DESIGN OF NON-REDUNDENT RADIX-4 BOOTH MULTIPLIER FOR ERROR TOLERANT COMPUTING

Dr.M. DEEPA Ph.D., P.J.V.E.SAI GANESH, K. YESWANTH BABU, K. TEJA KIRAN, P. RAVI KIRAN
ASSOCIATE PROFESSOR, FINAL YEAR B.TECH STUDENTS
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
USHA RAMA COLLEGE OF ENGINEERING AND TECHNOLOGY, VIJAYAWADA, INDIA

Abstract:

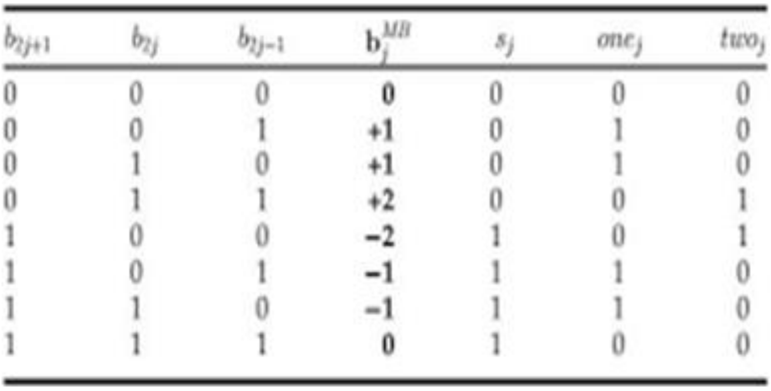
The Modified Booth multiplier is attractive to many multimedia and digital signal processing systems. This paper presents the design of 32 bit Modified Booth multiplier. The multipliers such as Braun array multiplier and Array multiplier are used for unsigned multiplication. This paper focusing on design of Modified Booth Multiplier which performs both signed and unsigned multiplication. Here used Carry Select Adder which increases the speed of multiplier operation. Booth encoder multiplier with Carry select Adder utilizes the minimum hardware, reduced chip area, low power dissipation and reduced the cost of the system.

Index Terms: Modified Booth multiplier, Carry Select Adder, Braun array multiplier

I. Introduction

The[1]modified-Booth algorithm is extensively used for high-speed multiplier circuits. Once, when array multipliers were used, the reduced number of generated partial products significantly improved multiplier performance. In designs based on reduction trees with logarithmic logic depth, however, the reduced number of partial products has a limited impact on overall performance. The Baugh-Wooley algorithm is a different scheme for signed multiplication, but is not so widely adopted because it may be complicated to deploy on irregular reduction trees. Again the Baugh-Wooley algorithm is for only signed number multiplication. The array multipliers and Braun array multipliers operates only on the unsigned numbers. Thus, the requirement of the modern computer system is a dedicated and very high speed multiplier unit that can perform multiplication operation on signed as well as unsigned numbers. In this paper we designed and implemented a dedicated multiplier unit that can perform multiplication operation on both signed and unsigned numbers, and this multiplier is called as AMBE multiplier.

table1: modified booth encoding.



b_{2j+1}	b_{2j}	b_{2j-1}	b_j^{MB}	s_j	one_j	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

II.MODIFIED BOOTH ALGORITHM

Modified Booth (MB) is a redundant radix-4 encoding technique [4], [5]. Considering the multiplication of the 2's complement numbers A, B, each one consisting of n=2k bits, B can be represented in MB form as:

$$\begin{aligned} B &= \langle b_{n-1} \ldots b_0 \rangle_{2's} = -b_{2k-1}2^{2k-1} + \sum_{i=0}^{2k-2} b_i 2^i \\ &= \langle \mathbf{b}_{k-1}^{MB} \ldots \mathbf{b}_0^{MB} \rangle_{MB} = \sum_{j=0}^{k-1} \mathbf{b}_j^{MB} 2^{2j} \end{aligned} \qquad \ldots\ldots\ldots\ldots(1)$$

