products to'1/5'th of the actual number.Grouping in radix-32 is shown below

0 111 0 ***1****110 1****0***10110.

Bold digit represents overlapped bit in grouping.

III. Conclusion

The archictecture presented in this paper was based on xilinx syntthesis tools In this paper a novel architecture for parallel MAC to model multiply and accumulate block which is based on different radix schemes such as radix-2,radix-4,radix-8,radix-16 and radix-32 for the reduction of partial products is presented.This is the key operation involved in todays digital signal preocessors and multimedia applications.In this paper the overall MAC performance is doubled when compared to its previous works as Pipelining mechanism is introduced to enhance the throughput of the operations. Many researchers have designed MAC based on backend tools for better area and power estimations.

IV. Acknowledgement

We would like to thank the Ms.C.Srilatha, Director, Aurora's Scientific, Technological and Research Acdemy, Hyderabad and Mr.Mohd Imaduddin, Head of the Department, Electronics and Communication Engineering, Aurora's Scientific, Technological and Research Acdemy, Hyderabad for providing enough resources, constant encouragement and good support.

References

[1] J. J. F. Cavanagh, *Digital Computer Arithmetic*. New York: McGraw-Hill, 1984.
[2] O. L. MacSorley, "High speed arithmetic in binary computers," *Proc.IRE*, vol. 49, pp. 67–91, Jan. 1961.
[3] S. Waser and M. J. Flynn, *Introduction to Arithmetic for Digital Systems Designers*. New York: Holt, Rinehart and Winston, 1982.
[4] A. R. Omondi, *Computer Arithmetic Systems*. Englewood Cliffs, NJ:Prentice-Hall, 1994.
[5] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Math.*, vol. IV, pp. 236–240, 1952.
[6] C. S. Wallace, "A suggestion for a fast multiplier," *IEEE Trans. Electron Comput.*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964.
[7] A. R. Cooper, "Parallel architecture modified Booth multiplier," *Proc. Inst. Electr. Eng. G*, vol. 135, pp. 125–128, 1988.
[8] F. Elguibaly, "A fast parallel multiplier–accumulator using the modified Booth algorithm," *IEEE Trans. Circuits Syst.*, vol. 27, no. 9, pp.902–908, Sep. 2000.
[9] G. Goto, T. Sato, M. Nakajima, and T. Sukemura, "A 54X54 regular structured tree multiplier," *IEEE J. Solid-State Circuits*, vol. 27, no. 9, pp. 1229– 1236, Sep. 1992.
[10] J. Fadavi-Ardekani, "MXN Booth encoded multiplier generator using optimizedWallace trees," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 2, pp. 120–125, Jun. 1993.

[11] N. R. Shanbag and P. Juneja, "Parallel implementation of a 4X4-bit multiplier using modified Booth's algorithm," *IEEE . Solid-State Circuits*, vol. 23, no. 4, pp. 1010–1013, Aug. 1988.

[12] Manas M. Ramteke,Prof. P. R. Indurkar1, Prof. Mrs. D. M. Khatri, High Speed Modified Booth's Signed 64x64 Bit Multiplier Using Wallace Structure by Radix-32 *International Journal on Recent and Innovation Trends in Computing and Communications*,ISSN: 2321-8169,Volume: 3 Issue: 7 ,4954 – 4957.

[13] Young-Ho Seo, Member, IEEE, and Dong- Wook Kim, Member, IEEE, *IEEE transactions on very large scale integration (VLSI) systems*, vol. 18, no. 2, february 2010 .

[14] JasbirKaur, Sumit Kumar, Performance Comparison of higher radix booth multiplier using 45nm technology.*International Journal of Innovative Research in Science, Engineering and Technology (An ISO 3297: 2007 Certified Organization) Vol. 5, Issue 1, Januray*

[15] Paladugu Srinivas Teja, Design of radix-8 booth multiplier using koggestone adder for high speed

Arithmetic applications,*Emerging Trends in Electrical, Electronics & Instrumentation Engineering: An international Journal (EEIEJ)*, Vol. 1, No. 1, February 2014.

[16] Sneha Manohar Ramteke, Alok Dubey, Yogeshwar Khandagare, VLSI Designing of Low Power Radix4 Booths Multiplier *International Journal of Electrical, Electronics and Computer Systems (IJEECS)* ISSN (Online): 2347-2820, Volume -2, Issue-2, 2014