International Journal of
Scientific Engineering and Technology Research

ISSN 2319-8885
Vol.06,Issue.15
April-2017,
Pages:2801-2806

www.ijsetr.com

A Novel Modified Radix-4 Booth Encoding Technique using Redundant Binary Multipliers

Y. KRISHNA[1], N. BENARJI[2], P. VALI BASHA[3]

[1]B.Tech Scholar, Dept of ECE, N.B.K.R Institute of Science &Technology, Vidyanagar, AP, India,
E-mail: yadati.krishna@gmail.com.

[2]B.Tech Scholar, Dept of ECE, N.B.K.R Institute of Science &Technology, Vidyanagar, AP, India,
E-mail: bennynbenarji@gmail.com.

[3]Assistant Professor, Dept of ECE, N.B.K.R Institute of Science &Technology, Vidyanagar, AP, India,
E-mail: Vali.gps@gmail.com.

Abstract: The requirement of the modern computer system is a dedicated and very high speed unique multiplier unit for signed and unsigned numbers. The work mainly deals with in improving multiplication process by using Redundant Binary Technique.The redundant binary in design of high speed digital multiplier is beneficial due to high modularity and carry free addition. Generally,in high radix modified booth encoding algorithm the partial products are reduced in multiplication process. But it yields complexity in producing in generation of hard multiples.Therefore booth encoding scheme along with redundant binary scheme solves this problem by using booth encoding, RB partial product generator,RB partial product accumulator,RB to NB converter stage.In this paper implemented in Verilog HDL.

Keywords: Redundant Binary, Modified Booth Encoding, RB Partial Product Generator, RB Multiplier.

I. INTRODUCTION

The nature of the work is introduced to improve the speed of the multiplier and to remove the hard multiples and to reduce the partial products than previous work. By improving the multipliers in the ALU processors from past to present they are giving better results comparing to previous improvements. So, now introducing the Redundant Binary Representation technique in digital multipliers to overcome drawbacks in previous techniques. Redundant Binary (RB) representations first introduced by Avizienis in 1961 for fast parallel arithmetic. This new Arithmetic was applied for fast Multiplication by takagi and implemented by Edamatsu.Multiplication is a most commonly used operation in many computing systems. Infact multiplication is nothing but addition since, multiplicand adds to itself multiplier number of times gives the multiplication value between multiplier and multiplicand. But considering the fact that this kind of implementation really takes huge hardware resources and the circuit operates at utterly low speed. In order to address this so many ideas have been presented so far for the last three decades. Each one is aimed at particular improvement according to the requirement.

One may be aimed at high clock speeds and another maybe aimed for low power or less area occupation. Either way ultimate job is to come up with an efficient architecture which can address three constraints of VLSI speed, area, and power. Among these three speeds is the one which requires special attention. If we observe closely multiplication operation involves two steps one is producing partial products and adding these partial products. Thus, the speed of a multiplier hardly depends on how fast generate the partial products and how fast we can add them together. If the numbers of partial products to be generated are of less than it is indirectly means that we have achieved the speed in generating partial products. Booth's algorithms are meant for this only. To speed up the addition among the partial products we need fast adder architectures. Since the multipliers have a significant impact on the performance of the entire system, many high performance algorithms and architectures have been proposed.

II. NUMBER SYSTEM REPRESENTATIONS

In computing signed digit number representation is required to encode negative numbers in binary number systems. In mathematics, negative numbers in any base are represented by prefixing them with a – sign. However, in computer hardware, numbers are represented in binary only without extra symbols, requiring a method of encoding the minus sign. The signed digit number representation makes it possible to perform addition without carry propagation chains that are used to speed up arithmetic operations

A.Redundant Signed Digit-Carry-Free Addition Algorithm

A symmetrical signed digit (SD) number can assume the following values i.e. [$-\alpha$,..-1,0,1….α]. Where themaximum value of α must be within the following range: $[(r-1)/2] \leq \alpha \leq r-1$. In order to yield minimumredundancy, one can