Digits $b^{MB}_j \in \{-2,-1,0,+1,+2\}, 0 \leq j \leq k-1$, are formed as follows:

$$b^{MB}_j = -2b_{2j+1} + b_{2j} + b_{2j-1} \quad \ldots\ldots\ldots \quad (2)$$

where $b-1 = 0$. Each MB digit is represented by the bits s, one and two (Table 1). The bit s shows if the digit is negative (s=1) or positive (s=0). One shows if the absolute value of a digit equals 1 (one=1) or not (one=0). Two shows if the absolute value of a digit equals 2 (two=1) or not (two=0). Using these bits, we calculate the MB digits b^{MB}_j as follows:

$$\mathbf{b}_j^{MB} = (-1)^{s_j} \cdot (one_j + 2two_j). \qquad \text{-------------------}(3)$$

Equations (4) form the MB encoding signals.

$$\begin{aligned} s_j = b_{2j+1}, \quad one_j = b_{2j-1} \oplus b_{2j}, \\ two_j = (b_{2j+1} \oplus b_{2j}) \wedge \overline{one}_j. \end{aligned} \qquad \text{----------}(4)$$

III.NON-REDUNDANT RADIX-4 SIGNED DIGIT ALGORITHM

In this section[4] 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 b^{NR-}_j take one of four values: {−2, −1, 0, +1} or $b^{NR+}_j \in \{-1, 0, +1, +2\}$ at the NR4SD⁻ or NR4SD⁺ algorithm, respectively. Only four different values are used and not five as in MB algorithm, which leads to $0 \leq j \leq k - 2$. As we need to cover the dynamic range of the 2's complement form, the most significant digit is MB encoded (i.e., $b^{MB}_{k-1} \in \{-2, -1, 0, +1, +2\}$). The NR4SD⁻ and NR4SD⁺ encoding algorithms are illustrated in detail in Fig. 1 and 2, respectively.

NR4SD ALGORITHM:

Step 1: Consider the initial values j = 0 and c0=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 and negatively signed sum of a Half Adder.

Step 4: Calculate the value of b_j^{NR-}.

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 as shown in Fig.1b. If (j = k), stop.

