Pre-Encoded Multipliers Based on Non-Redundant Radix-8 Signed-Digit Encoding

B.Sunny Manohar
Assistant Professor, Dhruva Institute of Engineering & Technology

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 extend, the Non-Redundant radix-8 Signed-Digit (NR8SD) encoding technique, which uses the digit values \(f 1; 0; +1; +2g\) or \(f 2; 1; 0; +1g\), is proposed leading to a multiplier design with less complex partial products implementation. Extensive experimental analysis verifies that the proposed pre-encoded NR8SD multipliers, including the coefficients memory, are more area and power efficient than the conventional Modified Booth scheme.

Index Terms: 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. 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]–[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 [9], [10], multipliers included in butterfly units of FFT processors use standard coefficients stored in ROMs. 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. Then, we explore a Non-Redundant radix-8 Signed-Digit (NR8SD) encoding scheme extending the serial encoding techniques of [6], [18]. The proposed NR8SD encoding scheme uses one of the following sets of digit values: \( \{1, 0, +1, +2\} \) or \( \{2, 1, 0, +1\} \). In order to cover the dynamic range of the 2’s complement form, all digits of the proposed representation are encoded according to NR8SD 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 NR8SD scheme reduces the memory size. Also, compared to the MB form, which uses five digit values \( \{2, 1, 0, +1, +2\} \), the proposed NR8SD encoding uses four digit values. Thus, the NR8SD-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.

### TABLE 1

**Modified Booth Encoding**

<table>
<thead>
<tr>
<th>( b_{2j+1} )</th>
<th>( b_j )</th>
<th>( s_j )</th>
<th>one</th>
<th>two</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>+1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### 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:

\[
B = \sum_{j=0}^{k-1} b_j MB_j 2^j = \sum_{j=0}^{2k} b_j 2^j \quad \text{with} \quad b_j = \begin{cases} 2, & \text{if } b_j = 2 \\ f 2, & \text{if } b_j = 1 \\ 0, & \text{if } b_j = 0 \\ +1, & \text{if } b_j = +1 \\ +2, & \text{if } b_j = +2 \\ +1g, & \text{if } b_j = +1g \end{cases}
\]

\[
= h b_{MB}^{MB} k_1 : \cdots : b_{MB}^{MB} 2^j + b_{MB}^{MB} 2^j = b_{MB}^{MB} 2^j + b_{MB}^{MB} 2^j + b_{MB}^{MB} 2^j + b_{MB}^{MB} 2^j + b_{MB}^{MB} 2^j .
\]

Digits \( b_{MB}^{MB} j 2 f 2; 1; 0; +1; +2g; 0 j k 1 \), are formed as follows:

\[
b_{MB}^{MB} j = 2b_{2j+1} + b_{2j} + b_{2j} 1.
\]
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 ($\text{one}=1$) or not ($\text{one}=0$). Two shows if the absolute value of a digit equals 2 ($\text{two}=1$) or not ($\text{two}=0$). Using these bits, we calculate the MB digits $b_{MB}^j$ as follows:

$$b_{MB}^j = (1)^{s_j} (\text{one}_j + 2\text{two}_j);$$

Equations (4) form the MB encoding signals.

$$\text{two}_j = (b_{2j+1} \land b_{2j}) \land \text{one}_j;$$

III. NON-REDUNDANT RADIX-8 SIGNED-DIGIT ALGORITHM

In this section, we present the Non-Redundant radix-4 Signed-Digit (NR8SD) encoding technique. As in MB form, the number of partial products is reduced to half. When encoding the 2’s complement number $B$,

![Block Diagram of the NR8SD Encoding Scheme at the (a) Digit and (b) Word Level.](image-url)
digits $b_{NR}^j$ take one of four values: $2, 1, 0, +1$ or $b_{NR}^j + 2 f 1, 0, +1, +2$ at the NR8SD or NR8SD+ algorithm, respectively. Only four different values are used and not five as in MB algorithm, which leads to 0 $k 2$. As we need to cover the dynamic range of the 2’s complement form, the most significant digit is MB encoded (i.e., $b_{MB}^k 2 f 2, 1, 0, +1, +2$).

