International Journal of Research
Available at https://edupediapublications.org/journals

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue-01
January 2018

Design of Fast and Low Area Pre-Encoded Multipliers Based on NR8SD Encoding technique for DSP/Multimedia applications

Saladi Balaji Gupta & Palivela Soundarya Mala
saladibalajigupta@gmail.com[1] & palivela.soundarya@gmail.com[2]
[1]PG Scholar, Dept of ECE, Godavari Institute of Engineering And Technology, Rajahmundry, East Godavari, Andhra Pradesh.
[2] Associate Professor, Dept of ECE , Godavari Institute of Engineering And Technology, Rajahmundry, East Godavari, Andhra Pradesh.

Abstract: *In this paper, we introduce an architecture of Non-Redundant radix-8 Signed-Digit (NR8SD) encoding technique, which uses the digit values{-3,-2,-1,0,+1,+2,+3,+4} or {-4,-3,-2,-1,0,+1,+2,+3} is proposed leading to a multiplier design with less complex partial products implementation compared to NR4SD multiplier.*

NR4SD multiplier is also more area efficient compared to MB multiplier, which uses the digit values {-2,-1,0,+1} or{-1,0,+1,+2} that leads to fast multiplication than the conventional Modified Booth Multiplier. These pre-encoded multipliers are based on off-line encoding of generating less number of coefficients for DSP applications.

Keywords: Modified Booth Encoding, Non Redundant Radix-4 Signed Digit (NR4SD)pre encoding multiplication by constants, Non Redundant Radix-8 Signed Digit (NR8SD)pre encoding multiplication ,common sub expressions sharing, Add-Multiply operation, arithmetic circuits.

I. INTRODUCTION

Now a days electronics was upgraded continuously , particularly in digital signal processing applications and multimedia applications. Now a day's they are growing day by day. In this DSP and Multimedia applications it requires so many arithmetic multiplications . This multiplication operations are performed by different types of multipliers. Multipliers play vital role in most of the high performance systems. A Digital multiplier is the basic component in general purpose microprocessor and in DSP. In DSP methods we use Fourier Transformations and discrete cosine transformations in discrete wavelet transformations. As we compared to other arithmetic operations multiplication is more delayed and power consuming. The main challenge in designing of multiplier is to reduce the delay and minimize the area with efficient power usage. To accomplish this test by decreasing the quantity of partial products. Although reducing the number of partial products by using higher radix booth encoder, but the number of hard multipliers that are costly to generate and also increases concurrently. By utilizing the Canonic marked Digit representation (CSD) we can encode minimum non-zero digit coefficients[1]. CSD multipliers will gives the less non-zero partial products,, which thusly diminishes their switching action. Be that as it may, the CSD encoding additionally have different constraints. Folding technique [2], which limit silicon zone by time multiplexing operations into single practical units, e.g., adders, multipliers, isn't conceivable as the CSD based multipliers are hard-wired to particular co-efficients. In [3], CSD-based programmable multiplier arrangement was proposed for social events of pre-chosen coefficients that give particular features . The degree of ROM used to accumulate the get-togethers of coefficients. It diminishes the area and power usage of the circuit. In any case, this multiplier design isn't versatile. since the fractional items age unit is intended for a specific gathering of coefficients and can't be reused for different gatherings. Also, this method is difficult extended to huge groups of pre-determined coefficients attaining at the same time high efficiency. Modified Booth (MB) encoding [4]–[7] solve the illuminate the previously mentioned limitations and decries the count of the partial products to half. By reducing the partial products to reduce the delay and minimize the area with efficient power usage.. In [8], Kim et al. proposed a procedure like [3], for laying out more capable MB multipliers for social affairs of pre-chosen coefficients with measure up to obstructions indicated in the before section. In [9], [10], multipliers are incorporated in butterfly units of FFT processors utilize standard coefficients put away in ROMs. The values of constant coefficients are recognized in advance, we encode the coefficients based on off-line encoding technique in MB encoding and store the resultant coefficients (i.e., 3 bits per digit) into a ROM. Utilizing this technique[09]– [10], we don't require any encoding circuit of the MB multiplier. We pass on to this plan as pre-encoded MB multiplier. At that point, we take a gander at the Non-Redundant radix-4 Signed-Digit (NR4SD) encoding configuration is expanding the serial encoding procedures of [6], [10]. In this

NR4SD encoding technique utilizes one of the accompanying arrangements of digit values{-2,-1,0,+1} or{-1,0,+1,+2} for cover the dynamic scope of the 2's complement shape, the digits are encoded by the NR4SD frame aside from MSB is MB encoded. We are encoded standard coefficients by utilizing encoding formula and store them in to a ROM in a lessened frame (i.e.,2 bits per digit). Which encoded coefficients require 3 bits per digit, it reduces memory size. It is also having some delay and required more area. To overcome this limitations we proposed NR8SD encoding technique . It uses one of the following sets of digit values {-3,-2,-1,0,+1,+2,+3,+4} or {-4,-3,-2,-1,0,+1,+2,+3} for cover the dynamic range of the 2's complement form the digits are encoded by the NR8SD frame aside from MSB is MB encoded.. We are encoded standard coefficients by utilizing encoding equation and store them in to a ROM in a diminished frame (i.e.,3 bits per digit).Compared to the NR4SD multiplier in which the encoded coefficients require 4 bits for every digit, the proposed NR8SD method lessens the measure of the memory. In NR8SD encoding we use five digit values. Thus, the NR8SD-based pre-encoded multipliers having a less complex partial products generation unit. We take a gander at the productivity of the previously mentioned pre-encoded multipliers thinking about the measure of the coefficients' ROM.

