IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 6, Issue 6, Ver. I (Nov. - Dec. 2016), PP 10-17
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

A New Architecture Design Implementation of Non- Redundant Radix-4 Signed Multiplier Using HDL

D.Hinduja[1], N.Srikanth[2], Dr.B.Subrahmaneswara Rao[3], J.E.N.Abhilash[4]
Department of ECE, Swarnandhra College of Engineering and Technology, Narsapur, A.P, India

Abstract: *This paper briefly presents architecture of pre-encoded multipliers for Digital Signal Processing applications based on off-line encoding of coefficients. Complex arithmetic operations are widely used in Digital Signal Processing (DSP) applications. To this extend, the Non-Redundant radix-4 Signed-Digit (NR4SD) encoding technique, which uses the digit values (-1;0;+1;+2) or (-2;-1; 0;+1) is proposed leading to a multiplier design with less complex partial products implementation. To implement some proposed pre-encoded NR4SD multipliers, including the coefficients memory to prove that they are more area and power efficient than the conventional Modified Booth scheme. By this proposed design the performance increases upto25% by decreasing 30%area and power consumption. By this critical path delay also decreases with decrease in area and power consumption.*

Index Terms: *Multiplying circuits, Modified Booth encoding, Pre-Encoded multipliers, VLSI implementation*

I. Introduction

Many electronic circuits are extensively used in DSP applications to provide better results in communication and multimedia etc..Most of the critical DSP applications uses large number of arithmetic & logical operations depends on applicable area and systems. Such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Finite Impulse Response (FIR) filters and signals' convolution. In all these applications multiplier is basic component multiplication. Multiplication is nothing but product of two numbers. For large number of multiplication we are designing multipliers based on modified booth algorithm to share data we use arithmetic operations for combining of information/binary data for better improvement in performance of communication systems. In the digital designing multiplication done by using addition of partial products. Several architectures have been proposed to optimize the performance of the MAC operation in terms of area occupation, critical path delay or power consumption

Targeting an optimized design of AM operators, fusion techniques are employed based on the direct recoding of the sum of two numbers (equivalently a number in carry-save representation in its Modified Booth (MB) form. Modified Booth (MB) encoding tackles the aforementioned limitations and reduces to half the number of partial products resulting to reduced area, critical delay and power consumption. However, a dedicated encoding circuit is required and the partial products generation is more complex. In audio and video codecs, fixed coefficients stored in memory, are used as multiplication inputs...Constant coefficients are known in advance, we encode the coefficients off-line based on the MB encoding and store the MB encoded coefficients (i.e., 3 bits per digit) into a ROM. Using this technique the encoding circuit of the MB multiplier is omitted. We refer to this design as pre-encoded MB multiplier. Then, we explore a Non-Redundant radix-4 Signed-Digit (NR4SD) encoding scheme extending the serial encoding techniques of multiplier.

The proposed NR4SD encoding scheme uses one of the following sets of digit values: {1; 0; +1; +2} or f{2; 1; 0; +1}. In order to cover the dynamic range of the 2's complement form, all digits of the proposed representation are en-coded according to NR4SD except the most significant one that is MB encoded. Using the proposed encoding formula, we pre-encode the standard coefficients and store them into a ROM in a condensed form (i.e., 2 bits per digit). Compared to the pre-encoded MB multiplier in which the encoded coefficients need 3 bits per digit, the proposed NR4SD scheme reduces the memory size. Also, compared to the MB form, which uses five digit values {2; 1; 0; +1; +2}, the proposed NR4SD encoding uses four digit values. Thus, the NR4SD-based pre-encoded multipliers include a less complex partial products generation circuit..

II. Motivation And Fused Am Implementation

2.1. Motivation

In this paper we focus to reduce the partial products of modified booth(MB) from 3 bits per digit to 2bit per digits. An optimized design of the AM operator is based on the fusion of the adder and the MB encoding unit into a single data path block (Fig. 1(b)) by direct recoding of the sum $Y = A + B$to its MB representation. The drawback of using an adder is that it inserts a significant delay in the critical path of the AM.

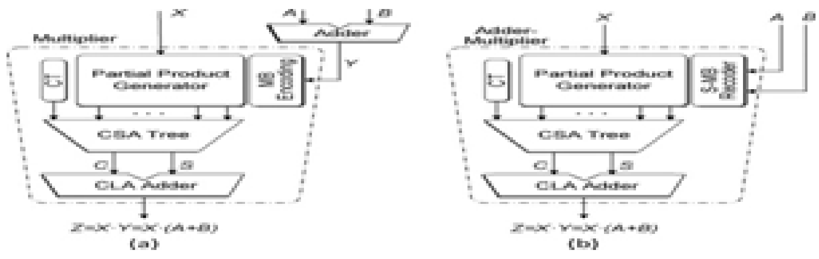


Fig. 1 AM operator based on the (a) conventional design and (b) fused design with direct recoding of the sum of A and B in its MB representation. The multiplier is a basic parallel multiplier based on the MB algorithm. The terms CT, CSA Tree and CLA Adder are referred to the Correction Term, the Carry-Save Adder Tree and the final Carry-Look-Ahead Adder of the multiplier.

