This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428691, IEEE Transactions on Computers

Pre-Encoded Multipliers Based on Non-Redundant Radix-4 Signed-Digit Encoding

K. Tsoumanis, N. Axelos, N. Moshopoulos, G. Zervakis and K. Pekmestzi

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-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.

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

1 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-4 Signed-Digit (NR4SD) encoding scheme extending the serial encoding techniques of [6], [18]. The proposed NR4SD 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 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

• *K. Tsoumanis, N. Axelos, N. Moshopoulos, G. Zervakis and K. Pekmestzi are with the Department of Electrical and Computer Engineering, National Technical University of Athens. E-mail: {kostastsoumanis, njaxel, nikos, zervakis, pekmes}@microlab.ntua.gr.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428691, IEEE Transactions on Computers

TABLE 1
Modified Booth Encoding

b_{2j+1}	b_{2j}	b_{2j-1}	$\mathbf{b}_j^{MB}$	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

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.

2 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:

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

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

$$\mathbf{b}_j^{MB} = -2b_{2j+1} + b_{2j} + b_{2j-1}, \tag{2}$$

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 $\mathbf{b}_j^{MB}$ as follows:

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

Equations (4) form the MB encoding signals.

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

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

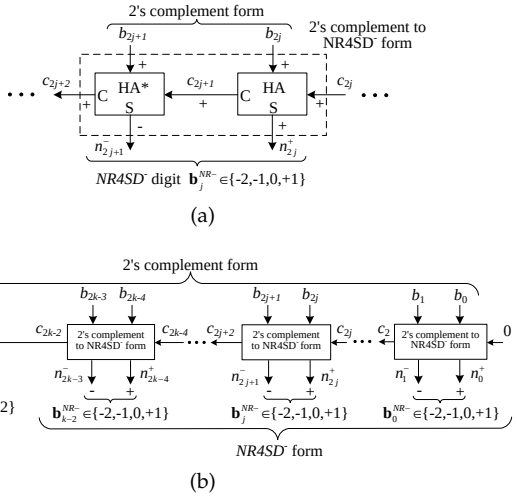

Fig. 1. Block Diagram of the NR4SD$^-$ Encoding Scheme at the (a) Digit and (b) Word Level.

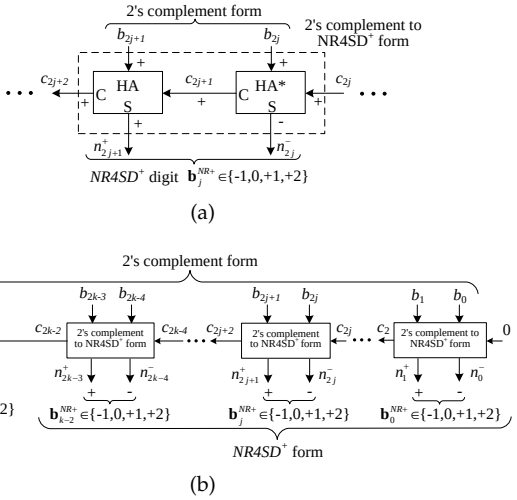

Fig. 2. Block Diagram of the NR4SD$^+$ Encoding Scheme at the (a) Digit and (b) Word Level.

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).

$$c_{2j+1} = b_{2j} \wedge c_{2j}, \quad n_{2j}^+ = b_{2j} \oplus c_{2j}.$$

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} (+) (Fig.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428691, IEEE Transactions on Computers

TABLE 2
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}	$\mathbf{b}_j^{NR-}$	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

TABLE 3
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}	$\mathbf{b}_j^{NR+}$	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

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

$$2c_{2j+2} - n^-_{2j+1} = b_{2j+1} + c_{2j+1}.$$

The following Boolean equations summarize the HA* operation:

$$c_{2j+2} = b_{2j+1} \vee c_{2j+1}, \quad n^-_{2j+1} = b_{2j+1} \oplus c_{2j+1}.$$

Step 4: Calculate the value of the $\mathbf{b}_j^{NR-}$ digit.

$$\mathbf{b}_j^{NR-} = -2n^-_{2j+1} + n^+_{2j}. \tag{5}$$

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 ($j < k-1$), 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_{2k-1}, b_{2k-2} and c_{2k-2} (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.