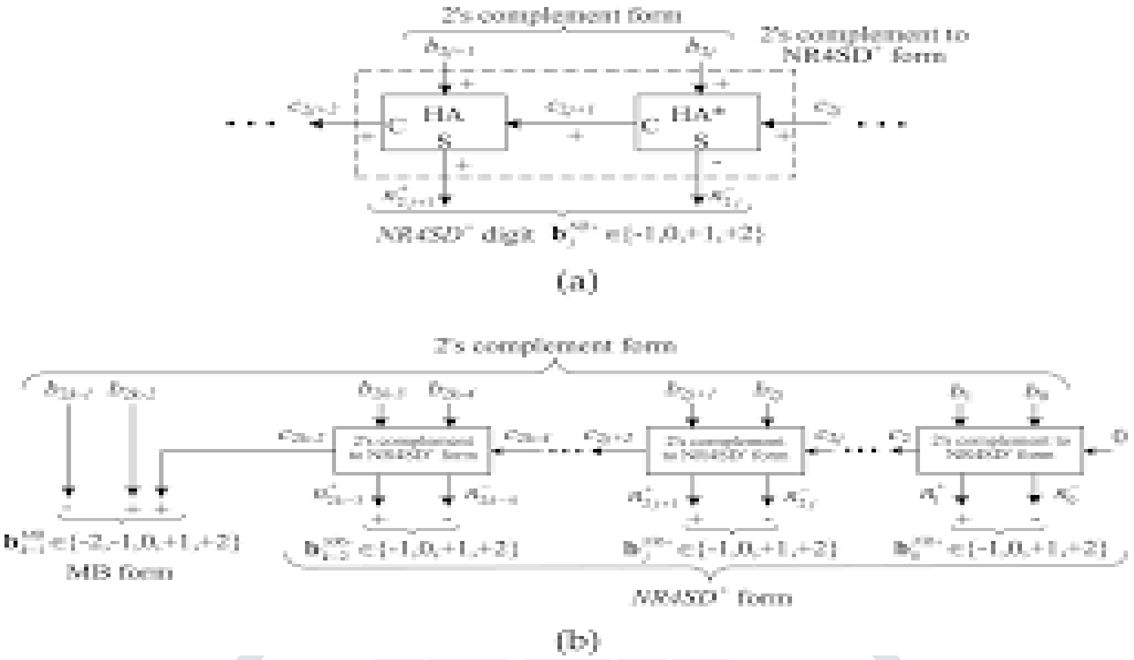
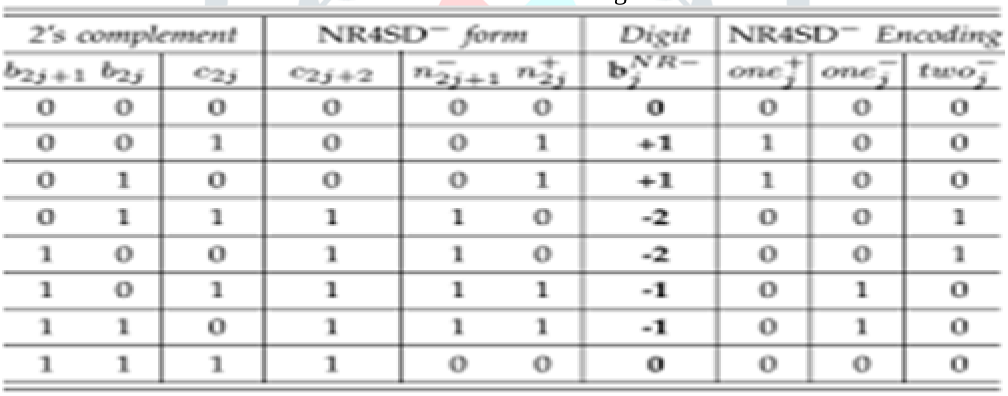
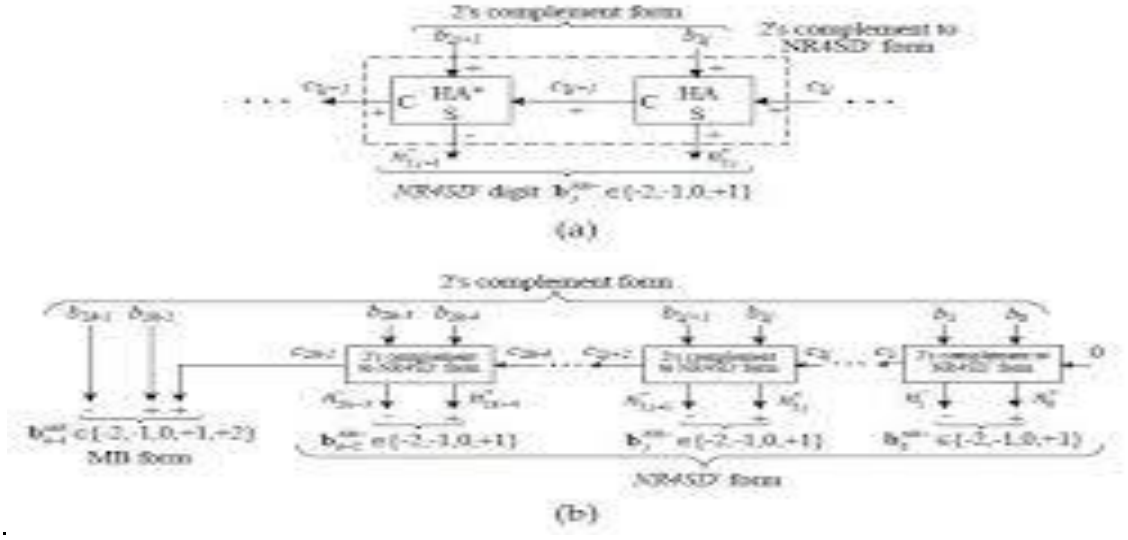

