Performance Improved Multipliers Based On Non Redundant Radix-4 Signed Digit Encoding

S.Sri Divya¹ K.s.n raju²
¹P.G Student VLSI Design (ECE), SVECW
²Assistant Professor, Dept of ECE, SVECW

Abstract: In this paper, we introduce architectures of pre-encoded multipliers based on non redundant radix-4 signed digit encoding technique. In this architecture of nr4sd and modified booth encoding is used. Using Modified booth is a redundant radix-4 encoding technique. To this extend radix-8 modified booth encoding which reduce the number of partial product implementation. Extensive experimental analysis verifies that the radix-8 mb encoding occupies less area and delay compared to nr4sd multipliers.

Key words-Multiplying circuits, modified booth encoding, pre-encoded multipliers, VLSI implementation.

INTRODUCTION

Multimedia and digital signal processing (DSP) applications (e.g., fast Fourier transform (FFT), audio/video Codec’s) carry out a large number of multiplications with coefficients that do not change during the execution of the application. Since the multiplier is a basic component for implementing computationally intensive applications, its architecture seriously affects their performance. Constant coefficients can be encoded to contain the least non-zero digits using the canonic signed digit (CSD) representation [1]. CSD multipliers comprise the fewest non-zero partial products, which in turn decreases their switching activity. However, the CSD encoding involves serious limitations. Folding technique [2], which reduces silicon area by time-multiplexing many operations into single functional units, e.g., adders, multipliers, is not feasible as the CSD-based multipliers are hard-wired to specific coefficients. In [3], a CSD-based programmable multiplier design was proposed for groups of pre-determined coefficients that share certain features. The size of ROM used to store the groups of coefficients is significantly reduced as well as the area and power consumption of the circuit.

However, this multiplier design lacks flexibility since the partial products generation unit is designed specifically for a group of coefficients and cannot be reused for another group. Also, this method cannot be easily extended to large groups of pre-determined coefficients attaining at the same time high efficiency Modified Booth (MB) encoding [4], [5], [6], [7] tackles the afore-mentioned 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 [8], Kim et al. proposed a technique similar to [3], for designing efficient MB multipliers for groups of pre-determined coefficients with the same limitations described in the previous paragraph. In [9], [10], multipliers included in butterfly units of FFT processors use standard coefficients stored in ROMs. In audio [11], [12] and video [13], [14] Codec’s, fixed coefficients stored in memory, are used as multiplication inputs. Since the values of 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 [15], [16], [17], 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 [6], NR4SD encoding scheme uses one of the following sets of digit value {-1,0,+1,+2} or {-2,-1,0,+1} In order to cover the dynamic range of the 2’s complement form, all digits of the NR4SD form.

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. We explore the efficiency of the aforementioned pre-encoded multipliers taking into account the size of the coefficients’ ROM.

**II. MODIFIED BOOTH ALGORITHM**

In order to achieve high-speed multiplication, multiplication algorithms using parallel counters, such as the modified Booth algorithm has been proposed, and some multipliers based on the algorithms have been implemented for practical use. This type of multiplier operates much faster than an array multiplier for longer operands because its computation time is proportional to the logarithm of the word length of operands.

Booth multiplication is a technique that allows for smaller, faster multiplication circuits, by recoding the numbers that are multiplied. It is possible to reduce the number of partial products by half, by using the technique of radix-4 Booth recoding. The basic idea is that, instead of shifting and adding for every column of the multiplier term and multiplying by 1 or 0, we only take every second column, and multiply by ±1, ±2, or 0, to obtain the same results. The advantage of this method is the halving of the number of partial products. To Booth recode the multiplier term, we consider the bits in blocks of three, such that each block overlaps the previous block by one bit. Grouping starts from the LSB, and the first block only uses two bits of the multiplier. Figure 3 shows the grouping of bits from the multiplier term for use in modified booth encoding.

![Fig 1. Modified booth encoder](image)

**III. NON-REDUNDANT RADIX-4 SIGNED DIGIT ALGORITHM**

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 bNRj take one of four values: f2; 1; 0; +1g or bNR+j2 f 1; 0; +1; +2g at the NR4SD- or NR4SD+ algorithm, respectively. Only four different values are used and not five as in MB algorithm, which leads to 0 j 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., bMBk 12 f 2; 1; 0; +1; +2g). The NR4SD- and NR4SD+ encoding algorithms are illustrated in detail in Fig. 3 and 4, respectively.
Fig. 3 Block Diagram of the NR4SD Encoding Scheme at the (a) Digit and (b) Word Level.