$$\begin{aligned} one_j^+ &= \overline{n^-_{2j+1}} \wedge n^+_{2j}, \\ one_j^- &= n^-_{2j+1} \wedge n^+_{2j}, \\ two_j^- &= n^-_{2j+1} \wedge \overline{n^+_{2j}}. \end{aligned} \tag{6}$$

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

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:

$$c_{2j+1} = b_{2j} \vee c_{2j}, \quad n^-_{2j} = b_{2j} \oplus c_{2j}.$$

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}.

$$c_{2j+2} = b_{2j+1} \wedge c_{2j+1}, \quad n^+_{2j+1} = b_{2j+1} \oplus c_{2j+1}.$$

Step 4: Calculate the value of the $\mathbf{b}_j^{NR+}$ digit.

$$\mathbf{b}_j^{NR+} = 2n^+_{2j+1} - n^-_{2j}. \tag{7}$$

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 ($j < k-1$), 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.

$$\begin{aligned} one_j^+ &= n^+_{2j+1} \wedge n^-_{2j}, \\ one_j^- &= \overline{n^+_{2j+1}} \wedge n^-_{2j}, \\ two_j^+ &= n^+_{2j+1} \wedge \overline{n^-_{2j}}. \end{aligned} \tag{8}$$

The minimum and maximum limits of the dynamic range in the NR4SD$^+$ form are $-2^{n-1} - 2^{n-4} - 2^{n-6} - \cdots - 1 < -2^{n-1}$ and $2^{n-1} + 2^{n-3} + 2^{n-5} + \cdots + 2 > 2^{n-1} - 1$. 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 4 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. We added a bar above the negatively signed digits in order to distinguish them from the positively signed ones.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428691, IEEE Transactions on Computers

TABLE 4
Numerical Examples of the Encoding Techniques

2's Complement	10000000	10011010	01011001	01111111
Integer	-128	-102	+89	+127
Modified Booth	$\bar{2}\,0\,0\,0$	$\bar{2}\,2\,\bar{1}\,\bar{2}$	$1\,2\,\bar{2}\,1$	$2\,0\,0\,\bar{1}$
NR4SD$^-$	$\bar{2}\,0\,0\,0$	$\bar{1}\,\bar{2}\,\bar{1}\,\bar{2}$	$2\,\bar{2}\,\bar{2}\,1$	$2\,0\,0\,\bar{1}$
NR4SD$^+$	$\bar{2}\,0\,0\,0$	$\bar{2}\,1\,2\,2$	$1\,1\,2\,1$	$2\,0\,0\,\bar{1}$

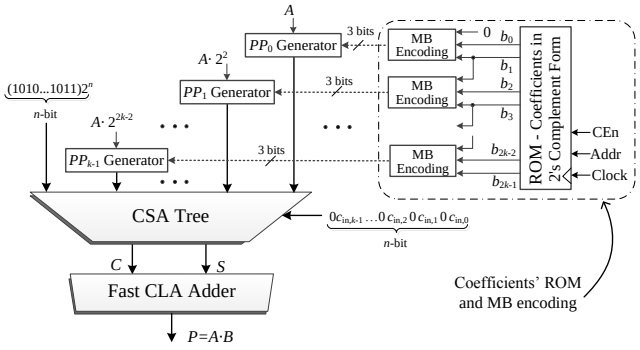

Fig. 3. System Architecture of the Conventional MB Multiplier.

4 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 NR4SD$^-$ / NR4SD$^+$ 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 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.

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 $A \cdot B$. The coefficient $B = \langle b_{n-1} \dots b_0 \rangle_{2's}$ 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 $A = \langle a_{n-1} \dots a_0 \rangle_{2's}$, 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:

$$PP_j = A \cdot \mathbf{b}_j^{MB} = \bar{p}_{j,n} 2^n + \sum_{i=0}^{n-1} p_{j,i} 2^i. \qquad (9)$$

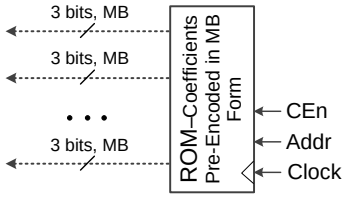

Fig. 5. The ROM of Pre-Encoded Multiplier with Standard Coefficients in MB Form.