II. DIFFERENT MULTIPLIERS

Binary Multiplication

In the binary number formation digits, called bits, are formed to the set. The consequence of multiplying any binary number by a single binary bit is whichever 0, or the first number. This makes shaping the partial-products simple and efficient. Including these fractional items is the time taken assignment for twofold multipliers. One coherent process is to shape the partial-products each one in turn and add them as they are produced. Regularly actualized by programming on processors that don't have any equipment multiplier, this procedure working great, yet it is moderate. Because it requires minimum one machine cycle to add the every additional partial products. For applications where this marvels does not give required execution, multipliers can be actualized specifically in equipment.

Hardware Multipliers

Equipment usage of move and include multipliers are give better execution over programming execution, however are still moderate. In this multipliers are create one partial product for each bit. Each extra partial -product is summed a carry must be propagated from the least significant bit (LSB) to the most significant bit (MSB). This phenomena is more time consuming and repeated procedure for adding every partial products. To overcome this time delay, encoding technique is used. This encoding method is diminishing the quantity of fractional items to be summed and expanding the execution. Simply such a system was first proposed by Booth . The genuine Booth's calculation transports over bordering strings of l's by utilizing the property that: 2" + 2(n-1) + 2(n-2) + . . . + 2hm) = 2(n+l) - 2(n-m). Booth's algorithm produces half of the encoded partial products from an N bit operand, designers are develop the various modifications in Booth algorithm by increasing or decreasing the number of partial products generation. After that resultant encoded incomplete items are included by utilizing reasonable technique. Altered 2-bit Booth encoding is utilized on most present day skimming point chips LU 881, MCA 861. In Booth calculation couple of engineers are attempting to create 3 bit Booth encoding. This encoding diminishes the quantity of fractional items to be summed by a factor of three IBEN 891. This 3 bit encoding is requires 3x multipliers for convey engender expansion..To achieve fast and better performance hardware multipliers by finding the more efficient methods for adding the partial products. Eliminating the delay in carry propagate additions to accomplish this ,They adding the partial -products in a redundant number representation. The advantage of a redundant illustration is that two numbers, or partial -products, can be added together without propagating a carry across the whole size of the number. One commonly used illustration is known as carry-save form. In this repetitive delineation two bits, known as the convey and whole, are utilized to speak to each piece position. When two numbers in carry -save form are added together any carries that result are no way to propagated more than one bit position. This addition of two numbers is done by carry-save form. It is faster then normal binary addition. One conventional technique that has been created for summing lines of partial products utilizing a convey spare portrayal is the cluster multiplier.

III. EXISTING METOD

Conventional MB Multiplier Design:

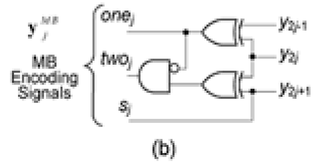
Fast multiplication operation is required in DSP and generally used processors . it is most important requirements in DSP. To increase this speed by using fast multipliers. High speed multiplication is achievable by way of series of additions, subtractions and shift operations. Required many frequent of additions for multiplication. The number to be incorporated is the multiplicand, the conditions that it is incorporated is the multiplier, and the result is the item. Every step in this addition produce a partial product. In may PC's , the operand having same length of bits. Exactly when the operands are deciphered as entire numbers, the thing is generally twofold the length of operands with a particular true objective to spare the information content. This frequent addition method that is proposed by the arithmetic particulars. It is delayed process , so it replace number of times by different types of algorithms that makes use of positional illustration. It is conceivable to depreciate multipliers into two sections. . The initial segment is devoted to produce the partial products and the second one gathers and adding them. The basic multiplication principle is two overlap i.e. evaluation of partial products and aggregation of the shifted partial products. It is performed by the accompanying increments of the sections of the moved partial products matrix. Multiplicand provide the proper bit to the multiplier for shift operation. The deferred, gated example of the multiplicand should all be in the equivalent segment of shifted partial product matrix. After added to shaping the product bit for the correct frame. Multiplication is Increase is thusly a multi operand operation. To stretch out the increase to both marked and unsigned.

Hear, driven to an adder and then the input X and the sum Y=A+B are drives to the a multiplier for obtaining the Z. the main imitation in the adder is major delay in the difficult path .This path depends on width of the input bits.

TABLE I
MODIFIED BOOTH ENCODING TABLE.

Binary			y_j^{MB}	MB Encoding			Input Carry
y_{2j+1}	y_{2j}	y_{2j-1}		sign$=s_j$	$\times1=one_j$	$\times2=two_j$	$c_{in,j}$
0	0	0	**0**	0	0	0	0
0	0	1	**+1**	0	1	0	0
0	1	0	**+1**	0	1	0	0
0	1	1	**+2**	0	0	1	0
1	0	0	**-2**	1	0	1	1
1	0	1	**-1**	1	1	0	1
1	1	0	**-1**	1	1	0	1
1	1	1	**0**	1	0	0	0

$$one_j = y_{2j-1} \oplus y_{2j}$$
$$two_j = (y_{2j+1} \oplus y_{2j}) \cdot \overline{one_j}$$
$$s_j = y_{2j+1}$$



(a)

Fig.1 (a)Boolean equations and (b)gate level representation for the execution of the MB encoding signals.

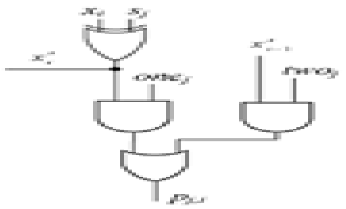


Fig. Production of the i-th Bit Pj,i of PPj for conventional MB Multiplier

For the calculation of the least and the most significant bits of the partial product we consider and correspondingly. Note that in the event that that , the quantity of the resulting partial products is and the most significant MB digit is shaped in light of sign augmentation of the underlying 2's complement number.