Table-1
NR4SD- Encoding

<table>
<thead>
<tr>
<th>2’s complement</th>
<th>NR4SD- form</th>
<th>Digit</th>
<th>NR4SD- Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_{2j+1} b_{2j}</td>
<td>c_{2j} c_{2j+2} n_{2j+1} n_{2j}</td>
<td>b_{j}^{NR}</td>
<td>one_{j}</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 0 1 0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>0 0 1 0</td>
<td>+1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 1 0 1</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>1 1 0 0</td>
<td>-2</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 1 1 1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 1 1 1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1 1 0 0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
3.1 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:

\[
c_{2j+1}^+ = b_{2j}^- - c_{2j}^-; \quad n_{2j}^- = b_{2j}^- c_{2j}^-;
\]

Step 3: Calculate the carry \( c_{2j+2} \) and the sum \( n_{2j+1}^+ \) of a HA with inputs \( b_{2j+1}^+ \) and \( c_{2j+1}^- \):

\[
c_{2j+2} = b_{2j+1}^+ \land c_{2j+1}^-; \quad n_{2j+1}^+ = b_{2j+1}^+ c_{2j+1}^-;
\]

Step 4: Calculate the value of the \( b_{NR}^+ \) digit. B NR \( +j = 2n_{2j+1} + 1n_{2j} \): (7)

Equation (7) results from the fact that \( n_{2j+1} + 1n_{2j} \) positively signed and \( n_{2j} \) 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 \( b_{2k} \), \( b_{2k+2} \) and \( c_{2k} \) (Fig. 2b). If (\( j = k \)), stop.

Table 3 shows how the NR4SD+ digits are formed. Equations (8) show how the NR4SD+ encoding signals one \( +j \), one \( j \) and two \( +j \) of Table 4 are generated.

One \( +j = n_{2j} + 1n_{2j} \);

One \( j = n_{2j} + 1n_{2j} \);

Two \( +j = n_{2j} + 1n_{2j} \);

The minimum and maximum limits of the dynamic range in the NR4SD+ form are \( 2n_{12} \leq 2n \) and \( 2n_{1} + 2n_{3} + 2n_{5} + 2 > 2n_{11} \). As observed in the NR4SD encoding technique, the NR4SD+ form has larger dynamic range than the 2’s complement form.

That result when applying the corresponding encoding techniques to each value of \( N \) we considered. We added a bar above the negatively signed digits in order to distinguish them from the positively signed ones.
IV. Pre-Encoded NR4SD Multipliers Design

The system architecture for the pre-encoded NR4SD multipliers is presented in Fig. 5. Two bits are now stored in ROM: \( n_{2j+1}, n_{2j+2} \) for the NR4SD, or \( n_{2j+1}, n_{2j+2} \) for the NR4SD+. In this way, we reduce the memory requirement to +1 bits per coefficient while the corresponding memory required for the pre-encoded MB scheme is \( 3n/2 \) bits per coefficient. 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. Compared to the pre-encoded MB multiplier, where the MB encoding blocks are omitted, the pre-encoded NR4SD multipliers need extra hardware to generate the signals of (6) and (8) for the NR4SD and NR4SD+ form, respectively.

Each partial product of the pre-encoded NR4SD- and NR4SD+ multipliers is implemented respectively, except for the \( P_{k1} \) that corresponds to the most significant digit. As this digit is in MB form, we use the PPG for the \( s_j \) bit. The partial products, properly weighted, and the correction term (COR) of (11) are fed into a CSA tree. The input carry \( c_{in} \) of (11) is calculated as \( c_{in} = \text{two}_{j} - \text{one}_{j} \) and \( c_{in} = \text{one}_{j} \) for the NR4SD- and NR4SD+ pre-encoded multipliers, respectively. The carry-save output of the CSA tree is finally summed using a fast CLA adder.

V. PROPOSED METHOD

5.1 Pre-encoded modified booth encoding:

The system architecture for the pre-encoded modified booth multiplier is presented in two bits are now stored in ROM: \( n_{2j+1}, n_{2j+2} \) for the NR4SD, or \( n_{2j+1}, n_{2j+2} \) for the NR4SD+. In this way, we reduce the memory requirements to \( n+1 \) bit per coefficient while the corresponding memory required for the pre-encoded MB scheme is \( 3n/2 \) bits per coefficient.
In this Rom is stored as coefficients pre-encoded in mb form. In nr4sd encoding technique radix-4 is used. In modified booth encoding technique radix value increased to 8.as the radix value increases number of the partial products will be reduced. Each of the pp generator output given to carry save adder. The carry save output of the csa tree is finally summed using a fast cla adder.

The generation of the bit $pi$ of the partial product $PPj$ is illustrated at gate level. For the computation of the least and most significant bits of $PPj$, $a1=0$ and $an= an-1$. After shaping the partial products, they are added, properly weighted, through a Carry Save Adder (CSA) tree along with the correction term. The CS output of the tree is leaded to a fast Carry Look Ahead (CLA) adder to form the final result $P = X \times Y$. In the pre-encoded MB multiplier scheme, the coefficient $B$ is encoded off-line according to the conventional MB form. The resulting encoding signals of $B$ are stored in a ROM. This contains the ROM with coefficients in 2’s complement form and the MB encoding circuit, is now totally replaced by the ROM with coefficients in MB form. The MB encoding blocks of Fig. 1 are omitted. The new ROM is used to store the encoding signals of $B$ and feed them into the partial product generators ($PPj$ Generators PPG) on each clock cycle. Since the n-bit coefficient $B$ needs three bits per digit when encoded in MB form, the ROM width requirement is $3n/2$ bits per coefficient. Thus, the width and the overall size of the ROM are increased by 50% compared to the ROM of the conventional scheme.