The generation of the i^{th} bit $p_{j,i}$ 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 $a_n = a_{n-1}$, respectively.

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

$$P = A \cdot B = COR + \sum_{j=0}^{k-1} PP_j 2^{2j}, \qquad (10)$$

$$COR = \sum_{j=0}^{k-1} c_{in,j} 2^{2j} + 2^n (1 + \sum_{j=0}^{k-1} 2^{2j+1}), \qquad (11)$$

where $c_{in,j} = (one_j \vee two_j) \wedge s_j$ (Table 1). The CS output of the tree is leaded to a fast Carry Look Ahead (CLA) adder [19] to form the final result $P = A \cdot B$ (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 (PP_j Generators - PPG) on each clock cycle.

Targeting to decrease switching activity, the value '1' of s_j in the last entry of Table 1 is replaced by '0'. The sign s_j is now given by the relation:

$$s_j = b_{2j+1} \oplus (b_{2j+1} \wedge b_{2j} \wedge b_{2j-1}). \qquad (12)$$

As a result, the PPG of Fig. 4a is replaced by the one of Fig. 4b. Compared to (4), (12) leads to a more complex design. However, due to the pre-encoding technique, there is no area / delay 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 $c_{in,j}$ of (11) is computed as $c_{in,j} = s_j$ 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*, *two* and *one* (Table

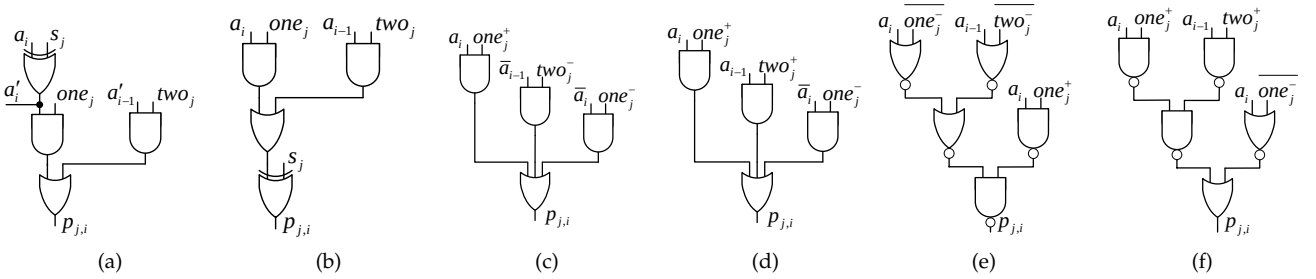

Fig. 4. Generation of the i^{th} Bit $p_{j,i}$ of PP_j for a) Conventional, b) Pre-Encoded MB Multipliers, c) NR4SD$^-$, d) NR4SD$^+$ Pre-Encoded Multipliers, and e) NR4SD$^-$, f) NR4SD$^+$ Pre-Encoded Multipliers after reconstruction.

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 NR4SD Multipliers Design

The system architecture for the pre-encoded NR4SD multipliers is presented in Fig. 6. Two bits are now stored in ROM: n^-_{2j+1}, n^+_{2j} (Table 2) for the NR4SD$^-$ or n^+_{2j+1}, n^-_{2j} (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 PP_{k-1} 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 s_j bit. The partial products, properly weighted, and the correction term (*COR*) of (11) are fed into a CSA tree. The input carry $c_{in,j}$ of (11) is calculated as $c_{in,j} = two^-_j \vee one^-_j$ and $c_{in,j} = one^-_j$ 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.

5 Implementation Results

We implemented in Verilog the multiplier designs of Table 5. The PPGs for the NR4SD$^-$, NR4SD$^+$ multipliers (Fig. 4c, 4d, respectively) contain a large number of inverters since all the A bits are complemented in case

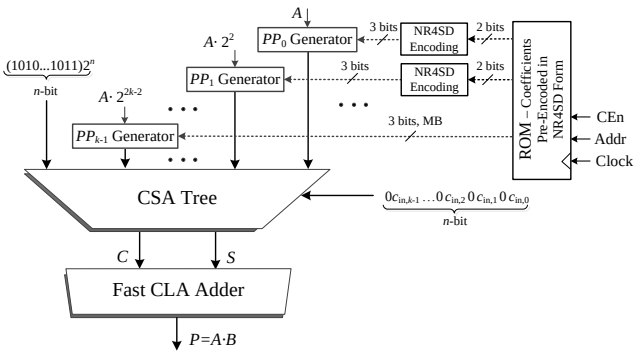

Fig. 6. System Architecture of the NR4SD Multipliers.