After this generated partial products are added and correctly weighted, through a Carry-Save Adder (CSA) tree with the Correction Term (CT) which is given by the accompanying equations:

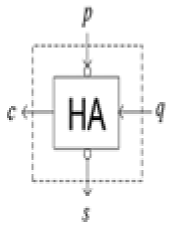
$$Z = X \cdot Y = CT + \sum_{j\ 0}^{k\ 1} PP_j \cdot 2^{2j}$$

HA*

$$c = p \vee q$$
$$s = p \oplus q$$

(a)

HA**



$$c = \overline{p} \wedge q$$
$$s = p \oplus q$$

(b)

Fig.3 Boolean conditions and schematics for marked (a) HA* (b) HA**

$$CT = CT(low) + CT(high) =$$
$$= \sum_{j=0}^{k-1} c_{in,j} \cdot 2^{2j} + 2^{n}\left(1 + \sum_{j=0}^{k-1} 2^{2j+1}\right)$$

where $c_{in,j} = (one_j \vee two_j) \wedge s_j$ (see Table I).

At long last, Output of the CSA tree is given to a quick Carry Look Ahead (CLA) adder to frame the ultimate result is Z = X . Y as appeared in Fig.3 (a).

Conventional MB Multiplier

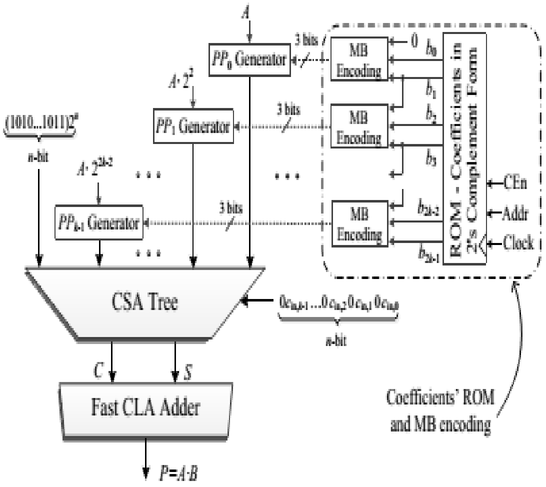


Fig. 4. System Architecture of the Conventional MB Multiplier.

Fig. 4 Displays the outline of the framework which contains the conventional MB multiplier and the ROM with coefficients in 2's supplement format. Let us assume the multiplication A.B. The coefficients B=(bn-1…b0) consists of n=2k bits and is motivated to the MB encoding blocks from a ROM. It stored the form of 2's complement form is there. It is encoded according to the MB algorithm (Section 2) and multiplied by A = (an …. a0) , which is in the form of 2's complement. We take note of that the ROM information transport width squares with the width of coefficient B (n bits) and that it yields one coefficient on each clock cycle.. The k number of partial products are produced as per following equation:

$$PP_j = A \cdot \mathbf{b}_j^{MB} = \bar{p}_{j,n}2^n + \sum_{i=0}^{n-1} p_{j,i}2^i.$$

The production of the ith bit pj;i of the partial product P Pj is described at gate level in Fig. 4a For the calculation of the LSB and MSB of P Pj ,we consider a 1=0 and an = an 1, in that order. In the wake of forming the partial products, they are adding and appropriately weighted by utilizing a Carry Save Adder (CSA) tree alongside the correction term (COR):

$$P = A \cdot B = COR + \sum_{j=0}^{k-1} PP_j 2^{2j},$$

$$COR = \sum_{j=0}^{k-1} c_{in,j}2^{2j} + 2^n(1 + \sum_{j=0}^{k-1} 2^{2j+1}),$$

Where cin;j = (onej _twoj)^sj (Table 1). The , Output of the CSA tree is given to a quick Carry Look Ahead (CLA) adder to shape the ultimate result is P= A.B (Fig. 4).

Pre-Encoded MB Multiplier Design In the pre-encoded MB multiplier plot, the coefficient B is encoded detached by the conventional MB algorithm (Table 1).The subsequent encoding signs of B are put away in a ROM. The hovered some portion of Fig. 4, which contains the ROM with coefficients in 2's supplement shape and the MB encoding circuit, is presently completely supplanted by the ROM of Fig. 5. The MB encoding squares of Fig. 4 are excluded The new ROM of Fig. 5 is used to store the encoding indications of B and energize them into the partial product generators (PPj Generators - PPG) on each clock cycle. Focusing to diminish switching movement, the esteem '1' of s j in the last passage of Table 1 is supplanted by '0'. The sign s j is presently given by the relation:

$$s_j = b_{2j+1} \oplus (b_{2j+1} \wedge b_{2j} \wedge b_{2j-1}).$$

Therefore, the PPG of Fig. 3a is supplanted by the one of Fig. 3b. Compared to leads to a more complex design. However,pre-encoding technique is achives no area / delay transparency at the circuit.

The incomplete items, authentically weighted, and the modification term (COR) of (11) are fed into a CSA tree. The info convey Cin,j of (11) is figured as Cin,j = sj in light of (12) and Table 1. The CS yield of the tree is at last converged by a quick CLA adder.

Be that as it may, the ROM width is expanded. Every digit demands three encoding bits (i.e., s, two and one (Table 1)) to be put away in the ROM. Since the n-bit coefficient B required three bits for every digit whenever encoding the in MB algorithm, the ROM width requisite is 3n/2 bits for each coefficient. Therefore, the width and the general size of the ROM are expanded by half contrasted with the ROM of the conventional form (Fig. 4).