2.2. Review of the Modified Booth Form

Modified Booth (MB) is a prevalent form used in multiplication. It is a redundant signed-digit radix-4 encoding technique. Its main advantage is that it reduces by half the number of partial products in multiplication comparing to any other radix-2 representation. Let us consider the multiplication of 2's complement numbers X and Y with each number consisting of $n=2k\mathrm{bits}$.

Table 1: Modified Booth Encoding Table

BINARY			b_j^{MB}	MB ENCODING		
Y_{2J-1}	Y_{2J}	Y_{2J+1}		sign=s_j	x1=One$_j$	x2=two$_j$
0	0	0	0	0	0	0
0	0	1	+1	0	1	0
0	1	0	+1	0	1	0
0	1	1	+2	0	0	1
1	0	0	-2	1	0	1
1	0	1	-1	1	1	0
1	1	0	-1	1	1	0
1	1	1	0	1	0	0

The multiplicand Y can be represented in MB form as

$\mathbf{Y}=<\mathbf{y_{n-1}},y_{n-2}......y_1y_0>_{2's}=-y_{2k-1}.2^{2k-1}+\sum_{i=0}^{n2k-2}$ yi 2j$=<y_{k-1}^{mb}y_{k-2}^{mb}......y_1^{mb}\,y_0^{mb}\ =\sum_{J=0}^{K-1}Y_J^{mb}\,.\,2^{2j}$ ....(1)

$y_j^{mb}{}_{=-}2_{y2j+1}+y_{2j}+y_{2j-1}$.................... (2)

Digits $y_j^{mb}\in\{-2,-1,0,+1,+2\},0\leq j\leq k-1$, correspond to three consecutive bits y_{2j+1},y_{2j} and y_{2j-1} with one bit overlapped and considering that $y_{-1}=0$.table 1 shows how they are formed by summarizing the MB encoding technique each digit is represented by three bits named s,one and two. The sign bit shows if the digit is negative(s=1) or positive(s=0). Using these three digits we calculate MB digits y_j^{mb} By following relation:

$y_j^{mb}=(-1)^{sj}.[one_j+2.two_j]$...................(3)

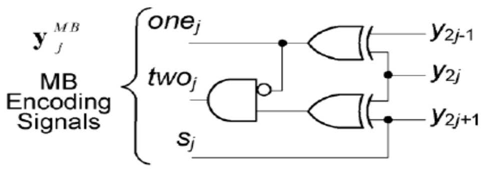
2.3. FAM Implementation

FAM design presented the multiplier is a parallel one based on the MB algorithm for the product XY. The term X are encoded based on MB algorithm and multiplied with the other term Y with their respective digits from 0 to n-1.both the terms have n=2k bits and if signed taken in the form 2's complement form.

$Pp_j=X.y_j^{mb}=\bar{p}_{J,n}2^n+\sum_{i=0}^{n-1}p_{i,j}2^i$......................(4)

The generation of the i-th bit p_{ji} of the partial product PP_i is based on the next logical expression

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



(a) . (b)

Fig. 2.(a) Boolean equations and (b) gate-level schematic for the implementation of the MB encoding signals**.**

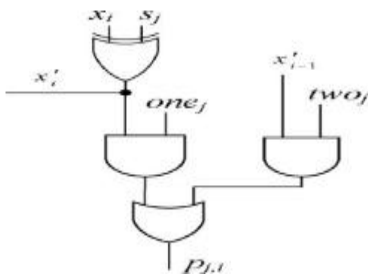


Fig. 3. Generation of the i-th bit p_i of the partial product PP for the conventional MB multiplier.

The least and the most significant bits of the partial product we consider $x_{-1}=0$ and $x=x_{-1}$ respectively. Note that in case that $n=2k+1$, the number of the resulting partial products is $\lfloor n/2 \rfloor+1=k+1$ and the most significant MB digit is formed based on sign extension of the initial 2's complement number.

$$p_{j,i} = \left(\left(x_i \bullet s_j\right) \wedge \bullet ne_j\right) \vee \left(\left(x_{i\ 1} \bullet s_j\right) \wedge tw\bullet_j\right). \qquad .... (5)$$

Then partial products are generated and also added, properly weighted through a Wallace Carry-Save Adder (CSA) tree along with the Correction Term (CT)

$$Z = X \cdot Y = CT + \sum_{j=\bullet} \boldsymbol{PP}_j \cdot 2^{2j} \qquad .... (6)$$

$$CT = CT(l\bullet w) + CT(hi\bullet h) =$$
$$= \sum_{j=\bullet}^{k-1} c_{\text{in},j} \cdot 2^2 + 2^n \quad \left(1 + \sum_{j=\bullet}^{k-1} 2^{2j+1}\right)$$

III. Sum To Modified Booth Recoding Technique

In S-MB recoding technique ***(S-MB)*** sum of two consecutive bits into one MB digit y_j^{mb} .three bits are included in forming MB digit. The most significant of them is negatively weighted and least significant is positive weight we need to use signed bit arithmetic or the transformation of two a for mentioned pairs of bits in MB form.