The NR8SD and NR8SD+ encoding algorithms are illustrated in detail in Fig. 1 and 2, respectively.

NR8SD Algorithm

Step 1: Consider the initial values $j = 0$ and $c0=0$. Step 2: Calculate the carry $c2j+1$ and the sum $n+2j$ of a Half Adder (HA) with inputs $b2j$ and $c2j$ (Fig. 1a).

$c2j+1 = b2j \land c2j; n+2j = b2j c2j$.

Step 3: Calculate the positively signed carry $c2j+1$ (+) and the negatively signed sum $n2j+1$ (−) of a Half Adder* (HA*) with inputs $b2j+1$ (+) and $c2j$ (+).

Step 3: Calculate the positively signed carry $c2j+2$ (+) and the negatively signed sum $n2j+1$ (−) of a Half Adder* (HA*) with inputs $b2j+1$ (+) and $c2j+1$ (+) (Fig. 2).

**TABLE 2**

<table>
<thead>
<tr>
<th>2’s complement</th>
<th>NR8SD form</th>
<th>Digit</th>
<th>NR8SD Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3 +2$</td>
<td>$2^2$</td>
<td>$n2j+1 n2j$</td>
<td>$b_{NR}^j$</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 0 1 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 1 1 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

**TABLE 3**

<table>
<thead>
<tr>
<th>2’s complement</th>
<th>NR8SD+ form</th>
<th>Digit</th>
<th>NR8SD+ Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^3$</td>
<td>$2^2$</td>
<td>$n2j+1 n2j$</td>
<td>$b_{NR}^j$</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>0 1 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>1 0 1 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 1 1 1</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

Fig. 2. Block Diagram of the NR8SD+ Encoding Scheme at the (a) Digit and (b) Word Level.
International Journal of Advance Research in Science and Engineering

Vol. No.6, Issue No. 07, July 2017

www.ijarse.com

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

$$2c = b + c :$$

The following Boolean equations summarize the HA$^*$ operation:

$$c = b \cdot c ; \quad n = b \cdot c :$$

Step 4: Calculate the value of the $b_{NR}^{j}$ digit.

$$b_{NR}^{j} = 2n_{2j+1} + n_{2j}$$

Equation (5) 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 $(j < k)$, go to Step 2. If $(j = k)$, encode the most significant digit based on the MB algorithm and considering the three consecutive bits to be $b_{2k} \cdot b_{2k}$ and $c_{2k}$ (Fig. 1b). If $(j = k)$, stop.

Table 2 shows how the NR8SD digits are formed. Equations (6) show how the NR8SD encoding signals one$^+$$_{j}$, one$^-$$_{j}$ and two$^+$_{j} of Table 2 are generated.

$$\begin{align*}
\text{one}^+_{j} &= n_{2j+1} \cdot n_{2j} \\
\text{one}^-_{j} &= n_{2j+1} \cdot n_{2j} \\
\text{two}^+_{j} &= n_{2j+1} \cdot n_{2j}
\end{align*}$$

The minimum and maximum limits of the dynamic range in the NR8SD form are $2^n - 2^{n-1} \leq 2^n - 2^{n-1} < 2^n$. We observe that the NR8SD form has larger dynamic range than the 2's complement form.

NR8SD$^+$ 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:

$$2c_{2j+1} = b_{2j} + c_{2j}$$

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

$$c_{2j+1} = b_{2j} - c_{2j} ; \quad n_{2j} = b_{2j} \cdot c_{2j}$$

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

$$c = b \cdot c ; \quad n = b \cdot c :$$

Step 4: Calculate the value of the $b_{NR}^{j+1}$ digit.

$$b_{NR}^{j+1} = 2n_{2j+1} + n_{2j}$$

Equation (7) 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 $(j < k)$, go to Step 2. If $(j = k)$, encode the most significant digit according to MB algorithm and considering the three consecutive bits to be $b_{2k} \cdot b_{2k}$ and $c_{2k}$ (Fig. 2b). If $(j = k)$, stop.