Pre-Encoded NR4SD Multipliers Design

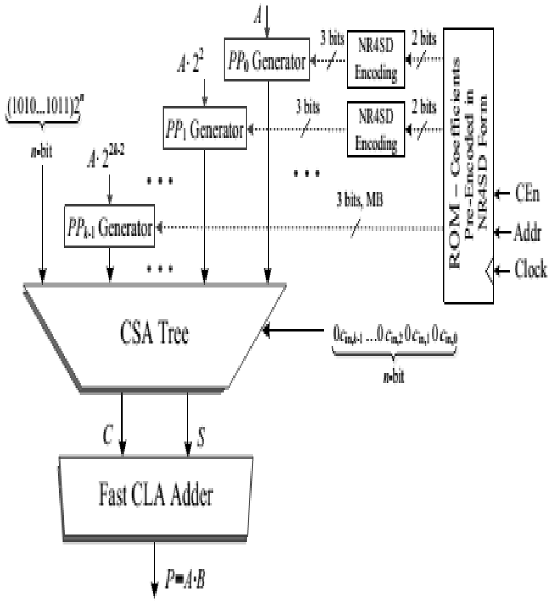


Fig.5 System Architecture of the NR4SD Multipliers

The system design for the pre-encoded NR4SD multipliers is displayed in Fig. 5. Two bits are now stored in ROM: n2j+1, n+2j(Table 2) for the NR4SD or n+2j+1, n2j(Table 3) for the NR4SD+form. Such a way, we decrease the memory necessity to +1 bits for every coefficient while the relating memory required for the pre-encoded MB algorithm is 3n/2 bits for each coefficient. Along these lines, the measure of put away bits is equivalent to that of the conventional MB configuration, aside from the most significant digit that needs an additional bit as it is MB encoded. Appeared differently in relation to the pre-encoded MB multiplier, where the MB encoding units are disposed of, the pre-encoded NR4SD multipliers require extra equipment to make the indications of (6) and (8) for the NR4SD and NR4SD+ algorithm. respectively. Every partial product of the pre-encoded NR4SD and NR4SD+ multipliers is implemented based on below Fig. 8c and 8d, respectively, excluding the PPk 1 that relates to the most significant digit. As this digit is in MB form, we utilize the PPG of Fig. 8b applying the change specified in Section 4.2 for the s j bit. The partial products, legitimately weighted, and the rectification term (COR) of (11) are encouraged into a CSA tree. The input carry cin;j of (11) is calculated as cin;j = twoj_ onej and cin;j = onej for the NR4SDand NR4SD+pre-encoded multipliers, respectively ,based on Tables 2 and 3. The convey spare yield of the CSA tree is at long last summed utilizing a quick CLA adder.Give us a chance to consider the increase of 2's supplement numbers X and Y with each number having of n=2k bits .The multiplicand Y can be represented in MB algorithm.

$$Y \ \langle y_{n-1}y_{n-2}\cdots y_1y_0\rangle_{2's} \ \ y_{2k-1}\cdot 2^{2k-1}+\sum_{i\ 0}^{2k\ 2} y_i\cdot 2^i$$

$$-\langle y_k^{MB}y_{k\ 2}^{MB}\cdots y_1^{MB}y_0^{MB}\rangle_{MB}-\sum_{j\ 0}^{k\ 1} y_j^{MB}\cdot 2^{2j} \qquad (1)$$

$$y_j^{MB} = -2y_{2j\ 1}+y_{2j}-y_{2j\ 1}. \qquad (2)$$

NR4SD − Algorithm:

Step 1: Consider the initial values j = 0 and C0=0.

Step 2: Determine the carry C2j+1 and the sum $n2j$ + of a Half Adder (HA) with inputs $b2j$ and $c2j$ (Fig. 6a).

$$c_{2j+1} = b_{2j} \wedge c_{2j}, \quad n_{2j}^{+} = b_{2j} \oplus c_{2j}.$$

Step 3: Determine the positively signed carry $c2j$+1(+) and the negatively signed sum $n2j$+1 − (−) of a Half Adder* (HA*) with inputs $b2j$+1(+) and $c2j$+1(+) (Fig. 6a). The outputs $c2j$+1and n2j+1of the HA* narrate to its inputs as follows:

$$2c_{2j+2} - n_{2j+1}^{-} = b_{2j+1} + c_{2j+1}.$$

The subsequent Boolean equations review the HA* operation:

$$c_{2j+2} = b_{2j+1} \vee c_{2j+1}, \quad n_{2j+1}^{-} = b_{2j+1} \oplus c_{2j+1}.$$

Step 4: Determine the value of the bNRj digit.

$$\mathbf{b}_j^{NR-} = -2n_{2j+1}^{-} + n_{2j}^{+}.$$

Equation (5) outcome from the fact that $n2j$+1 − is negatively signed and $n2j$ + is positively signed.

Step 5: j := j + 1.

Step 6: If (j < k-1), go to Step 2. If (j = k-1), encode the MSB depends on the MB algorithm and allowing for the three consecutive bits to be $b2k$−1 , $b2k$−2and $c2k$−2 (Fig. 6b). If (j = k),

Table 2 shows the formation for NR4SD_digits.. Equations (6)show how the NR4SD_ encoding signals one_j^{+} , one_j^{-} and two_j^{-} of Table 2 are generated

$$\begin{aligned} one_j^{+} &= \overline{n_{2j+1}^{-}} \wedge n_{2j}^{+}, \\ one_j^{-} &= n_{2j+1}^{-} \wedge n_{2j}^{+}, \\ two_j^{-} &= n_{2j+1}^{-} \wedge \overline{n_{2j}^{+}}. \end{aligned}$$