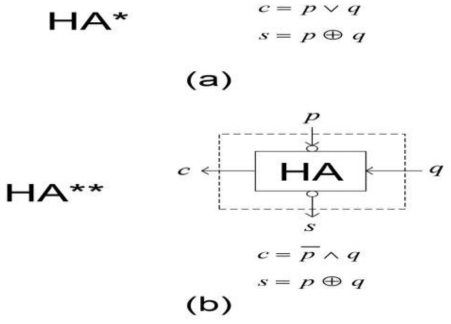


Fig. 4. Boolean equations and schematics signed for signed (a) HA* and (b) HA**.

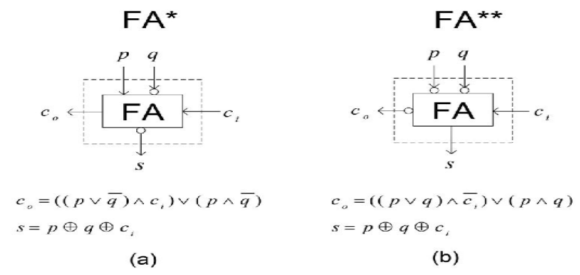

FA*
p q
c_o
FA
c_i
s
$c_o = ((p \vee \overline{q}) \wedge c_i) \vee (p \wedge \overline{q})$
$s = p \oplus q \oplus c_i$
(a)
FA**
p q
c_o
FA
c_i
s
$c_o = ((p \vee q) \wedge \overline{c_i}) \vee (p \wedge q)$
$s = p \oplus q \oplus c_i$
(b)

Fig. 5. Boolean equations and schematics for (a) FA* and (b) FA**.

Table 2
Ha* Basic Operation

Inputs		*Output*	*Outputs*	
p (+)	q (+)	*Value*[1]	c (+)	s (-)
0	0	0	0	0
0	1	+1	1	1
1	0	+1	1	1
1	1	+2	1	0

[1] $OutputValue = 2 \cdot c - s = p + q$

Tables 3
Ha* Basic Operation

Inputs		*Output*	*Outputs*	
p (-)	q (+)	*Value*[3]	c (+)	s (-)
0	0	0	0	0
0	1	+1	1	1
1	0	-1	0	1
1	1	0	0	0

[3] $OutputValue = 2 \cdot c - s = -p + q$

For this purpose, we develop a set of bit-level signed Half Adders (HA) and Full Adders (FA) considering their inputs and outputs to be signed. We use two types of signed has which are referred as HA* and HA**. Tables2 - 4 are their truth tables and in Fig. 4we present their corresponding Boolean equations.

Table 4
Fa* Basic Operation

Inputs		Output	Outputs	
$p\ (+)$	$q\ (+)$	Value[1]	$c\ (+)$	$s\ (-)$
0	0	0	0	0
0	1	+1	1	1
1	0	+1	1	1
1	1	+2	1	0

[1] $OutputValue = 2 \cdot c - s = p + q$

Table 5
Fa Basic Operation**

Inputs		Output	Outputs	
$p\ (-)$	$q\ (+)$	Value[3]	$c\ (+)$	$s\ (-)$
0	0	0	0	0
0	1	+1	1	1
1	0	-1	0	1
1	1	0	0	0

[3] $OutputValue = 2 \cdot c - s = -p + q$

*We also implement the operation of HA*dual operation, HA**operation, FA* operation, FA** operation* shown in tables respectively.

***3.1. S-MB Recoding Scheme**:* The first scheme of the pro-posed recoding technique is referred as S-MB1 and is illustrated in detail in Fig. 6 for both even (Fig. 6(a)) and odd (Fig. 6(b)) bit-width of input numbers. As can be seen in Fig. 6, the sum of A and B is given by the next relation:

$$\text{where } y_j^{MB} = -2s_{2j+1} + s_{2j} + c_{2j}.$$

Both bits s_{2j+1} and s_{2j} are extracted from the recoding cell of Fig. 6. A conventional FA with inputs a_{2j}, b_{2j} and b_{2j-1} produces the carry $c_{2j+1} = (a_{2j} \wedge b_{2j}) \vee (b_{2j-1} \wedge (a_{2j} \vee b_{2j}$