Copyright @ 2017 IJSETR. All rights reserved.

choose the maximum magnitude α = [r/2]. If r = 2, then number representation is known asRedundant Binary Signed Digit (RBSD) number system. A digit set need not be the standard. Radix-r, a digit set [–k, μ], having the condition k + μ + 1≥r may be Converted to a radix-r digit set[α, β], for α + β + 1 ≥γ. The redundancy δof the number system is defined asδ = α + β + 1 −γ. In the binary signed digit number system, each digit can assume any one of three values {-1, 0, 1}. As a result redundancy is introduced in a system i.e. a number can be represented in more than one way. Due to the presence of redundancy one can perform carry-propagation free addition and hence parallel addition of two redundant numbers can be performed in a constant time independent of the word length of operands.

Example:

1 0 1 - (5)
+1 1 0 - (6)

1 0 1 1 - (11)

(Carry generated)

For 4-bit redundant representation of above example, then 6 can be written as $=10\bar{1}0 = 1 \times 2^3 + \bar{1} \times 2 = 8 - 2 = 6$

$$\begin{array}{r} 0101 \\ +01\bar{1}0 \\ \hline 11\bar{1}1 \end{array} \quad \textbf{(1)}$$

Thus no carry propagation was required. The above representation is possible using Binary Signed Arithmetic, for radix 2 & for more general case Signed number representation for radix greater than or equal to 2.

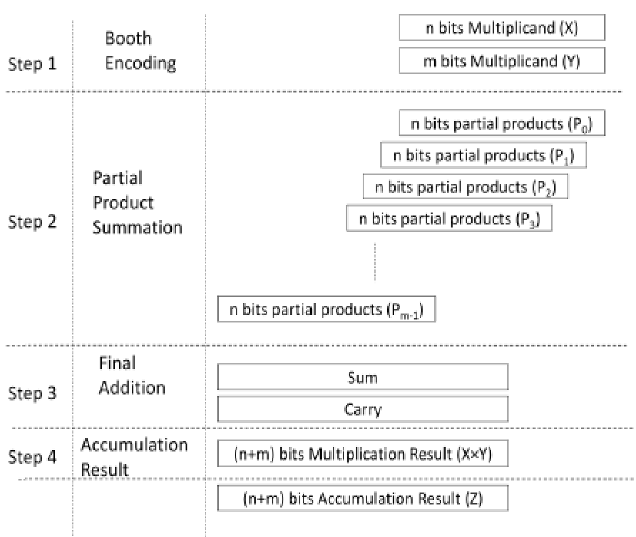

Fig.1. Basic Arithmetic steps of Multiplication and Accumulation.

II. EXISTING MULTIPLICATION METHOD

In Multiplier and Accumulation structure the multiplier can be divided into four operation steps. First step- booth algorithm, second step – partial product summation, third step – final addition, fourth step – accumulation as shown in Fig. 1.

Step1: Multiplication process done between n bits Multiplicand (X) and m bits Multiplier (Y).

Step 2: n bits Partial products (P0~Pj) will be generated after multiplying n bits and m bits.

Step 3: Final addition between Partial products Summation (S) bits and Carry(C) bits.

Step4: Accumulation results takes place between multiplication results (X*Y) and finally will get Accumulation Result (Z).

III. ALGORITHM OF THE MULTIPLIER

A. Radix - 4

Modified Booth Encoding (Radix-4) can effectively be applied to reduce the number of partial product rows to half in parallel multipliers. This is performed by grouping three adjacent multiplier bits (B = bn-1 bn-2…b0) to select one of the signed multiples as shown in Table I. The side bits of each group are overlapped with the two adjacent groups. The lastbit in each group is referred as reference bit. The first group is coded by adding "0" as reference bit prior to least significant bit position i.e., (b1, b0, 0).Depending on these select signals, the partial product rows are generated by selecting one of the combination {-2A,-A, 0, A, 2A} of the multiplicand (A = an-1an-2…a0). The twice of multiplicand (2A) in Table I is obtained by left shifting the multiplicand by one bit position and negation operation is achieved by inverting each bit of multiplicand (i.e. one's complement) and adding "1" to its least significant bit position.