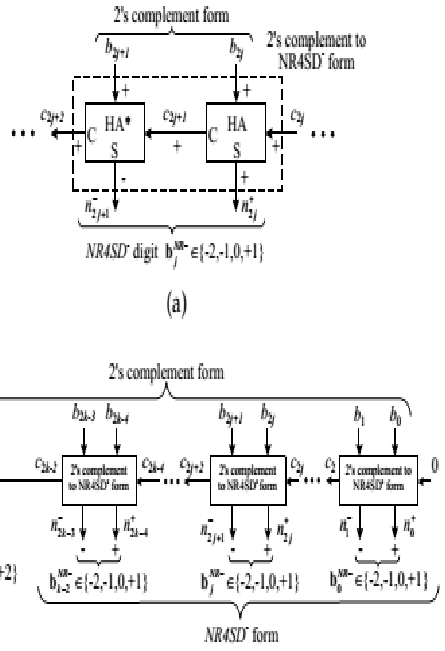

2's complement form
2's complement to NR4SD⁻ form
b_{2j+1}
b_{2j}
c_{2j+2}
c_{2j+1}
c_{2j}
HA*
HA
C
S
n_{2j+1}^{-}
n_{2j}^{+}
NR4SD⁻ digit $\mathbf{b}_j^{NR-} \in\{-2,-1,0,+1\}$
(a)
2's complement form
b_{2k-1} b_{2k-2}
b_{2k-3} b_{2k-4}
b_{2j+1} b_{2j}
b_1 b_0
c_{2k-2}
c_{2k-4}
c_{2j+2}
c_{2j}
c_2
0
2's complement to NR4SD⁻ form
n_{2k-3}^{-} n_{2k-4}^{+}
n_{2j+1}^{-} n_{2j}^{+}
n_1^{-} n_0^{+}
$b_{k-1}^{MB}\in\{-2,-1,0,+1,+2\}$
MB form
$b_{k-2}^{NR-}\in\{-2,-1,0,+1\}$
$b_j^{NR-}\in\{-2,-1,0,+1\}$
$b_0^{NR-}\in\{-2,-1,0,+1\}$
NR4SD⁻ form

Fig. 6. Block Diagram of the NR4SD- Encoding Scheme at the (a) Digit and (b) Word Level.

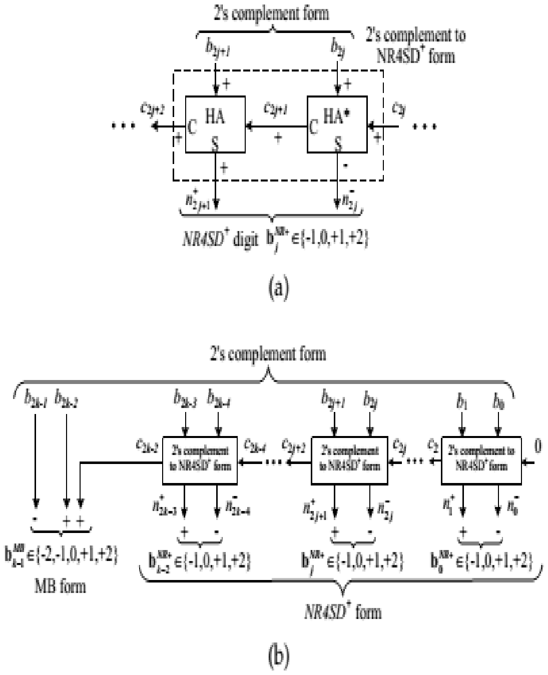

2's complement form
b_{2j+1}
b_{2j}
2's complement to NR4SD+ form
c_{2j+2}
HA
c_{2j+1}
HA*
c_{2j}
n^+_{2j+1}
n^-_{2j}
NR4SD+ digit $b_j^{NR+} \in\{-1,0,+1,+2\}$
(a)
2's complement form
b_{2k-1} b_{2k-2}
b_{2k-3} b_{2k-4}
b_{2j+1} b_{2j}
b_1 b_0
2's complement to NR4SD+ form
$b_{k-1}^{MB} \in\{-2,-1,0,+1,+2\}$
MB form
$b_{k-2}^{NR+} \in\{-1,0,+1,+2\}$
$b_j^{NR+} \in\{-1,0,+1,+2\}$
$b_0^{NR+} \in\{-1,0,+1,+2\}$
NR4SD+ form
(b)

Fig. 7. Block Diagram of the NR4SD+ Encoding format at the (a) Digit and (b) Word Level.

TABLE 2

NR4SD⁻ Encoding

2's complement			NR4SD⁻ form			Digit	NR4SD⁻ Encoding		
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^-_{2j+1}	n^+_{2j}	b_j^{NR-}	one_j^+	one_j^-	two_j^-
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	+1	1	0	0
0	1	0	0	0	1	+1	1	0	0
0	1	1	1	1	0	-2	0	0	1
1	0	0	1	1	0	-2	0	0	1
1	0	1	1	1	1	-1	0	1	0
1	1	0	1	1	1	-1	0	1	0
1	1	1	1	0	0	0	0	0	0

TABLE 3

NR4SD⁺ Encoding

2's complement			NR4SD⁺ form			Digit	NR4SD⁺ Encoding		
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^+_{2j+1}	n^-_{2j}	b_j^{NR+}	one_j^+	one_j^-	two_j^+
0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	1	+1	1	0	0
0	1	0	0	1	1	+1	1	0	0
0	1	1	0	1	0	+2	0	0	1
1	0	0	0	1	0	+2	0	0	1
1	0	1	1	0	1	-1	0	1	0
1	1	0	1	0	1	-1	0	1	0
1	1	1	1	0	0	0	0	0	0

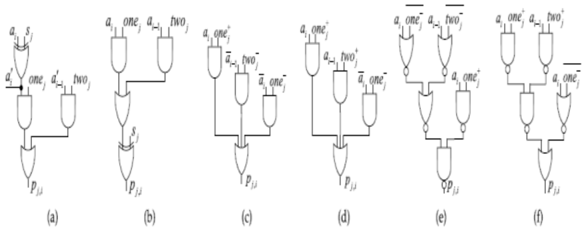


Fig. 8. Generation of the ith Bit pj;i of PPj for a) Conventional, b) Pre-Encoded MB Multipliers, c) NR4SD+, d) NR4SD+ Pre-Encoded Multipliers, and e) NR4SD-, f) NR4SD- Pre-Encoded Multipliers after reconstruction.