$$Y = A + B = y_k \cdot 2^{2k} + \sum_{j=0}^{k-1} y_j^{MB} \cdot 2^{2j}$$

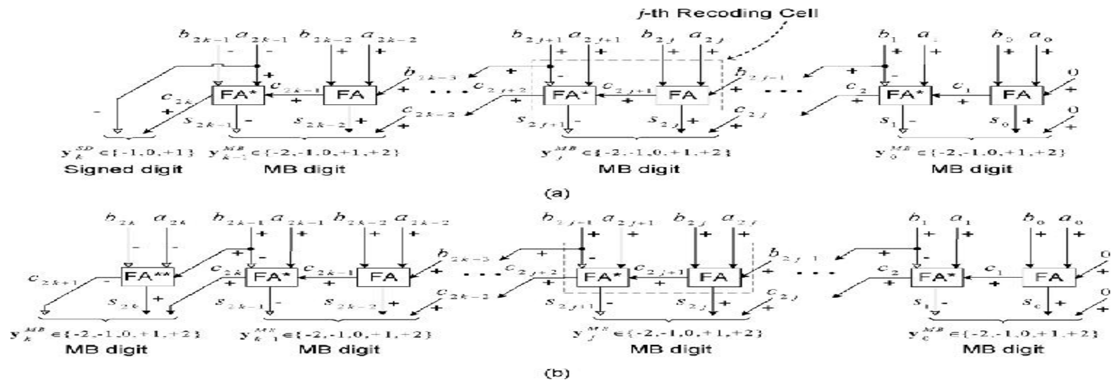
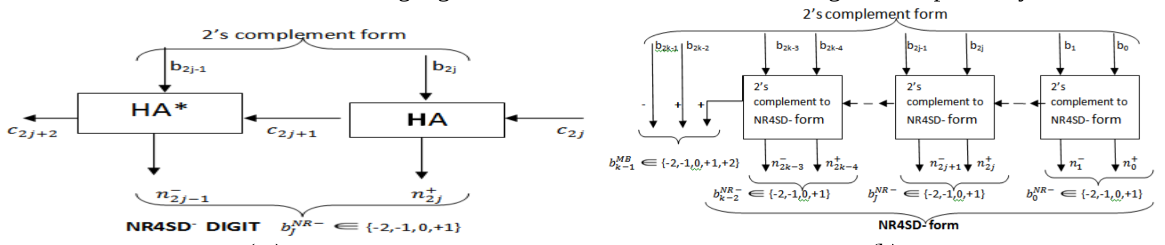

j-th Recoding Cell
FA*
FA
FA**
$y_k^{SD} \in \{-1,0,+1\}$
Signed digit
$y_{k-1}^{MB} \in \{-2,-1,0,+1,+2\}$
MB digit
(a)
(b)

Fig. 6. S-MB recoding scheme for (a) even and (b) odd number of bits

When we form the most significant digit (MSD) of the S-MB1 recoding scheme, we distinguish two cases: In the first case, the bit-width of A and B is even, otherwise A and B comprise of odd numbers.
$T_{S\text{-}MB1} = T_{FA,carry} + T_{FA*,sum}$……………….(9)

IV. Non-Redundant Radix-4 Signed-Digit Algorithm

In this section, we present the Non-Redundant radix-4 Signed-Digit (NR4SD) encoding technique. As in MB form, the number of partial products is reduced to half. When encoding the 2's complement number B, digits $\mathbf{b}^{nr}_j{}^-$ take one of four values:{- 2; -1; 0; +1} or $\mathbf{b}^{nr}_j{}^+$ belongs to { 1; 0; +1; +2} at the NR4SD$^-$ or NR4SD$^+$ algorithm, respectively. Only four different values are used and not five as in MB. As we need to cover the dynamic range of the 2's complement form, the most significant digit is MB encoded (i.e., $\mathbf{b}^{\mathbf{mb}}_{\mathbf{k-1}} \in$ {-2; 1; 0; +1; +2}. The NR4SD$^-$ and NR4SD$^+$ encoding algorithms are illustrated in detail in fig 8&9 respectively.


2's complement form
b_{2j-1}
b_{2j}
HA*
HA
c_{2j+2}
c_{2j+1}
c_{2j}
n^-_{2j-1}
n^+_{2j}
NR4SD$^-$ DIGIT $b_j^{NR-} \in \{-2,-1,0,+1\}$
2's complement to NR4SD- form
$b_{k-1}^{MB} \in \{-2,-1,0,+1,+2\}$
NR4SD- form

(a) *(b)*

Fig.7. Block Diagram of the NR4SD$^-$ Encoding Scheme at the (a) Digit and (b) Word Level.

4.1. NR4SD$^-$ Algorithm

Step 1: Consider the initial values j = 0 and c_0=0.
Step 2: Calculate the carry c_{2j+1} and the sum n^+_{2j} of a Half Adder (HA) with inputs b_{2j} and c_{2j} (Fig. 1a).
$C_{2j+1} = b_{2j} \wedge c_{2j}$; $n^+_{2j} = b_{2j} \oplus c_{2j}$:
Step 3: Calculate the positively signed carry c_{2j+2} (+) and the negatively signed sum n_{2j+1} (-) of a Half Adder* (HA*) with inputs b_{2j+1} (+) and c_{2j+1} (+) (Fig1a). The outputs c_{2j+2} and n_{2j+1} of 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:
$C_{2j+2} = b_{2j+1} \text{-} c_{2j+1}$; $n_{2j+1} = b_{2j+1} \oplus c_{2j+1}$:
Step 4: Calculate the value of the $\mathbf{b}^{nr}{}_j$ digit.

$$b_j^{nr-} = -2n^-_{2j+1} + n^+_{2j}$$ .... (10)

Equation (10) results from the fact that n_{2j+1} is negatively signed and n^+_{2j} is positively signed.
Step 5: j := j + 1.
Step 6: If (j < k 1), go to Step 2. If (j = k 1), encode the most significant digit based on the MB algorithm and considering the three consecutive bits to be b_{2k1}, b_{2k2} and c_{2k2} (Fig. 1b). If (j = k), stop. Table 2 shows how the NR4SD digits are formed. Equations (6) show how the NR4SD encoding signals one^+_j, one^-_j and two^+_j of Table 2 are generated.