**5.2 Radix 8 Booth encoder:**
Radix-8 Booth encoding 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 Radix-8 algorithm reduces the number of partial products to $n/3$, where $n$ is the number of multiplier bits. Thus it allows a time gain in the partial products summation.

**Fig. 6. System architecture of mb multiplier**

**Fig. 7. Grouping of multiplier terms**
Table-2
Radix-8 Modified booth encoding

<table>
<thead>
<tr>
<th>$B_{2j+2}$</th>
<th>$B_{2j+1}$</th>
<th>$B_{2j}$</th>
<th>$B_{2j-1}$</th>
<th>$b_{mb}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>+2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>+3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

VI. SIMULATION RESULT
The architecture of pre-encoded multipliers is designed. The programming language used in this is verilog HDL and simulated using 13.2 and isim simulator. Design properties are Spartan 3E family, XC3S500E device with speed grade -5.

Fig8. Simulation result for nr4sd multiplier
Performance Improved Multipliers Based On...

VII. COMPARISONS

In section VI simulation results are presented. Comparison of NRR4SD multiplier and radix-8 modified booth multiplier is discussed in this section for synthesis results and device properties Spartan3E family, VQ100 package, XC3S500E device with a speed grade of -5 is used.

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
<th>Note(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Latches</td>
<td>8</td>
<td>9,212</td>
<td>1%</td>
<td></td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>209</td>
<td>9,312</td>
<td>2%</td>
<td></td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>118</td>
<td>4,656</td>
<td>2%</td>
<td></td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>118</td>
<td>118</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
<td>118</td>
<td>0%</td>
<td></td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>209</td>
<td>9,312</td>
<td>2%</td>
<td></td>
</tr>
<tr>
<td>Number of bonded 100s</td>
<td>34</td>
<td>66</td>
<td>51%</td>
<td></td>
</tr>
<tr>
<td>IOB Flip Flops</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of BUSGMUXes</td>
<td>1</td>
<td>24</td>
<td>4%</td>
<td></td>
</tr>
<tr>
<td>Average Pinout of Non-Clock Nets</td>
<td>3.75</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig.9. Simulation result for radix-8 modified booth multiplier

Fig.10. Area report for NRR4SD multiplier

Timing Summary:
Minimum period: No path found
Minimum input arrival time before clock: 23.302ns
Maximum output required time after clock: 6.216ns
Maximum combinational path delay: No path found

Fig.11. Delay report for NRR4SD multiplier

Area and delay reports of the NRR4SD multiplier are shown in Fig.10 and Fig.11 from device utilization summary it is noticed that 4656 slices are available in this device, but only 118 slices are used in this design and number of 4 input luts are available 9312 but this design utilizes only 209 4 input luts and delay observed in this method is 23.302ns.
Area and delay reports of the radix-8 modified booth multiplier is shown in fig. 12 and fig. 13 from device utilization summary it is noticed that 4656 slices are available in this device, but only 44 slices are used in this design and number of 4 input luts are available 9312 but this design utilizes only 87 4 input luts and delay observed in this method is 13.756 ns. This radix-8 modified booth multiplier shows better result when compared to nr4sd multiplier in terms of area and delay.

**Device utilization graph for multipliers**

![Area report for radix-8 modified booth multiplier](image1.png)

![Area report for nr4sd multiplier and radix-8 mb multiplier](image2.png)
By device utilization summary area utilized for the nr4sd multiplier and radix-8 mb multiplier is noticed and shows the comparison clearly. radix-8 mb multiplier occupies less area compared to nr4sd multiplier.

VII. CONCLUSION

New designs of pre-encoded multipliers are evaluated by off-line encoding the standard coefficients and storing them in system memory. The proposed encoding these coefficients in the pre encoded radix-8 modified booth form. This pre-encoded mb multiplier design reduce the number of partial products because of increasing the radix value when compared to non redundant radix-4 signed digit encoding[NR4SD] form. Extensive experimental analysis verifies the better results in terms of area and delay when compared to nr4sd multiplier.

ACKNOWLEDGEMENT

The authors would like to thank our beloved principal Dr. G. Srinivasa Rao and vice Principal Dr. P. Srinivasa Raju for shri Vishnu engineering college for women for guidance and encouragement to do research by providing facilities and also like to acknowledge the anonymous reviewers for their suggestions that helps to improve the presentation of this paper.

REFERENCES