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

Yayathi Raju Thippani¹ | K. Swapna²

¹PG Scholar, Vaagdevi College of Engineering, Telangana.
²Associate Professor, Vaagdevi College of Engineering, Telangana.

ABSTRACT

In this paper, we introduce an architecture of pre-encoded multipliers for Digital Signal Processing applications based on off-line encoding of coefficients. To this extent, 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. Extensive experimental analysis verifies that the proposed pre-encoded NR4SD multipliers, including the coefficients memory, are more area and power efficient than the conventional Modified Booth scheme.

KEYWORDS: Multiplying circuits, Modified Booth encoding, Pre-Encoded multipliers, VLSI implementation

I. INTRODUCTION

Multimedia and Digital signal processing (DSP) applications (e.g., Fast Fourier Transform (FFT), audio/video CoDecs) 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.pre-determined coefficients attaining at the same time high efficiency.

Modified Booth (MB) encoding [4–7] 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 [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 audio [11], [12] and video [13], [14] CoDecs, 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]–[17], the encoding circuit of the MB multiplier is omitted. We refer to this design as pre-encoded MB multiplier. a condensed form (i.e,2 bits per digit).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 in-clude 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

Modified Booth (MB) is a redundant radix-4 encoding technique \[6\], \[7\]. 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:

\[
\begin{align*}
\text{TABLE 1} & \quad \text{Modified Booth Encoding} \\
\hline
b_{2j+1} & b_{2j} & b_{2j-1} & \text{mMB} & s & \text{one} & \text{two} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & +1 & 0 & 0 & 1 \\
0 & 1 & 0 & +1 & 0 & 0 & 1 \\
0 & 1 & 1 & +2 & 0 & 0 & 1 \\
1 & 0 & 0 & -2 & 1 & 0 & 1 \\
1 & 0 & 1 & -1 & 1 & 1 & 0 \\
1 & 1 & 0 & -1 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 \\
\end{align*}
\]

**TABLE 2**

**2’s complement form**

\[
\begin{align*}
\text{NR4SD form} & \quad \text{Digit} & \quad \text{NR4SD Encoding} \\
\hline
b_{2j+1} & b_{2j} & b_{2j-1} & n_{2j+1} & n_{2j} & b_{2j+1}^{\text{NR/SD}} & \text{onej} & \text{onej} & \text{twoj} \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & +1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & +1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & -2 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 & -2 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 1 & -1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\end{align*}
\]

**Fig. 2. Block Diagram of the NR4SD Encoding Scheme at the (a) Digit and (b) Word Level.**

1a). The outputs \(c_{2j+2}\) and \(n_{2j+1}\) of the HA* relate to its inputs as follows:

\[
2c_{2j+2} + n_{2j+1} = b_{2j+1} \cdot c_{2j+1}:
\]

The following Boolean equations summarize the HA* operation:

\[
c_{2j+2} = b_{2j+1} \cdot c_{2j+1} - 2_0^{2j}:
\]

Equation (5) results from the fact that \(n_{2j+1}\) is negatively signed and \(n_{2j+2}\) is positively signed.

Step 5: \(j := j + 1\).

where \(b = 0\). 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 \(b^{\text{MB}}\) as follows:

\[
b^{\text{MB}}_j = (-1)^s \cdot \text{(onej + 2twoj)}
\]

Equations (4) form the MB encoding signals.

\[
s_j = b_{2j+1}, \quad \text{onej} = b_{2j-1} \oplus b_{2j},
\]

\[
twoj = (b_{2j+1} \oplus b_{2j}) \wedge \text{onej}.
\]

**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:

The outputs of the HA* are analyzed at gate level in the following equations:

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

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

Step 4: Calculate the value of the \(b^{\text{NR/SD}}\) digit.
\( b_j^{NR} = 2n_2^{j+1} + n_2^j \) \hspace{1cm} (7)

Equation (7) results from the fact that \( n \cdot 2^{j+1} \) is positively signed and \( n \cdot 2^j \) 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. Equations (8) show how the NR4SD encoding signals one \( +j \), one \( -j \) and two \( +j \) of Table 4 are generated.