Table 6
Nr4sd- Encoding

2's compliment			NR4SD$^-$ form			b_j^{nr-}	NR4SD$^-$ ENCODING		
b_{2J+1}	b_{2J}	c_{2J}	c_{2J+2}	n^-_{2j+1}	n^+_{2j}		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

$$\text{one}_j^+ = \overline{n_{2j+1}} \wedge n2^+_j;$$
$$\text{one}_j = n_{2j+1} \wedge \underline{n2^+_j};$$
$$\text{two}_j = n_{2j+1} \wedge n2^+_j;$$ .... (12)

The minimum and maximum limits of the dynamic range in the NR4SD form are $-2^{n-1}\text{-}2^{n-3}\text{-}2^{n-5}$.........$2 < 2^{n-1}$ and $2^{n+1} + 2^{n+4} + 2^{n+6} + 2 + 1 > 2^{n-1} < 1$. We observe that the NR4SD form has larger dynamic range than the 2's complement form.

4.2. NR4SD$^+$ Algorithm

Step 1: Consider the initial values j = 0 and c_0=0. Step 2: Calculate the carry positively signed c_{2j+1} (+) and the negatively signed sum n_{2j} (-) of a HA* with inputs b_{2j} (+) and c_{2j} (+) (Fig. 2a). The carry c_{2j+1} and the sum n_{2j} of the HA* relate to its inputs as follows:
$2c_{2j+1} - n_{2j} = b_{2j} + c_{2j}$:
The outputs of the HA* are analyzed at gate level in the following equations:
$C_{2j+1} = b_{2j} \text{ -} c_{2j}$; $n_{2j} = b_{2j} \oplus c_{2j}$:
Step 3: Calculate the carry c_{2j+2} and the sum n^+_{2j+1} of a HA with inputs b_{2j+1} and c_{2j+1}.
$C_{2j+2} = b_{2j+1} \wedge c_{2j+1}$; $n^+_{2j+1} = b_{2j+1} \oplus c_{2j+1}$:
Step 4: Calculate the value of the $\mathbf{b}^{nr+}{}_j$ digit.

$$b_j^{nr+} = +2n^+_{2j+1} - n^-_{2j}$$ .... (13)

Equation (11) results from the fact that n^+_{2j+1} is positively signed and n_{2j} is negatively signed.
Step 5: j:= j + 1.
Step 6: If (j < k -1), go to Step 2. If (j = k-1), encode the most significant digit according to MB algorithm and considering the three consecutive bits to be b_{2k-1}, b_{2k-2} and c_{2k-2} (Fig. 2b). If (j = k), stop. Table 3 shows how the NR4SD$^+$ digits are formed. Following equations show how the NR4SD$^+$encoding signals one^+_j, one_j and two^+_j of Table 4 are generated.

$$\text{one}_j^+ = \underline{n2^+_{j+1}} \wedge n_{2j};$$
$$\text{one}_j = n2^+_{j+1} \wedge \underline{n_{2j}};$$
$$\text{two}_j^+ = n2^+_{j+1} \wedge n_{2j};$$ .... (14)

The minimum and maximum limits of the dynamic range in the NR4SD$^+$ form are -2^{n-1} -2^{n-4} -$2^{n-6............}$1 < - 2^{n-1} and 2^{n-1}+2^{n-3}+2^{n-5}+ +2 > 2^{n-1}-1. As observed in the NR4SD encoding technique, the NR4SD$^+$ form has larger dynamic range than the 2's complement form. Considering the 8-bit 2's complement number N, table exposes the limit values -2^8 = -128, 2^8 -1 = 127, and two typical values of N, and presents the MB, NR4SD and NR4SD$^+$ digits that result
when applying the corresponding encoding techniques to each value of N we considered

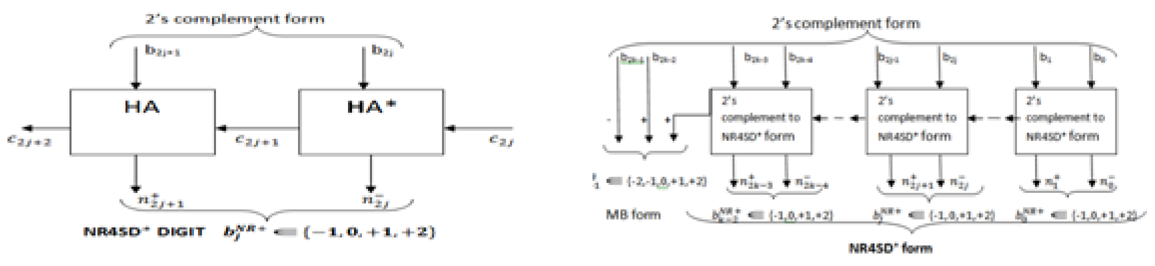

(a) *(b)*
Fig. 8. Block Diagram of the NR4SD$^+$Encoding Scheme at the (a) Digit and (b) Word Level.