Table 3 shows how the NR8SD$^+$ digits are formed. Equations (8) show how the NR8SD$^+$ encoding signals one$^+_{j}$, one$^-_{j}$ and two$^+_{j}$ of Table 4 are generated.

$$\begin{align*}
\text{one}^+_{j} &= n_{2j+1} + n_{2j} \\
\text{one}^-_{j} &= n_{2j+1} + n_{2j} \\
\text{two}^+_{j} &= n_{2j+1} + n_{2j}
\end{align*}$$

The minimum and maximum limits of the dynamic range in the NR8SD$^+$ form are $2^n - 2^{n-1} \leq 2^{n-1} < 2^n - 2^{n-1}$. As observed in the NR8SD encoding technique, the NR8SD$^+$ form has larger dynamic range than the 2's complement form.

Considering the 8-bit 2's complement number $N$, Table 4 exposes the limit values $2^8 = 128$, $2^8 = 127$, and two typical values of $N$ and presents the MB, NR8SD and NR8SD$^+$ digits 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 MULTIPLIERS DESIGN

In this section, we explore the implementation of pre-encoded multipliers. One of the two inputs of these multipliers is pre-encoded either in MB or in NR8SD / NR8SD’ representation. We consider that this input comes from a set of fixed coefficients (e.g. the coefficients for a number of filters in which this multiplier will be used in a dedicated system or the sine table required in an FFT implementation). The coefficients are encoded off-line based on MB or NR8SD 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.

where \( c_{MB} = (\text{one, two})^{\times} b_{4} \) (Table 1). The CS output of the tree is led to a fast Carry Look Ahead (CLA) adder [19] to form the final result \( P = A \times B \) (Fig. 3).

4.2 Pre-Encoded MB Multiplier Design

In the pre-encoded MB multiplier scheme, the coefficient B is encoded off-line according to the conventional MB form (Table 1). The resulting encoding signals of B are stored in a ROM. The circled part of Fig. 3, which contains the ROM with coefficients in 2’s complement form and the MB encoding circuit, is now totally replaced by the ROM of Fig. 5. The MB encoding blocks of Fig. 3 are omitted. The new ROM of Fig. 5 is used to store the encoding signals of B and feed them into the partial product generators.
4.1 Conventional MB Multiplier

Fig. 3 presents the architecture of the system which comprises the conventional MB multiplier and the ROM with coefficients in 2's complement form. Let us consider the multiplication A \times B. The coefficient B = b_n b_{n-1} \ldots b_1 b_0, consists of n=2k bits and is driven to the MB encoding blocks from a ROM where it is stored in 2's complement form. It is encoded according to the MB algorithm (Section 2) and multiplied by A = a_n a_{n-1} \ldots a_1 a_0, which is in 2's complement representation. We note that the ROM data bus width equals the width of coefficient B (n bits) and that it outputs one coefficient on each clock cycle.

The k partial products are generated as follows:

$$PP_i = A b_i = P_{MB} a_i + \sum_{k=1}^{n-1} a_{n-k} b_i$$

As a result, the PPG of Fig. 4a is replaced by the one of Fig. 4b. Compared to (4), (12) leads to a more compact design. However, due to the pre-encoding technique, there is area overhead at the circuit. The partial products, properly weighted, and the correction term (COR) of (11) are fed into a CSA tree. The input carry \(c_{in1} = c_{s1} \) based on (12) and Table 1. The CS output of the tree is finally merged by a fast CLA adder.

However, the ROM width is increased. Each digit requests three encoding bits (i.e., s, t, and one (Table 1)) to be stored in the ROM. 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 (Fig. 3).

4.3 Pre-Encoded NR8SD Multipliers Design

The system architecture for the pre-encoded NR8SD multipliers is presented in Fig. 6. Two bits are now stored in ROM: \(n_{2j+1}, n_{2j}\) (Table 2) for the NR8SD or \(n_{2j+1}, n_{2j}\) (Table 3) for the NR8SD\(^+\) 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 multipler, where the MB encoding blocks are omitted, the pre-encoded NR8SD multipliers need extra hardware to generate the signals of (8) and (9) for the NR8SD and NR8SD\(^+\) form, respectively. The NR8SD encoding blocks of Fig. 6 implement the circuitry of Fig. 7.
Each partial product of the pre-encoded NR8SD and NR8SD+ multipliers is implemented based on Fig. 4c and 4d, respectively, except for the PP_k in that corresponds to the most significant digit. As this digit is in MB form, we use the PPG of Fig. 4d applying the change mentioned in Section 4.2 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 \( c_{i+1} \) of (11) is calculated as \( c_{i+1} = \text{two} \cdot c_i \) and \( c_{i+1} = \text{one} \) for the NR8SD and NR8SD+ 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.