IV. PROPOSED SYSTEM

The Non-Redundant radix-8 Signed-Digit (NR8SD) encoding technique, which utilizes the digit values {-3,- 2,-1,0,+1,+2,+3,+4} or {-4,- 3,- 2,-1,0,+1,+2,+3} is proposed prompting a multiplier plan with less complex partial products execution. General experimental examination checks the proposed pre-encoded NR8SD multipliers, having with coefficients memory, are area and time friendly than compare to the Non-Redundant radix-4 Signed-Digit (NR4SD) encoding. Then, At that point, we investigate a Non-Redundant radix-8 Signed-Digit (NR8SD) encoding plan broadening the serial encoding systems Utilizing this system, the encoding circuit of the MB multiplier is precluded. We suggest to this design as pre-encoded MB multiplier.

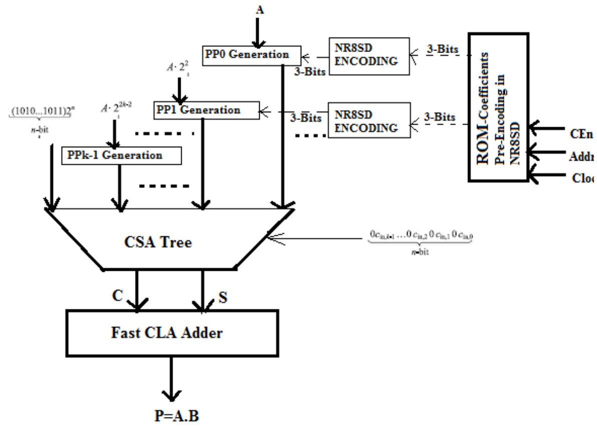


Fig.9: System Architecture of the NR8SD Multipliers

Table 4.Radix-8 Booth Encoding:

Input Bits of Multiplier				Partial Product
X(2)	X(1)	X(0)	X(-1)	Y
0	0	0	0	0
0	0	0	1	+1
0	0	1	0	+1
0	0	1	1	+2
0	1	0	0	+2
0	1	0	1	+3
0	1	1	0	+3
0	1	1	1	+4
1	0	0	0	-4
1	0	0	1	-3
1	0	1	0	-3
1	0	1	1	-2
1	1	0	0	-2
1	1	0	1	-1
1	1	1	0	-1
1	1	1	1	0

In this proposed system is more area and time efficient then compare to existing system. This efficiency is achieved by reducing the partial products. In radix-8 algorithm input bits encoded by grouping the three bits at a time to get one partial product. For Example if we have multiplier length with 24-bit then we have to prepare 8 partial products only using NR8SD pre-encoded multiplier while we are preparing 12 partial products in NR4SD multiplier.

In this system encoding module getting inputs from ROM coefficients pre encoded module.NR8SD encoding module are encoding the input bits according to the radix-8 table. This are generated the encoded bits. This encoded bits going to partial products generation PPg block. In this partial product generator are generate the partial products and send to CSA tree. This CSA Tree are performing the adding operation and generate one sum row and one carry row. In existing method more Number of partial products are produced it leads to increase the delay. Hear less number of partial products are generated. So, Achieving the less delay . After this sum and Carry are propagated to Fast CLA adder .In this CLA having Speed due to computing carry bit i without waiting for carry bit i – 1. This are producing the final output P=A.B.

Table 5.$NR8SD^+ \ FORM$

2'COMPLEMENT				$NR8SD^+$ $FORM$			Digit
b_{2j+2}	b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^+_{2j+1}	n^-_{2j}	b_{2j}
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	+1
0	0	1	0	0	1	1	+2
0	0	1	1	0	1	0	+2
0	1	0	0	0	1	0	+2
0	1	0	1	1	0	1	+3
0	1	1	0	1	0	1	+3
0	1	1	1	1	0	0	+4
1	0	0	0	0	0	0	+4
1	0	0	1	0	1	1	-3
1	0	1	0	0	1	1	-3
1	0	1	1	0	1	0	-2
1	1	0	0	0	1	0	-2
1	1	0	1	1	0	1	-1
1	1	1	0	1	0	1	-1
1	1	1	1	1	0	0	0

Table 6:NR8SD¯ FORM:

2'COMPLEMENT				$NR8SD^-$ $FORM$			Digit
b_{2j+2}	b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^+_{2j+1}	n^-_{2j}	b_{2j}
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	+1
0	0	1	0	0	1	1	+2
0	0	1	1	0	1	0	+2
0	1	0	0	0	1	0	+2
0	1	0	1	1	0	1	+3
0	1	1	0	1	0	1	+3
0	1	1	1	1	0	0	-4
1	0	0	0	0	0	0	-4
1	0	0	1	0	1	1	-3
1	0	1	0	0	1	1	-3
1	0	1	1	0	1	0	-2
1	1	0	0	0	1	0	-2
1	1	0	1	1	0	1	-1
1	1	1	0	1	0	1	-1
1	1	1	1	1	0	0	0

The outputs *c*2*j*+1and n2j+1of the HA* relate to its inputs as follows:

$$2c_{2j+2} - n^-_{2j+1} = b_{2j+1} + c_{2j+1}.$$

The following Boolean equations summarize the HA* operation

$$c_{2j+2} = b_{2j+1} \vee c_{2j+1}, \quad n^-_{2j+1} = b_{2j+1} \oplus c_{2j+1}.$$

Calculate the value of the bNRj digit.

$$\mathbf{b}_j^{NR-} = -2n^-_{2j+1} + n^+_{2j}.$$

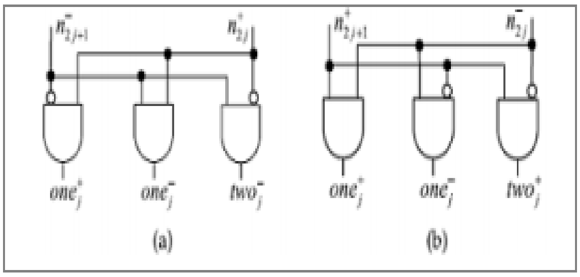


Fig.10. Additional circuit needed in the NR8SD multipliers to complete the (a) NR8SD– and (b) NR8SD+encoding.