TABLE I: Modified Booth's Encoding Unit For Radix 4

x_{i+1}	x_i	x_{i-1}	Y	Comments
0	0	0	0	String of Zeros
0	0	1	1.A	End of 1's
0	1	0	1.A	A single 1
0	1	1	2.A	End of 1's
1	0	0	-2.A	Beginning of 1's
1	0	1	-1.A	A single 0
1	1	0	-1.A	Beginning of 1's
1	1	1	0	String of ones

B. Radix - 2

The Multiplier uses a Redundant binary representation, which is one of the signed digit representation proposed byAvizienis to perform fast parallel arithmetic. It has a fixedradix-2 and $\{\bar{1}, 0, 1\}$as a digit set, where $\bar{1}$, represents -1. In redundant binary representation, a number can be expressed as

$$X = -(X_{n-1}).2^{n-1} + \sum_{i=0}^{n-2} X_i.2^i \quad \textbf{(2)}$$

which is similar to 2's complement representation of a number, except that $x_i$can take "$\bar{1}$"as value. Two Normal Binary digits can be coded to form a RB digit as shown in Table 2. The coding scheme for RB representation is given as

$$R_i = X_i^{+} - \overline{X_i^{-}} \quad \textbf{(3)}$$

Where, x_i+ and x_i- are represented in 2's complement form. This coding scheme can be applied to represent a RB numberas the summation of two NB numbers in one simple step. This is elaborated by thefollowing expressions. The summation of two NB numbers Xand Y can be represented as

$$X + Y = X - (Y) \quad \textbf{(4)}$$

In two's complement representation; -Y can be obtained by reversing all bits of Y and then adding "1" to its least significant bit position. This process is explained as follows:

$$-Y = \overline{Y} + 1 \quad \textbf{(5)}$$

Where $\bar{y}$is obtained by reversing all bits of Y. By Substituting above equations summation of two NB number can be expressed as

$$X + Y = \bar{X} - \bar{y} - 1 \quad \textbf{(6)}$$

Let, xi and yi be the ith digits of X and Y respectively. Also we define

$$(x_i, y_i) = x_i - \overline{y_i}$$
$$(X, Y) = X - \overline{Y} \quad \textbf{(7)}$$

Where $\bar{y}_i$, is the inversion of yi . The term xi -$\bar{y}_i$, will take one of the three values 1, 0, or $\bar{1}$, Instead of using the signed and absolute value representation, in this paper RB numbers are represented according to definition as follows (as tabulated in Table II.)

$$(1,0) = 1, (1,1) = (0,0) = 0 \, and \, (0,1) = \bar{1} \quad \textbf{(8)}$$

Thus a RB number can be form by a pair of two NB numbers (X ,Y) where two NB digits (xi ,yi) combine to form a single RB digit. From the expression above, (X , Y) can be equate to X+Y+1 , i.e.

$$X + Y = (X, \overline{Y}) - 1$$
$$= (X, \overline{Y}) + (0,1) \quad \textbf{(9)}$$

Since, from equation above. - 1 is equal to (0, 1)Thus, by inverting one of the two partial product rows and adding (0, 1) to its least significant bit position, a RB partial product row is generated, which is equivalent to the summation of two NB partial product rows. In 2's complement form sign bit of a number is represented by MSB, and hence prior to RB encoding scheme MSB of both the NB partial product rows are inverted. Both the MBE and RB Partial product generator produces incorrect results. One source of error is when MBE generates negative multiples of multiplicand. Negative partial product can be most efficiently generated by first bit reversing the multiplicand followed by the addition of "1" at its LSB position in the partial product summing tree, since in two's complement form negation of a number needs carry propagation addition. Also, another source of error is RB encoding scheme which requires the addition of "-1" at the LSB of partial product row to generate the exact summation of two NB partial product rows. A single error-correcting word, collectively representing the extra bits from both MBE and RB encoding together, eliminates all the possibilities of occurrence of inaccurate results.