Table 7
Nr4sd+ Encoding

2's compliment			NR4SD⁺ form			b_j^{nr+}	NR4SD⁺ ENCO	
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^+_{2j+1}	n^-_{2j}		One$_j^+$	One$_j^-$
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	+1	1	0
0	1	0	0	0	1	+1	1	0
0	1	1	1	1	0	+2	0	0
1	0	0	1	1	0	+2	0	0
1	0	1	1	1	1	-1	0	1
1	1	0	1	1	1	-1	0	1
1	1	1	1	0	0	0	0	0

V. Pre-Encoded Nr4sd Multipliers Design

The system architecture for the pre-encoded NR4SD multipliers is Two bits are now stored in ROM: n_{2j+1}, n^+_{2j} .for the NR4SD-or n^+_{2j+1}, n_{2j} for the NR4SD$^+$ form.

Table 8
Numerical Examples of the Encoding Techniques

2's Complement	10000000	10011010	01011001	01111111
Integer	-128	-102	+89	+127
Modified Booth	2 0 0 0	2 2 1 2	1 2 2 1	2 0 0 1
NR4SD	2 0 0 0	1 2 1 2	2 2 2 1	2 0 0 1
NR4SD	2 0 0 0	2 1 2 2	1 1 2 1	2 0 0 1

Thus, the amount of stored bits is equal to that of the conventional MB design, except for the most significant digit that needs an extra bit as it is MB encoded.

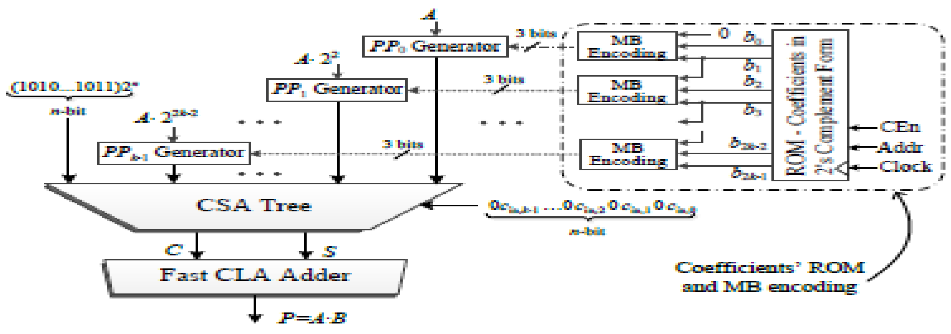


Fig. 9. System Architecture of the Conventional MB multiplier

Compared to the pre-encoded MB the pre-encoded NR4SD multipliers need extra hardware to generate the signals for the NR4SD+ and NR4SD$^+$ form, respectively.

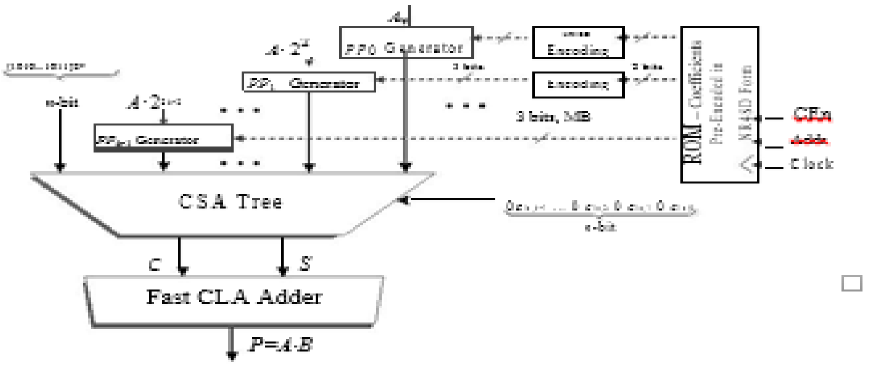
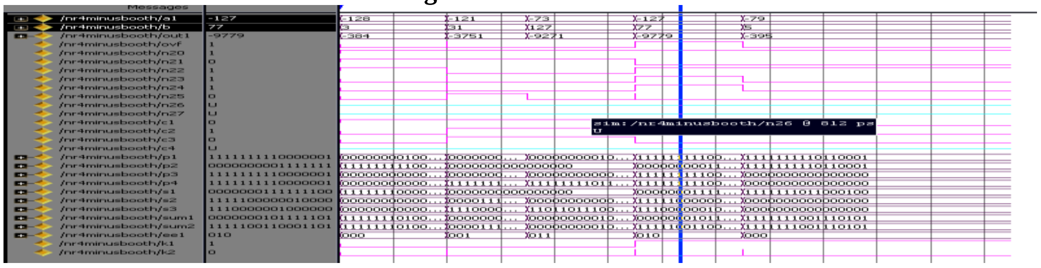