5 IMPLEMENTATION RESULTS

We implemented in Verilog the multiplier designs of Table 5. The PPGs for the NR8SD, NR8SD+ multipliers (Fig. 4c, 4d, respectively) contain a large number of inverters since all the A bits are complemented in case of a negative digit. In order to avoid these inverters and, thus, reduce the area/power/delay of NR8SD, NR8SD+ pre-encoded multipliers, the PPGs for the NR8SD, NR8SD+ multipliers were designed based on primitive NAND and NOR gates, and replaced by Fig. 4c, 4f, respectively.

The CSA tree and CLA adder were imported from Synopsys DesignWare library. The ROM for the 2's complement or pre-encoded coefficients is a synchronous ROM of 512 words often met at DSP systems, e.g., speech CoDecs or audio filtering [20]. The width of each ROM depends on the multiplier architecture (Table 5). A finite state machine synchronized the data flow and the multiplier operation but was not considered in the area/power calculations.

We used Synopsys Design Compiler and the Faraday 90 nm standard cell library to synthesize the evaluated designs, considering the highest optimization degree and keeping the hierarchy of the designs. The 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

<table>
<thead>
<tr>
<th>Design</th>
<th>ROM width</th>
<th>Input B Encoding Type</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional MB</td>
<td>n-bit</td>
<td>MB</td>
<td>MB encoding</td>
</tr>
<tr>
<td>MB</td>
<td>n+1-bit</td>
<td>MB</td>
<td>Fully Pre-Encoded</td>
</tr>
<tr>
<td>2's complement</td>
<td>n+1-bit</td>
<td>NR8SD</td>
<td>Pre-Encoded</td>
</tr>
<tr>
<td>NR8SD</td>
<td>n+1-bit</td>
<td>NR8SD+</td>
<td>Partially</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design</th>
<th>D</th>
<th>A</th>
<th>A.D</th>
<th>P</th>
<th>P.D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional MB</td>
<td>2.01</td>
<td>18045</td>
<td>38079</td>
<td>11.20</td>
<td>22.51</td>
</tr>
<tr>
<td>MB</td>
<td>1.05</td>
<td>10070</td>
<td>37205</td>
<td>11.00</td>
<td>21.45</td>
</tr>
<tr>
<td>NR8SD</td>
<td>1.05</td>
<td>18357</td>
<td>35790</td>
<td>11.00</td>
<td>21.45</td>
</tr>
<tr>
<td>NR8SD+</td>
<td>1.05</td>
<td>18450</td>
<td>36045</td>
<td>11.10</td>
<td>21.65</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design</th>
<th>D</th>
<th>A</th>
<th>A.D</th>
<th>P</th>
<th>P.D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional MB</td>
<td>2.26</td>
<td>32354</td>
<td>73121</td>
<td>16.00</td>
<td>42.49</td>
</tr>
<tr>
<td>MB</td>
<td>2.18</td>
<td>30251</td>
<td>65948</td>
<td>16.00</td>
<td>36.62</td>
</tr>
<tr>
<td>NR8SD</td>
<td>2.18</td>
<td>28822</td>
<td>62832</td>
<td>16.00</td>
<td>36.62</td>
</tr>
<tr>
<td>NR8SD+</td>
<td>2.18</td>
<td>29573</td>
<td>64470</td>
<td>17.40</td>
<td>37.53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Design</th>
<th>D</th>
<th>A</th>
<th>A.D</th>
<th>P</th>
<th>P.D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional MB</td>
<td>2.34</td>
<td>50190</td>
<td>117455</td>
<td>29.50</td>
<td>69.03</td>
</tr>
<tr>
<td>MB</td>
<td>2.28</td>
<td>47490</td>
<td>108277</td>
<td>27.30</td>
<td>62.24</td>
</tr>
<tr>
<td>NR8SD</td>
<td>2.28</td>
<td>43779</td>
<td>99610</td>
<td>26.40</td>
<td>60.19</td>
</tr>
<tr>
<td>NR8SD+</td>
<td>2.28</td>
<td>46318</td>
<td>100668</td>
<td>27.50</td>
<td>62.98</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
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 PrimeTime 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, 32 & 64 bits. Table 6 summarizes the performance of each architecture at minimum possible clock period. We observe that the pre-encoded NR8SD 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 NR8SD 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.