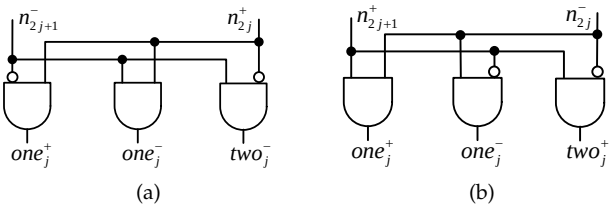

Fig. 7. Extra Circuit Needed in the NR4SD Multipliers to Complete the (a) NR4SD$^-$ and (b) NR4SD$^+$ Encoding.

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.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428691, IEEE Transactions on Computers

TABLE 5
Multiplier Designs

Design		*Input A*	*Input B Encoding*		*ROM width*
			Type	*Technique*	
Conventional MB		2's complement n-bit	MB	MB encoding	n-bit
Pre-Encoded	MB		MB	Fully Pre-Encoded	$3n/2$-bit
	NR4SD$^-$		NR4SD$^-$	Partially Pre-Encoded	$(n+1)$-bit
	NR4SD$^+$		NR4SD$^+$		$(n+1)$-bit

TABLE 6
Performance at Lowest Clock Periods

Design		D[a]	A[b]	A×D	P[c]	P×D
16 bits						
Conventional MB		2.01	18945	38079	11.20	22.51
Pre-Encoded	MB	1.95	19079	37205	11.00	21.45
	NR4SD$^-$	1.95	18357	35796	11.00	21.45
	NR4SD$^+$	1.95	18485	36045	11.10	21.65
24 bits						
Conventional MB		2.26	32354	73121	18.80	42.49
Pre-Encoded	MB	2.18	30251	65948	16.80	36.62
	NR4SD$^-$	2.18	28822	62832	16.80	36.62
	NR4SD$^+$	2.18	29573	64470	17.40	37.93
32 bits						
Conventional MB		2.34	50199	117465	29.50	69.03
Pre-Encoded	MB	2.28	47490	108277	27.30	62.24
	NR4SD$^-$	2.28	43779	99816	26.40	60.19
	NR4SD$^+$	2.29	46318	106068	27.50	62.98

a. Delay for ROM+Multiplier in ns.
b. Area occupation for ROM+Multiplier in um^2.
c. Power consumption for ROM+Multiplier in mW.

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 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 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.

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, NR4SD$^-$ and NR4SD$^+$ 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 subcomponents 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[1] at 16, 24 and 32 bits, respectively) and power consumption (-7.25%, -9.08% 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 NR4SD designs (ROM and multiplier) deliver improvements in area complexity (up to 7.28% on average for the pre-encoded NR4SD$^-$ design at 32 bits) and power dissipation (up to 9.46% on average for the pre-encoded NR4SD$^+$ design at 24 bits) compared to the conventional MB scheme. We note that the gains that concern the multipliers of the pre-encoded NR4SD designs over the one of the conventional MB scheme are much higher. This is mainly due to the less complex PPG circuit of the NR4SD 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 NR4SD designs over the 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TC.2015.2428691, IEEE Transactions on Computers