2's complement form
2's complement to NR4SD+ form
b_{2j+1}
b_{2j}
HA
HA*
C
S
c_{2j+2}
c_{2j+1}
c_{2j}
n^{+}_{2j+1}
n^{-}_{2j}
NR4SD+ digit $b_j^{NR+} \in \{-1,0,+1,+2\}$
(a)
2's complement form
2's complement to NR4SD+ form
$b_{k-1}^{MB} \in \{-2,-1,0,+1,+2\}$
MB form
$b_j^{NR+} \in \{-1,0,+1,+2\}$
NR4SD+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

fig. 1. block diagram of the nr4sd− encoding scheme at the (a) digit and (b) word level

table1: nr4sd- encoding



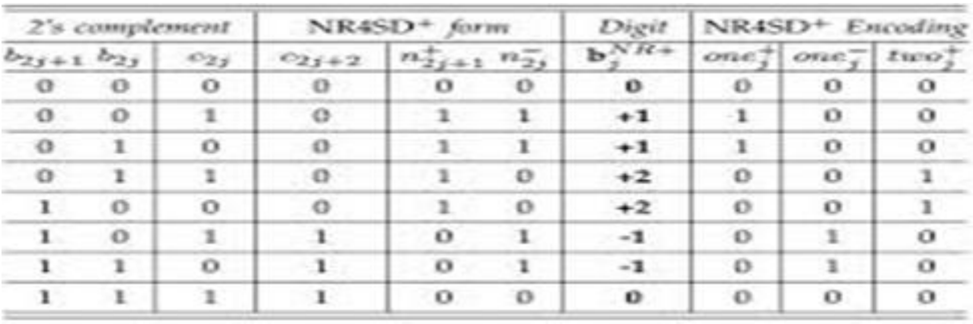
2's complement			NR4SD− form			Digit	NR4SD− Encoding		
b_{2j+1}	b_{2j}	c_{2j}	c_{2j+2}	n^{-}_{2j+1}	n^{+}_{2j}	b_j^{NR-}	one_j^{+}	one_j^{-}	two_j^{-}
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	+1	1	0	0
0	1	0	0	0	1	+1	1	0	0
0	1	1	1	1	0	-2	0	0	1
1	0	0	1	1	0	-2	0	0	1
1	0	1	1	1	1	-1	0	1	0
1	1	0	1	1	1	-1	0	1	0
1	1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



.

fig. 2. block diagram of the nr4sd+ encoding scheme at the (a) digit and (b) word level.

table3: 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

IV.Algorithm of the Modified Booth Multiplier

Multiplication consists of three steps:

1) The first step to generate the partial products.

2) The second step to add the generated partial products until the last two rows are remained.

3) The third step to compute the final multiplication results by adding the last two rows. The modified Booth algorithm reduces the number of partial products by half in the first step. We used the Modified Booth Encoding (MBE) scheme proposed in [2]. It is known as the most efficient Booth encoding and decoding scheme. To multiply X by Y using the modified Booth algorithm starts from grouping Y by three bits and encoding into one of {-2, -1, 0,1,2}. Table1 shows the rules to generate the encoded signals by MBE scheme and Fig.1(a) shows the corresponding logic diagram. The Booth decoder generates the partial products using the encoded signals.