TABLE II: RB Encoding

Xi^+	Xi^-	RB Digit
0	0	0
0	1	$\bar{1}$
1	0	1
1	1	0

The error correction word has the form

$$0E_M \; 0E_R \; 0E_M \; 0E_R \quad \textbf{(10)}$$

Where,

$E_R = \bar{1}$, when it corrects the term from RB Encoding alone

= 0, when it corrects the term from both RB encoding and MBE

$E_M = 1$, When the negative multiplicand is Generated due to MBE

= 0 ,otherwise

C. Radix 8

Radix-8 Booth recoding applies the same algorithm as that of Radix-4, but now we take quartets of bits instead of triplets. Each quartet is codified as a signed digit using Table III. Radix-8 algorithm reduces the number of partialproducts to n/3, where n is thenumber of multiplier bits. Thus it allows a time gain in the partial products summation.

Sign Extension Corrector: Sign Extension Corrector is designed to enhance the ability of the booth multiplier to multiply not only the unsigned number but as well as the signed number. The working principle of sign extension that converts signed multiplier signed unsigned multiplier as follows. One bit control signal called signed-unsigned(s_u) bit is used to indicate whether the multiplication operation is signed number or unsigned number .when sign-unsign s_u=0, it indicates unsigned number multiplication and when s_u=1, it indicates signed number multiplication.

TABLE III: Radix -8 Booth Recoding

Multiplier Bits Y_{i+2}	Y_{i+1}	Y_i	Y_{i-1}	Recoded Operation on multiplicand, X
0	0	0	0	0X
0	0	0	1	+X
0	0	1	0	+X
0	0	1	1	+2X
0	1	0	0	+2X
0	1	0	1	+3X
0	1	1	0	+3X
0	1	1	1	+4X
1	0	0	0	-4X
1	0	0	1	-3X
1	0	1	0	-3X
1	0	1	1	-2X
1	1	0	0	-2X
1	1	0	1	-1X
1	1	1	0	-1X
1	1	1	1	0X

TABLE IV: SUBE Operation

Sign-unsign	Type of operation
0	Unsigned multiplication
1	Signed multiplication

Partial Product Generator: A product formed by multiplying the multiplicand by one digit of the multiplier when the multiplier has more than one digit. Partial products are used as intermediate steps in calculating larger products. Partial product generator is designed to produce the product by multiplying the multiplicand A by 0, 1, -1, 2, -2,-3,-4, 3, 4. For product generator, multiply by zero means the multiplicand is multiplied by "0".Multiply by "1" means the product still remains the same as the multiplicand value. Multiply by "-1" means that the product is the two's complement form of the number. Multiply by "-2" is to shift left one bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by one place. . Multiply by "-4" is to shift left two bit the two's complement of the multiplicand value and multiply by "2" means just shift left the multiplicand by two place. Here we have an odd multiple of the multiplicand, 3Y, which is not immediately available. To generate it we need to perform this previous add: 2Y+Y=3Y. But we are designing a multiplier for specific purpose and thereby the multiplicand belongs to a previously known set of numbers which are stored in a memory chip. We have tried to take advantage of this fact, to ease the bottleneck of the radix-8 architecture, that is, the generation of 3Y. In this manner we try to attain a better overall multiplication time, or at least comparable to the time we could obtain using radix-4 architecture (with the additional advantage of using a less number of transistors). To generate 3Y with 8-bit words we only have to add 2Y+Y, that is, to add the number with the same number shifted one position to the left.

IV. DESIGN OF 64 BIT REDUNDANT BINARY MULTIPLIER

The Redundant Binary Representation Technique will eliminate the hard multiples generation problem by increasing the radix values.