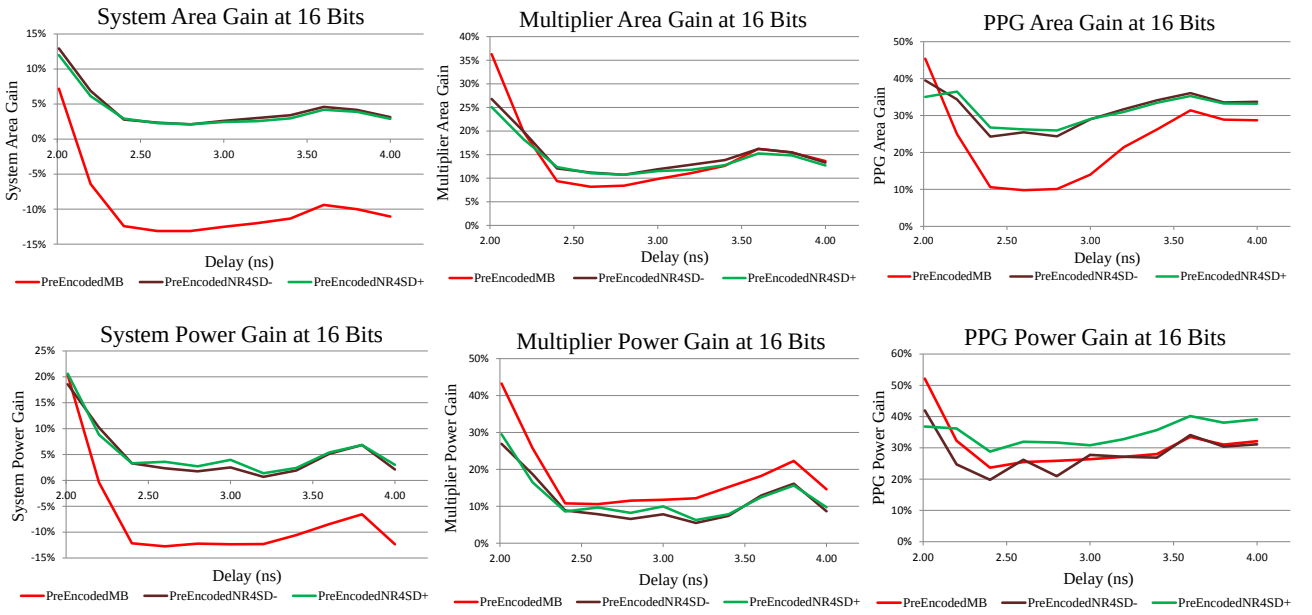

Fig. 8. Area / Power Gains of the Pre-Encoded Designs Over the Conventional MB Scheme at 16 Bits.

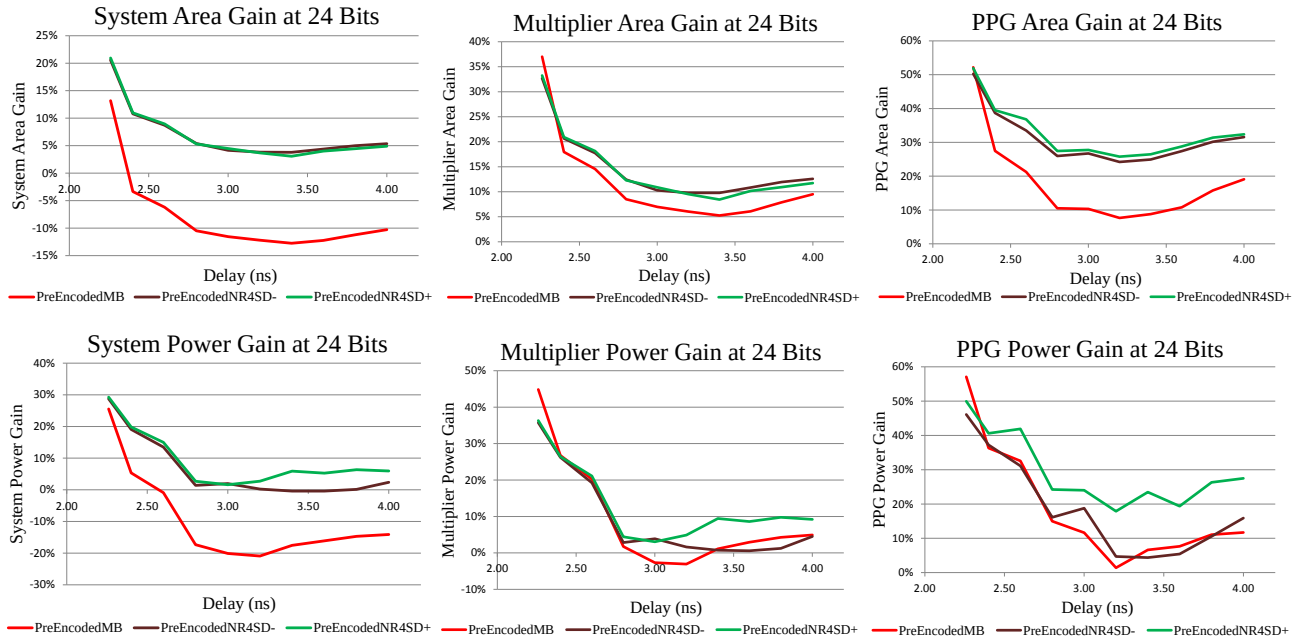

Fig. 9. Area / Power Gains of the Pre-Encoded Designs Over the Conventional MB Scheme at 24 Bits.