This proposed system are used in Digital Signal processing applications. In such a way fast multiplication operation is being achieved by NR8SD by reducing number of partial products.

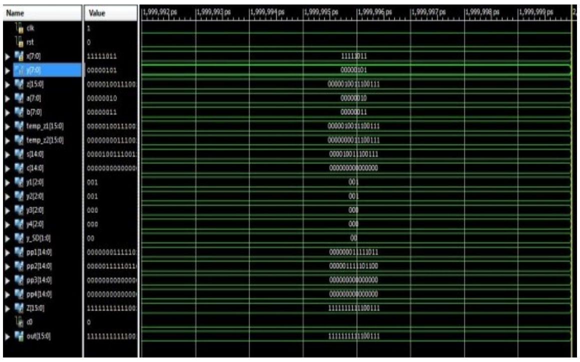
V. EXPERIMENTAL RESULTS

Existing system results

The simulation of the program is done using ModelSim tool and Xilinx ISE Design Suite 13.2. The results for the multiplication of 4x4 and 8x8 using Modified Booth Multiplier is shown in this section.

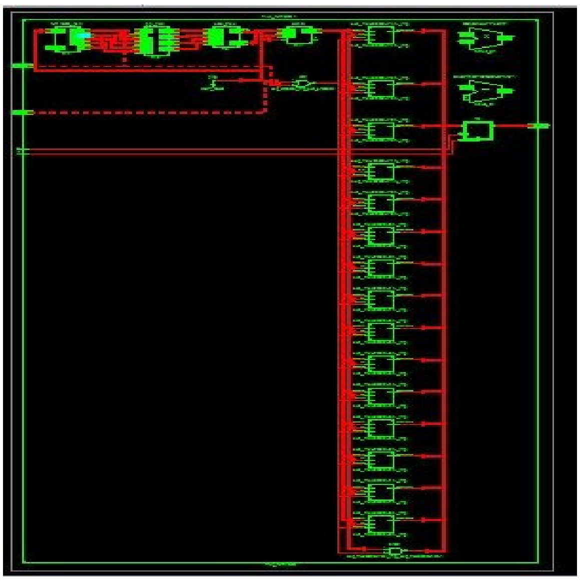
Synthesis Results:

Simulation output:

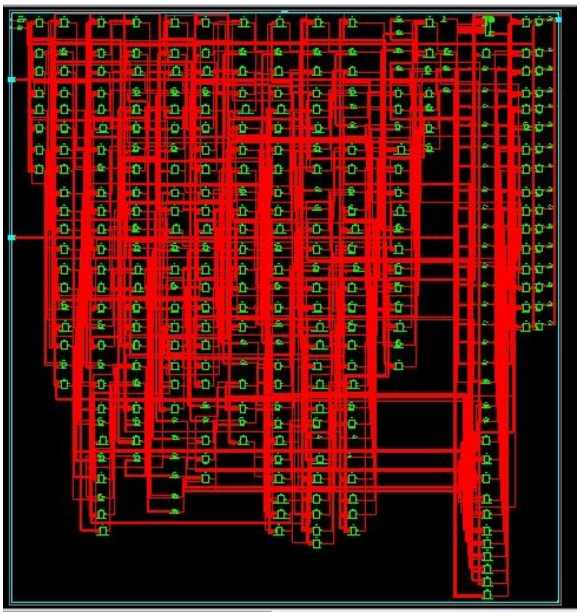


```
Name          Value
clk           1
rst           0
x[7:0]        11111011
y[7:0]        00000101
z[15:0]       0000010011100
a[7:0]        00000010
b[7:0]        00000011
temp_z1[15:0] 0000010011100
temp_z2[15:0] 0000000011100
s[14:0]       0000100111001
c[14:0]       0000000000000
y1[2:0]       001
y2[2:0]       001
y3[2:0]       000
y4[2:0]       000
y_SD[1:0]     00
pp1[14:0]     0000000111110
pp2[14:0]     0000011111011
pp3[14:0]     0000000000000
pp4[14:0]     0000000000000
Z[15:0]       1111111111100
c0            0
out[15:0]     1111111111100
```

RTL schematic:



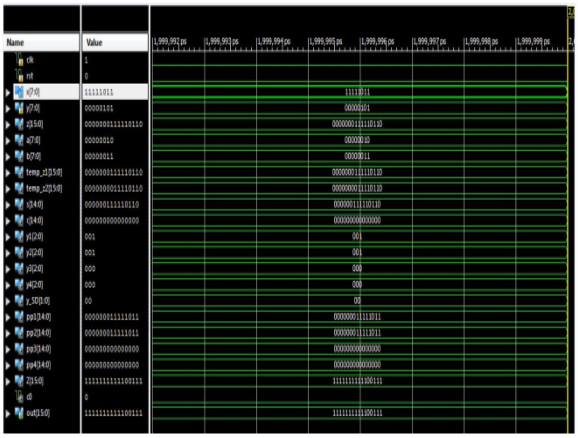
Technology Schematic:



Design Summary.