Fig.10.System Architecture of the NR4SD Multipliers

VI. Implementation Results

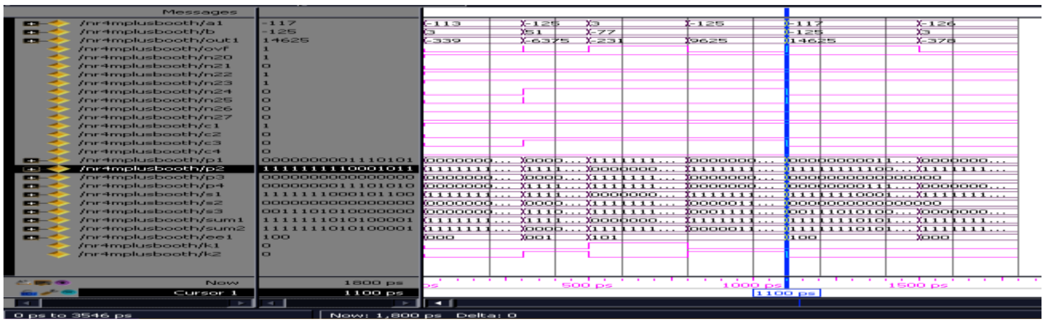
1. Simulation Results For NR4SD$^-$ Encoding



```
Messages
/nr4minusbooth/a1      -127
/nr4minusbooth/b       77
/nr4minusbooth/out1    -9779
/nr4minusbooth/ovf     1
/nr4minusbooth/n20     1
/nr4minusbooth/n21     0
/nr4minusbooth/n22     1
/nr4minusbooth/n23     1
/nr4minusbooth/n24     1
/nr4minusbooth/n25     0
/nr4minusbooth/n26     U
/nr4minusbooth/n27     U
/nr4minusbooth/c1      0
/nr4minusbooth/c2      1
/nr4minusbooth/c3      0
/nr4minusbooth/c4      U
/nr4minusbooth/p1      1111111110000001
/nr4minusbooth/p2      0000000001111111
/nr4minusbooth/p3      1111111110000001
/nr4minusbooth/p4      1111111110000001
/nr4minusbooth/s1      0000000111111100
/nr4minusbooth/s2      1111100000010000
/nr4minusbooth/s3      1110000000100000
/nr4minusbooth/sum1    0000000101111101
/nr4minusbooth/sum2    1111100110001101
/nr4minusbooth/ee1     010
/nr4minusbooth/k1      1
/nr4minusbooth/k2      0
```

From the above simulation result of proposed nr4sd multiplier we verified nr4sd- block by giving inputs x=-3 and y=2 and we got the output of -6.in this we are using partial products -2,-1,0,1 internally

2. Simulation Results For NR4SD$^+$ Encoding



```
Messages
/nr4mplusbooth/a1      -117
/nr4mplusbooth/b       -125
/nr4mplusbooth/out1    14625
/nr4mplusbooth/ovf     1
/nr4mplusbooth/n20     1
/nr4mplusbooth/n21     0
/nr4mplusbooth/n22     1
/nr4mplusbooth/n23     1
/nr4mplusbooth/n24     0
/nr4mplusbooth/n25     0
/nr4mplusbooth/n26     0
/nr4mplusbooth/n27     0
/nr4mplusbooth/c1      1
/nr4mplusbooth/c2      0
/nr4mplusbooth/c3      0
/nr4mplusbooth/c4      0
/nr4mplusbooth/p1      00000000001110101
/nr4mplusbooth/p2      11111111110001011
/nr4mplusbooth/p3      00000000000000000
/nr4mplusbooth/p4      00000000011101010
/nr4mplusbooth/s1      11111111000101100
/nr4mplusbooth/s2      00000000000000000
/nr4mplusbooth/s3      00111010100000000
/nr4mplusbooth/sum1    11111111010100001
/nr4mplusbooth/sum2    11111111010100001
/nr4mplusbooth/ee1     100
/nr4mplusbooth/k1      0
/nr4mplusbooth/k2      0
Now        1800 ps
Cursor 1   1100 ps
0 ps to 3546 ps        Now: 1,800 ps   Delta: 0
```

From the above simulation result of proposed nr4sd multiplier we verified nr4sd+ block by giving inputs x=-5 and y=-3 and we got the output of -15.In this we are using partial products -1,0,1,+2 internally.

VII. Performance Comparison

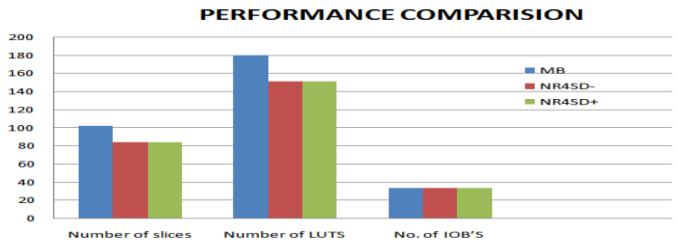
The above architecture has been simulated and synthesized on FPGA XCV3S400K using XILINX ISE- 12.4 tool in HDL code.