V.MODIFIED BOOTH MULTIPLIER DESIGN

In this section,[4]we explore the implementation of pre-encoded multipliers. One of the two inputs of these multipliers is pre-encoded either in MB or in NR4SD /NR4SDþ representation. We consider that this input comes from a set of fixed coefficients.The coefficients are encoded off-line based on MB or NR4SD algorithms and the resulting bits of encoding are stored in a ROM. Since our purpose is to estimate the efficiency of the proposed multipliers, we first present a review of the conventional MB multiplier in order to compare it with the pre-encoded schemes.

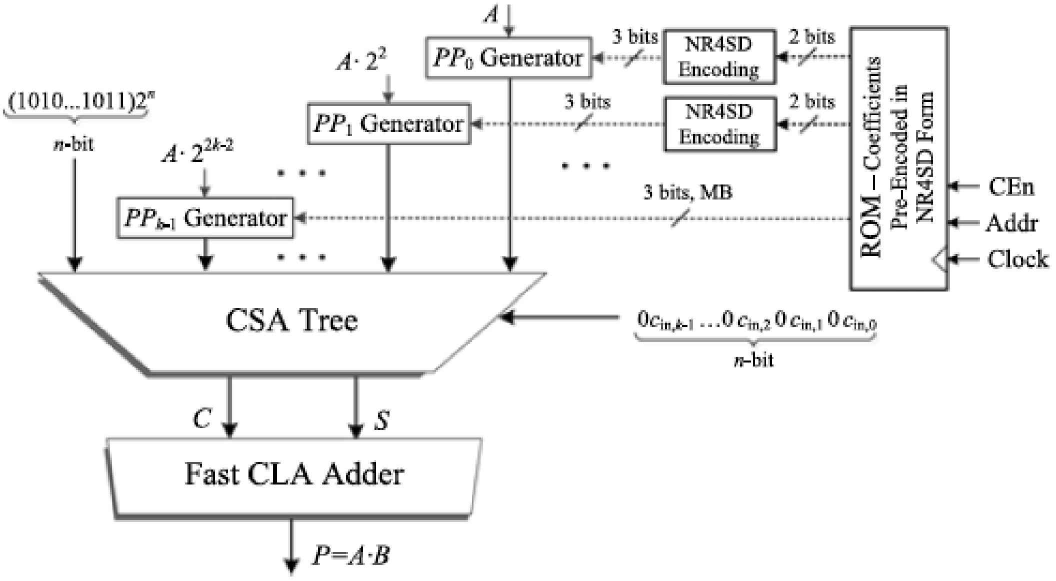
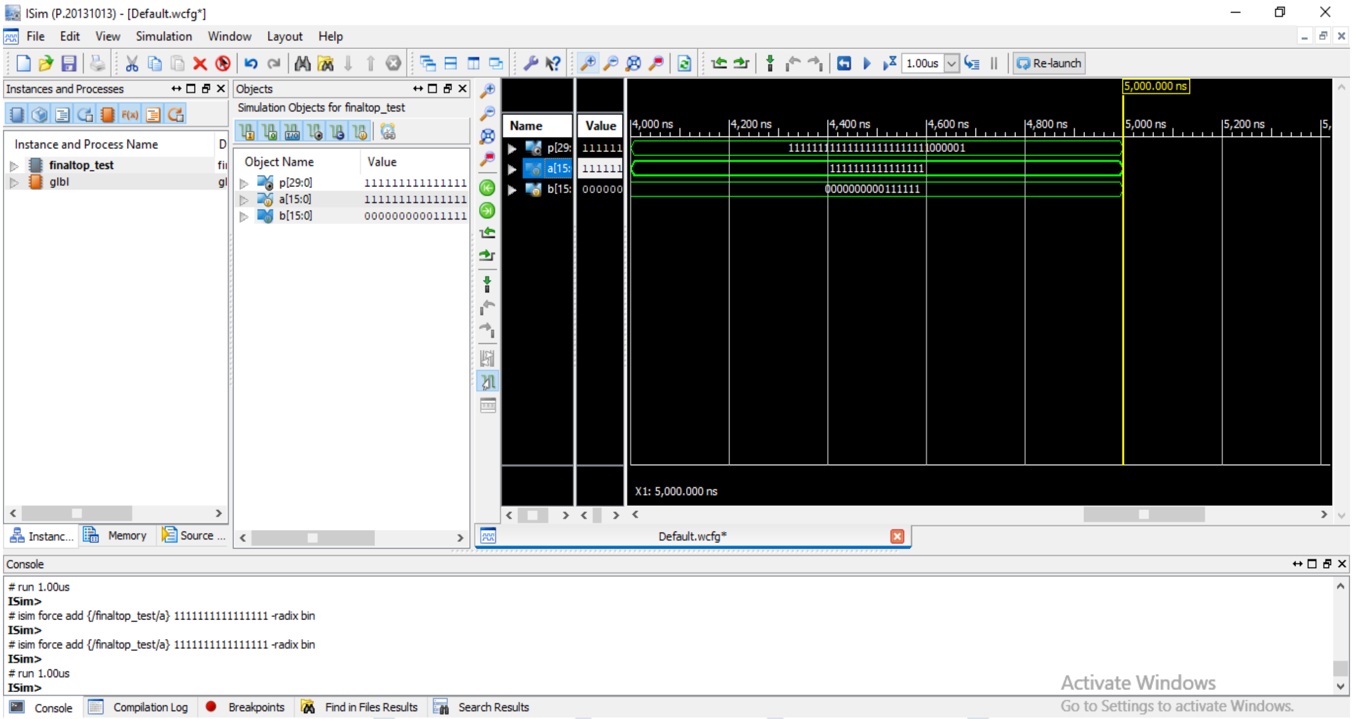