### Table 4

<table>
<thead>
<tr>
<th>2's Complement</th>
<th>10000000</th>
<th>10011010</th>
<th>01011001</th>
<th>01111111</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>-128</td>
<td>-102</td>
<td>+89</td>
<td>+127</td>
</tr>
<tr>
<td>Modified Booth</td>
<td>2000</td>
<td>2212</td>
<td>1221</td>
<td>2001</td>
</tr>
<tr>
<td>NR4SD-</td>
<td>2000</td>
<td>1212</td>
<td>2221</td>
<td>2001</td>
</tr>
<tr>
<td>NR4SD+</td>
<td>2000</td>
<td>2122</td>
<td>1121</td>
<td>2001</td>
</tr>
</tbody>
</table>

The generation of the \( i \)th bit \( p_{ij} \) of the partial product \( PP_j \) is illustrated at gate level in Fig. 4a [6], [7]. For the computation of the least and most significant bits of \( PP_j \), we consider \( a_1 = 0 \) and \( a_n = 1 \), respectively.

After shaping the partial products, they are added, properly weighted, through a Carry Save Adder (CSA) tree along with the correction term (COR):

The system architecture for the pre-encoded NR4SD multipliers is presented in Fig. 6. Two bits are now stored in ROM: \( n_2^{j+1}, n_2^j \) (Table 2) for the NR4SD or \( n \cdot 2^{j+1}, n_2^j \) (Table 3) for the NR4SD+ form. In this way, we reduce the memory requirement to \( n + 1 \) 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. The NR4SD encoding blocks of Fig. 6 implement the circuitry of Fig. 7.

Each partial product of the pre-encoded NR4SD and NR4SD* multipliers is implemented based on Fig. 4c and 4d, respectively, except for the PPk1 that corresponds to the most significant digit. As this digit is in MB form, we use the PPG of Fig. 4b applying the change mentioned in Section 4.2 for the sj bit. The partial products, properly weighted, and the correction term (COR) of (11) are fed into a CSA tree. The input carry cin;j of (11) is calculated as cin;j = twoi - onei and cin;j = onei for the NR4SD and NR4SD* pre-encoded multipliers, respectively, based on Tables 2 and 3. The carry-save output of the CSA tree is finally summed using a fast CLA adder.

### III. Implementation Results

We implemented in Verilog the multiplier designs of Table 5.

![Fig. 6. System Architecture of the NR4SD Multipliers.](image)

![Fig. 7. Extra Circuit Needed in the NR4SD Multipliers to Complete the (a) NR4SD and (b) NR4SD*](image)

Encoding of a negative digit. In order to avoid these inverters and, thus, reduce the area/power/delay of NR4SD, NR4SD* pre-encoded multipliers, the PPGs for the NR4SD, NR4SD* multipliers were designed based on primitive NAND and NOR gates, and replaced by Fig. 4e, 4f, respectively.

<table>
<thead>
<tr>
<th>Design</th>
<th>Input A</th>
<th>Input B Encoding</th>
<th>ROM Technique</th>
<th>ROM width</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional MB</td>
<td>MB</td>
<td>MB</td>
<td>n-bit</td>
<td></td>
</tr>
<tr>
<td>Pre-encoded MB</td>
<td>2's complement</td>
<td>MB</td>
<td>Fully Pre-Encoded</td>
<td>3n/2-bit</td>
</tr>
<tr>
<td>Pre-encoded NR4SD</td>
<td>NR4SD</td>
<td>Partially Pre-Encoded</td>
<td>(n+1)-bit</td>
<td></td>
</tr>
<tr>
<td>Pre-encoded NR4SD*</td>
<td>NR4SD</td>
<td>Pre-Encoded</td>
<td>(n+1)-bit</td>
<td></td>
</tr>
</tbody>
</table>