Proposed results:

Simulation output:

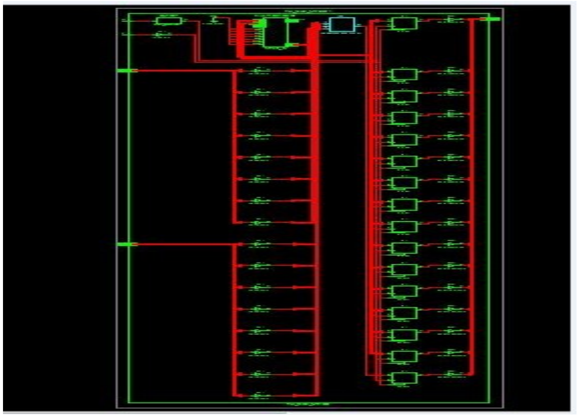


```
Name            Value
clk             1
rst             0
x[7:0]          11111011
y[7:0]          00000101
z[15:0]         0000000111110110
a[7:0]          00000010
b[7:0]          00000011
temp_z1[15:0]   0000000111110110
temp_z2[15:0]   0000000011110110
s[14:0]         000000111110110
c[14:0]         000000000000000
y1[2:0]         001
y2[2:0]         001
y3[2:0]         000
y4[2:0]         000
y_5D[1:0]       00
pp1[14:0]       000000011111011
pp2[14:0]       000000011111011
pp3[14:0]       000000000000000
pp4[14:0]       000000000000000
Z[15:0]         1111111111100111
c0              0
out[15:0]       1111111111100111
```

Synthesis Results:

RTL schematic:

Technology Schematic:



Design Summary:

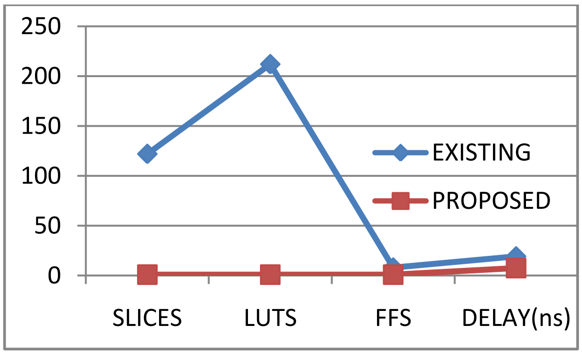
Device Utilization Summary(estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	1	4656	0%
Number of 4input LUTS	1	9312	0%
Number of bonded IOBs	34	232	14%
Number of MULT18X18SIOs	1	20	5%
Number of GCLKs	1	24	4%

Comparison Table.

	AREA			DELAY(ns)
	SLICES	LUTS	FFS	
EXISTING	122	212	8	19.22ns
PROPOSED	1	1	1	7.609ns

Comparison Graph.

Device Utilization Summary(estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	122	4656	2%
Number of Slice Flip Flops	8	9312	0%
Number of 4input LUTS	212	9312	2%
Number of bonded IOBs	34	232	14%
Number of MULT18X18SIOs	1	20	5%
Number of GCLKs	1	24	4%



VI. CONCLUSION

Novel design of pre-encoded multipliers are investigated by off-line encoding the standard coefficients and putting away them in system memory. We propose encoding these coefficients in the Non-Redundant radix-8 Signed-Digit (NR8SD) technique. The proposed pre-encoded NR8SD multiplier are more area and time efficient contrasted with the NR4SD multiplier, customary and pre-encoded MB multipliers. Extensive experimental analysis verifies the gains of the proposed pre-encoded NR8SD multipliers in terms of area complexity and power consumption compared to the NR4SD multiplier and NR8SD multipliers achieved much high speed response compared to NR4SD multipliers.

REFERENCES:

[1]B. Paul, S. F. (Nov. 2009). *Rom-based logic (RBL) design: A lowpower* (Vols. vol. 44, no. 11).

[2]Bernstein, R. (July 1986.). Multiplication by Integer Constants,"Software—Practice and Experience,. *vol. 16, no. 7,* , pp. 641-652.

[3]Booth, A. (1951). "A Signed Binary Multiplication Technique,"Quarterly J. Mechanical Applications of Math.,. *vol. IV, no. 2*, pp. 236-240.

[4]D.J. Magenheimer, L. P. (Aug. 1988). IntegerMultiplication and Division on the HP Precision Architecture. *IEEE Trans. Computers , vol. 37*, no. 8, pp. 980-990.

[5]Huang, Z. (2003). *"High-level optimization techniques for low-power multiplier.* Los Angeles, CA, USA,: Dept. Comput. Sci., Univ. California.

[6]J. Park, K. M. (, Apr. 2003.). High-performance fir filter design. *IEEE Trans. Very Large Scale Integr. Syst , vol. 11, no. 2*, pp. 244–253.

[7]Kolluru, M. (US6108633). *Audio decoder core constants rom optimization.* Patent US6108633.

[8]M.D.Ercegovac and T. Lang, D. A. (2003).

[9]Parhi, K. K. (2007.). *VLSI Digital Signal Processing Systems: Design and Implementation,.* Hoboken, NJ, USA: Wiley.

Reitwiesner, G. W. (1960.). *Binary arithmetic* (Vols. vol. 1,).

[10]Y.-E. Kim, K.-J. C.-G. (2010). CSD-based programmable multiplier design for predetermined coefficient groups. *IEICE Trans.Fundam. Electron. Commun. Comput. Sci., , vol. 93,*, pp. 324–326,.

BIOGRAPHIES:

Saladi Balaji Gupta is a PG Scholar, in VLSI and Embedded Systems, from Godavari Institute of Engineering And Technology, Rajahmundry, Andhra Pradesh

Palivela Soundarya Mala is an Associate Professor in Department of ECE, in Godavari Institute of Engineering And Technology, Rajahmundry, Andhra Pradesh