A. Design of 64x64 based RB multiplier

The block diagram of 64*64 consists of 3 stages:

- Booth encoder and partial product generator stage (BEPPG stage)
- Redundant binary adder summing tree stage (RBA summing stage)
- Redundant binary to NB conversion stage (RB-to-NB stage)

Step 1: Booth Encoder and Partial Product Generator stage (BEPPG stage): Booth encoder and partial product generator affect the efficiency of the partial product generation. The number of partial products that can be saved by this stage impacts the cost, performance, and power consumption of the RB summing tree and the multiplier as a whole. In the first stage, 16 CRBBE-4 slices are used to generate the control signals from the multiplier. The hard multiple 5X is generated. The multiplicand bits are shifted and selected into 16 rows of RB partial products in 16 slices of RBPPG.

Step 2: Redundant Binary Adder summing tree stage (RBA summing stage): Here all the partial products generate 128 bits. These bits are added by the Redundant Binary Adders (RBA)denoted as1, 2, 3, 4. RBA had divided into sub blocks with 128 bits.

Step 3: Redundant binary to NB conversion stage (RB-to-NB stage): An RB-to-NB converter converts the final accumulation result to NB representation. Due to the unequal delay profile of the final RB result bits, the conversion can be carried out in uneven groups of consecutive digits according to their arrival time. The carry generation of the next group of digits can be evaluated with a carry-look ahead adder as they do not depend on the final summation results in the RBA tree stage as shown in Figs.2 and 3.

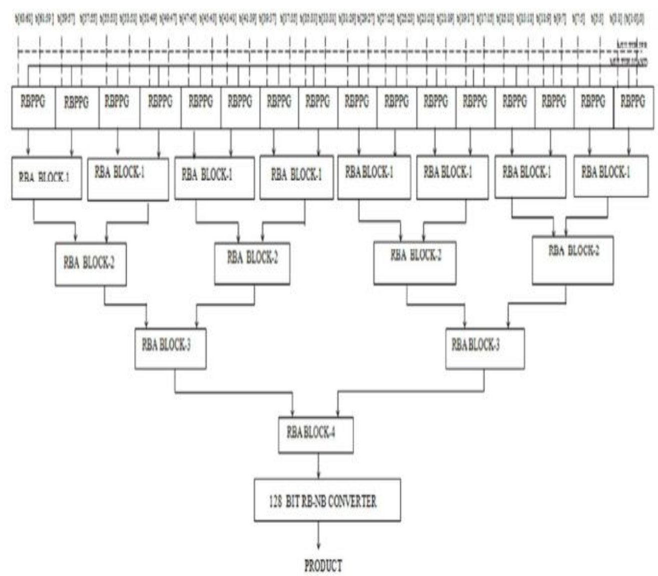

Fig. 2. Block Diagram 64x64 bit High Performance Redundant Binary Multiplier.

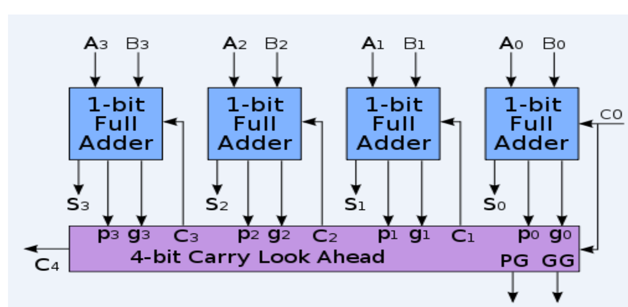

Fig.3. 4-Bit carrylook-ahead adder.

Table V compares the number of RBPP accumulation stages in Different 8bit RB multipliers i.e radix-2, radix -4 , radix -8 Multipliers.

TABLE V: Different RB Multipliers

parameter	Radix-2	Radix-4	Radix-8
Delay	4.270 ns	3.851 ns	4.044 ns
Number of slice LUTs	157	94	151
Number of bonded IOBs	32	32	32

V. RESULTS AND COMPARISON

This paper presents a high performance 64x64 bit. Redundant binary multiplier with modified redundant binary partial product generator, RB summing tree, and 128 RB to NB Converter. Multiplier designs are synthesized using Xilinx ISE 14.3 targeting a Xilinx Virtex-7 FPGA device using randomly generated input patterns. Top View module, RTL Schematic View are mentioned in Fig. 4 & Fig. 5 and Design Summary report of designed multiplier are given in Figs.6 and 7.