Device Utilization Summary (estimated values)

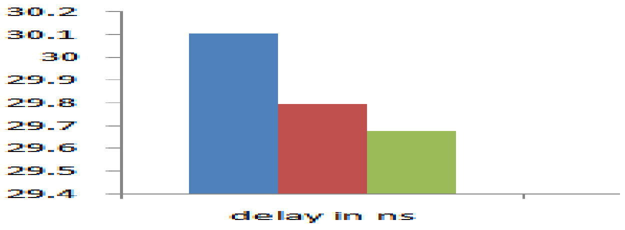
S.NO	Logic Utilization	Available	MB	**NR4SD-**	**NR4SD+**
			Used	Used	Used
1	Number of slices	3584	102	84	84
2	Number of LUTS	7168	180	151	151
3	No. Of IOB'S	97	33	33	33
4	Maximum delay(ns)		30.108	29.790	29.674

VIII. Comparision between Modified Booth And Nr4sd Algorithm

1. Performance



2. Delay



IX. Conclusion

In this paper, new designs of pre-encoded multipliers are explored by off-line encoding the standard coefficients and storing them in system memory. We propose encoding these coefficients in the Non-Redundant radix-4 Signed-Digit (NR4SD) form in encoding. we reduce the partial products of MB algorithm with our proposed design. By this proposed design the performance increases upto25% by decreasing 30%area and power consumption. The proposed NR4SD multiplier design is simulated and verified in vhdl code so this design gives more performance than the modified booth scheme in digital signal processing circuits.

References

[1]. A. Amaricai, M. Vladutiu, and O. Boncalo, "Design issues and imple-mentations for floating-point divide-add fused," IEEE Trans. Circuits Syst. II–Exp. Briefs, vol. 57, no. 4, pp. 295–299, Apr. 2010.

[2]. E. E. Swartzlander and H. H. M. Saleh, "FFT implementation with fused floating-point operations," IEEE Trans. Comput., vol. 61, no. 2, 284–288, Feb. 2012.

[3]. J. J. F. Cavanagh, Digital Computer Arithmetic. New York: McGraw-Hill, 1984.

[4]. S. Nikolaidis, E. Karaolis, and E. D. Kyriakis-Bitzaros, "Estimation of signal transition activity in FIR filters implemented by a MAC archi-tecture," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 19, no. 1, pp. 164–169, Jan. 2000.

[5]. O. Kwon, K. Nowka, and E. E. Swartzlander, "A 16-bit by 16-bit MAC design using fast 5: 3 compressor cells," J. VLSI Signal Process. Syst., vol. 31, no. 2, pp. 77–89, Jun. 2002.

[6]. L.-H. Chen, O. T.-C. Chen, T.-Y. Wang, and Y.-C. Ma, "A multiplica-tion-accumulation computation unit with optimized compressors and minimized switching activities," in Proc. IEEE Int, Symp. Circuits and Syst., Kobe, Japan, 2005, vol. 6, pp. 6118–6121.

[7]. Y.-H. Seo and D.-W. Kim, "A new VLSI architecture of parallel multiplier–accumulator based on Radix-2 modified Booth algorithm,"

[8]. IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 18, no. 2, pp. 201–208, Feb. 2010.

[9]. A. Peymandoust and G. de Micheli, "Using symbolic algebra in algo-rithmic level DSP synthesis," in Proc. Design Automation Conf., Las Vegas, NV, 2001, pp. 277–282.

[10]. W.-C. Yeh and C.-W. Jen, "High-speed and low-power split-radix FFT," IEEE Trans. Signal Process., vol. 51, no. 3, pp. 864–874, Mar. 2003.

[11]. C. N. Lyu and D. W. Matula, "Redundant binary Booth recoding," in Proc. 12th Symp. Comput. Arithmetic, 1995, pp. 50–57.

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

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

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

[15]. O. Macsorley, "High-speed arithmetic in binary computers," Proc. IRE, vol. 49, no. 1, pp. 67–91, Jan. 1961.

[16]. W.-C. Yeh and C.-W. Jen, "High-speed booth encoded parallel multiplier design," IEEE Trans. Comput., vol. 49, no. 7, pp. 692–701, Jul. 2000.

[17]. Z. Huang, "High-level optimization techniques for low-power multiplier design," Ph.D. dissertation, Department of Com-puter Science,

[18]. University of California, Los Angeles, CA, 2003.

[19]. Z. Huang and M. Ercegovac, "High-performance low-power left-to-right array multiplier design," IEEE Trans. Comput., vol. 54, no. 3, pp. 272–283, Mar. 2005.

[20]. Y.-E. Kim, K.-J. Cho, and J.-G. Chung, "Low power small area modified booth multiplier design for predetermined coeffi-cients," IEICE Trans. Fundam. Electron. Commun. Comput. Sci., vol. E90-A, no. 3, pp. 694–697, Mar. 2007.

[21]. C. Wang, W.-S. Gan, C. C. Jong, and J. Luo, "A low-cost 256-point fft processor for portable speech and audio appli-cations," in Int. Symp. on Integrated Circuits (ISIC 2007), Sep. 2007, pp. 81–84.

[22]. A. Jacobson, D. Truong, and B. Baas, "The design of a recon-figurable continuous-flow mixed-radix fft processor," in IEEE Int. Symp. on Circuits and Syst. (ISCAS 2009), May 2009, pp. 1133–1136