Memory compiler of the same library provided the physical ROMs for the coefficients. Since the ROMs required for the pre-encoded multipliers are larger than the one for the conventional MB scheme, access time is increased. However, the pre-encoded designs may achieve lower clock periods than the conventional MB one because the encoding circuits that are included in the critical path, are omitted or less complex. We first synthesized each design at the lowest achievable clock period and then, each pre-encoded design at the clock period achieved by the conventional MB scheme. We also synthesized all designs at higher clock periods targeting to explore their behavior under different timing constraints in terms of area and power consumption. For each clock period, we simulated all designs using Modelsim and 20 different sets of 512 ROM words. For the conventional MB multiplier, the 2's complement inputs were randomly generated with equal possibility of a bit to be 0 or 1. Using a high level programming language, we generated the pre-encoded values of B which we then stored in the ROMs of pre-encoded designs. Finally, we used Synopsys Prime Time to calculate power consumption.

The performance of the proposed designs is considered with respect to the width of the input numbers, i.e., 16, 24 and 32 bits. Table 6 summarizes the performance of each architecture at minimum possible clock period. We observe that the pre-encoded NR4SD architectures are more area efficient than the conventional or pre-encoded MB designs with respect to their performance in the lowest possible clock periods. Regarding power dissipation, the pre-encoded NR4SD scheme consumes the least power which, in the cases of 16 and 24 bits of input width, is equal to the power consumed by the pre-encoded MB design.
TABLE 6
Performance at Lowest Clock Periods

<table>
<thead>
<tr>
<th>Design</th>
<th>$D^A$</th>
<th>$A^D$</th>
<th>AD</th>
<th>$P_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional MB</td>
<td>2.01</td>
<td>18945</td>
<td>38079</td>
<td>11.20</td>
</tr>
<tr>
<td>MB</td>
<td>1.95</td>
<td>19079</td>
<td>37205</td>
<td>11.00</td>
</tr>
<tr>
<td>NR4SD</td>
<td>1.95</td>
<td>18357</td>
<td>35796</td>
<td>11.00</td>
</tr>
<tr>
<td>NR4SD$^*$</td>
<td>1.95</td>
<td>18485</td>
<td>36045</td>
<td>11.10</td>
</tr>
<tr>
<td>16 bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-encoded</td>
<td>2.26</td>
<td>32354</td>
<td>73121</td>
<td>18.80</td>
</tr>
<tr>
<td>MB</td>
<td>2.18</td>
<td>30251</td>
<td>65948</td>
<td>16.80</td>
</tr>
<tr>
<td>NR4SD</td>
<td>2.18</td>
<td>28822</td>
<td>62832</td>
<td>16.80</td>
</tr>
<tr>
<td>NR4SD$^*$</td>
<td>2.18</td>
<td>29573</td>
<td>64470</td>
<td>17.40</td>
</tr>
<tr>
<td>24 bits</td>
<td>32 bits</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-encoded</td>
<td>2.34</td>
<td>50199</td>
<td>117465</td>
<td>29.50</td>
</tr>
<tr>
<td>MB</td>
<td>2.28</td>
<td>47490</td>
<td>108277</td>
<td>27.30</td>
</tr>
<tr>
<td>NR4SD</td>
<td>2.28</td>
<td>43779</td>
<td>99816</td>
<td>26.40</td>
</tr>
<tr>
<td>NR4SD$^*$</td>
<td>2.29</td>
<td>46318</td>
<td>106068</td>
<td>27.50</td>
</tr>
</tbody>
</table>

a. Delay for ROM+Multiplier in ns.
b. Area occupation for ROM+Multiplier in $\text{um}^2$.
c. Power consumption for ROM+Multiplier in mW.

IV. 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. The proposed pre-encoded NR4SD multiplier designs are more area and power efficient compared to the conventional and pre-encoded MB designs. Extensive experimental analysis verifies the gains of the proposed pre-encoded NR4SD multipliers in terms of area complexity and power consumption compared to the conventional MB multiplier.

REFERENCES