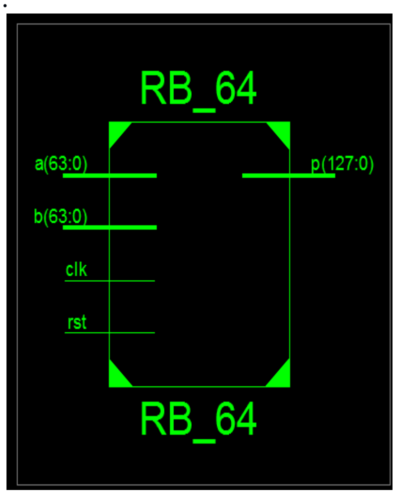

Fig.4. top view module.

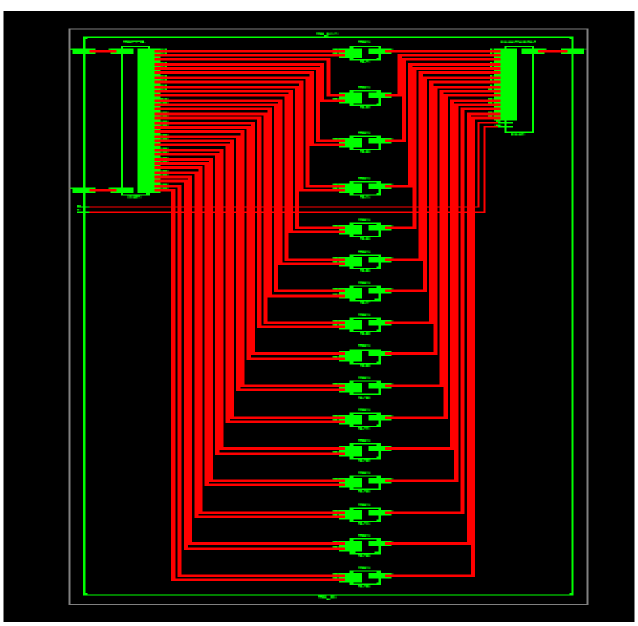

Fig.5. RTL schematic.

radix_8 Project Status (04/10/2017 - 15:51:45)			
Project File:	RB_64x64.xise	Parser Errors:	No Errors
Module Name:	RB_64	Implementation State:	Synthesized
Target Device:	xc7vx330t-3ffg1157	•Errors:	No Errors
Product Version:	ISE 14.3	•Warnings:	1 Warning (1 new)
Design Goal:	Balanced	•Routing Results:	
Design Strategy:	Xilinx Default (unlocked)	•Timing Constraints:	
Environment:	System Settings	•Final Timing Score:	

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	7583	204000	3%
Number of fully used LUT-FF pairs	0	7583	0%
Number of bonded IOBs	258	600	43%
Number of BUFG/BUFGCTRLs	1	32	3%

Fig.6.design summary report.

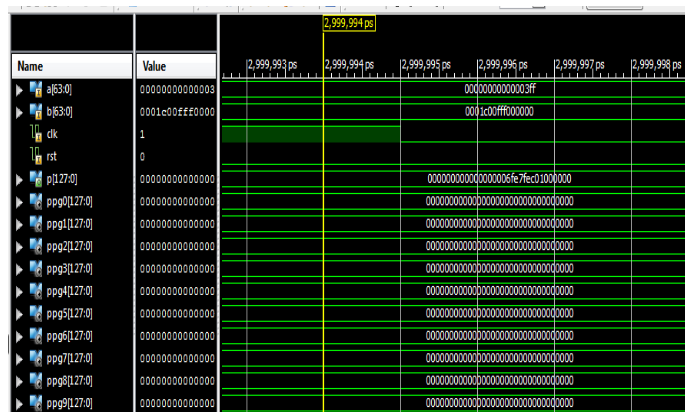

Fig.7. Simulation Results.