With respect to the input width, Fig. 8-10 depict the area and power gains that the system (i.e., ROM + multiplier), the multiplier and the PPG of the pre-encoded MB, NR8SD and NR8SD+ designs present over the conventional MB scheme. The comparison among the designs starts at the lowest common achievable clock period for all designs and continues at higher clock periods by increasing the clock period by step 0.2 ns until it reaches 4 ns. We first compare the entire designs incorporating the required ROMs. Then, we make a comparison among the multipliers of all schemes as they are implemented based on different encoding techniques. Also, we compare the PPGs of the multipliers because they are key components occupying significant area in the multipliers.

In Fig. 8-10, the pre-encoded MB scheme delivers losses in area complexity (9.47%, 7.71% and 6.44% on average at 16, 24, 32 and 64 bits, respectively) and power consumption (7.25%, 9.03% and 7.01% on average at 16, 24 and 32 bits, respectively). This was expected considering that the size of the ROM required by the pre-encoded MB design is by 50% larger than the ROM of the conventional MB scheme. However, the proposed pre-encoded NR8SD designs (ROM and multiplier) deliver improvements in area complexity (up to 7.28% on average for the pre-encoded NR8SD design at 32 and 64 bits) and power dissipation (up to 9.46% on average for the pre-encoded NR8SD+ design at 24 bits) compared to the conventional MB scheme. We note that the gains that concern the multipliers of the pre-encoded NR8SD designs over the one of the conventional MB scheme are much higher. This is mainly due to the less complex PPG circuit of the NR8SD designs (Fig. 4e, 4f) compared to the one of the conventional MB design (Fig. 4a) considering that the partial products generation largely contributes to the area complexity and power dissipation of a multiplier. Fig. 8-10 verify the area and power gains of the PPG of the NR8SD designs over one of the conventional MB scheme.

1. The average gains/losses for a specific input width are calculated considering the gains/losses over the conventional MB scheme for all clock periods that concern the input width of interest.

Fig. 8. Area / Power Gains of the Pre-Encoded Designs Over the Conventional MB Scheme at 16 Bits.
Fig. 9. Area / Power Gains of the Pre-Encoded Designs Over the Conventional MB Scheme at 24 Bits.

As clock period increases, the datapath of the multiplication circuit changes and the standard cells used for its synthesis become less complex regarding area occupation, internal capacitance and ports' load. However, the ROM used in each evaluated design is a standard cell and its critical delay, area occupation and both internal and ports' load remain unchanged as clock period increases. Thus, the multiplier changes sharply as clock period increases but the ROM does not change. The power dissipation of the multiplier is sharply decreased as clock period increases since both frequency and overall load charge decrease, while the power consumption of the ROM is linearly decreased following the frequency reduction. Also, the experimental analysis is based on ROMs generated using the memory compiler of the Faraday 90 nm standard cell library, but the measurements of the systems (i.e., ROM + Multiplier) could change using memories of emerging technologies [21]. Thus, the area and power values for the multipliers of the designs are useful for explorations of the proposed pre-encoded designs based on different memory technologies.

Fig. 10. Area / Power Gains of the Pre-Encoded Designs Over the Conventional MB Scheme at 32 Bits.
VI. 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-8 Signed-Digit (NR8SD) form. The proposed pre-encoded NR8SD 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 NR8SD multipliers in terms of area complexity and power consumption compared to the conventional MB multiplier.

REFERENCES