fig.3.system architecture for nr4sd multiplier.

VI.RESULTS



Instance and Process Name	D
finaltop_test	fi
glbl	gl

Object Name	Value
p[29:0]	111111111111111
a[15:0]	111111111111111
b[15:0]	000000000011111

Name	Value	
p[29:	111111	111111111111111111111111000001
a[15:	111111	1111111111111111
b[15:	000000	0000000000111111

```
# run 1.00us
ISim>
# isim force add {/finaltop_test/a} 1111111111111111 -radix bin
ISim>
# isim force add {/finaltop_test/a} 1111111111111111 -radix bin
ISim>
# run 1.00us
ISim>
```

fig.4.output of radix-4 booth multiplier.

VII. Future Scope

This method can also be implemented for 32 bit, 64 bit and 124 bit. Multipliers are used to perform arithmetic operations in advanced microprocessors and microcontrollers with more speed and less power consumption using Modified Booth scheme.

VIII.CONCLUSION

A new design of multipliers is explored by off-line encoding the standard coefficients and storing them in system memory. The proposed multiplier designs are more area and power efficient compared to the conventional MB designs. Synthesis and Simulation analysis in Xilinx 14.5 software using Verilog Hardware Description Language verifies the gains of the proposed multipliers in terms of area complexity and power consumptions compared to the conventional MB multiplier.

REFERENCES

1. J. Liang, J. Han, and F. Lombardi, “New metrics for the reliability of approximate and probabilistic adders, “IEEE Trans. Computers, vol. 63, pp. 1760-1771, Sep.

2. A. Momeni, J. Han, P. Montuschi, and F. Lombardi, “Design and analysis of approximate compressors for multiplication,” IEEE Trans. Computers, vol. 64, pp. 984–994, 2015.

3. H. Jiang, F. Qiao, F. Lombardi, “Approximate radix-8 Booth multipliers for low-power and high-performance operation”, IEEE Trans. Computers, vol. 65, pp. 2638 – 2644, Aug. 2016.

4. Z. Huang, “High-level optimization techniques for low-power multiplier design,” Ph.D. dissertation, Department of Computer Science, University of California, Los Angeles, CA, 2003.