VI. CONCLUSION

The high performance 64x64 bit RB multiplier architecture has been designed in this paper, by considering a tradeoff between area, power and delay. The RBR technique removes hard multiples and reduces the partial products. By removing hard multiples in the Digital multiplier of the processor, Digital multiplier will perform much faster than existing method. Hence speed of the processor has been increased and it is used for higher multiplications by proposed method. In future RBR technique is used for Digital Signal Processing, Fast Fourier Transformations and multimedia applications.

VII. REFERENCES

[1] A. Avizienis, “Signed-digit number representations for fast parallel arithmetic,” IRE Trans. Electron. Computers, vol. EC-10, pp. 389–400, 1961.

[2] N. Takagi, H. Yasuura, and S. Yajima, “High-speed VLSI multiplication algorithm with a redundant binary addition tree,” IEEE Trans.Computers, vol. C-34, pp. 789-796, 1985.

[3] Y. Harata, Y. Nakamura, H. Nagase, M. Takigawa, and N. Takagi, “A high speed multiplier using a redundant binary adder tree,” IEEE J.Solid-State Circuits, vol. SC-22, pp. 28-34, 1987.

[4] H. Makino, Y. Nakase, and H. Shinohara, “A 8.8-ns 54x54-bit multiplier using new redundant binary architecture,” in Proc. Int. Conf. Computer Design (ICCD), pp. 202-205, 1993.
[5] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Makino, “An 8.8-ns 54×54-bit multiplier with high speed redundant binary architecture,” IEEE J. Solid-State Circuits, vol. 31, pp. 773-783, 1996.
[6] Y. Kim, B. Song, J. Grosspietsch, and S. Gillig, “A carry-free 54b×54b multiplier using equivalent bit conversion algorithm,” IEEE J. Solid- State Circuits, vol. 36, pp. 1538–1545, 2001.
[7] W. Yeh and C. Jen, “High-speed Booth encoded parallel multiplier design,” IEEE Trans. Computers, vol. 49, pp. 692-701, 2000.
[8] S. Kuang, J. Wang, and C. Guo, “Modified Booth multiplier with a regular partial product array,” IEEE Trans. Circuits Syst. II, vol. 56, pp. 404-408, 2009.
[9] J. Kang and J. Gaudiot,“A simple high speed multiplier design,” IEEE Trans. Computers, vol. 55, pp.1253-1258, 2006.
[10] Y. He and C. Chang, “A new redundant binary Booth encoding for fast 2-bit multiplier design,” IEEE Trans. Circuits Syst.I,Reg. Papers,vol. 56, pp. 1192–1199, 2009.
[11] F. Lamberti, N. Andrikos, E. Antelo, and P. Montuschi, “Reducing the computation time in (short bit-width) two’s complement multipliers,” IEEE Trans. Computers, vol. 60, pp. 148- 156, 2011.
[12] Xiaoping Cui, Weiqiang Liu, Xin Chen, Earl E. Swartzlander, and Fabrizio Lombardi, Modified Partial Product Generator for Redundant Binary Multipliers”, IEEE Transactions onComputers,vol.65,pp. 1165-1171, June 2015.

Author’s Profile:

Y.Krishna currently doing B.tech Degree in (ECE) from N.B.K.R. Institute of science & Technology, Vidyanagar. His General Area of Interest is switching theory and logical design, embedded systems and Electronic devices and circuits.

N.Benarji currently doing B.tech Degree in (ECE) from N.B.K.R. Institute of science & Technology, Vidyanagar. His General Area of Interest is signals and systems, pulse and digital circuits and Electronic devices and circuits.

P.Vali BashaMtech., He received his Master of Technology degree from JNTUA.Currently working as Assistant Professor in ECE department of N.B.K.R. Institute of Science & Technology, Vidyanagar, A.P. India. He has teaching experience in the stream of engineering education. He has published in International Journals and National Conferences. He has attended many workshops and Seminars. His research areas are Low Power VLSI, Digital IC Design, Signal processing, and image processing and communications systems.