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.

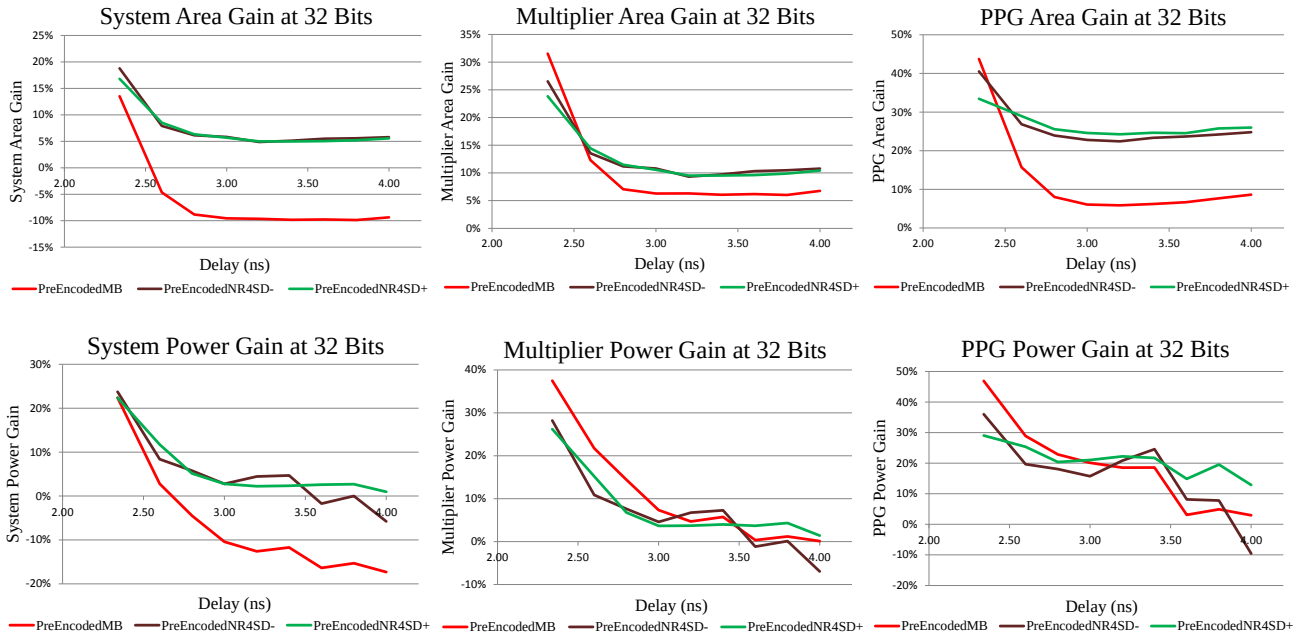

Fig. 10. Area / Power Gains of the Pre-Encoded Designs Over the Conventional MB Scheme at 32 Bits.

6 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

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

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

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

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

[11] Y. T. Han, J. S. Koh, and S. H. Kwon, "Synthesis filter for mpeg-2 audio decoder," Patent US 5 812 979, Sep., 1998.

[12] M. Kolluru, "Audio decoder core constants rom optimization," Patent US 6 108 633, Aug., 2000.

[13] H.-Y. Lin, Y.-C. Chao, C.-H. Chen, B.-D. Liu, and J.-F. Yang, "Combined 2-d transform and quantization architectures for h.264 video coders," in *IEEE Int. Symp. on Circuits and Syst. (ISCAS 2005)*, vol. 2, May 2005, pp. 1802–1805.

[14] G. Pastuszak, "A high-performance architecture of the double-mode binary coder for h.264.avc," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 7, pp. 949–960, Jul. 2008.

[15] J. Park, K. Muhammad, and K. Roy, "High-performance fir filter design based on sharing multiplication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 2, pp. 244–253, Apr. 2003.

[16] K.-S. Chong, B.-H. Gwee, and J. S. Chang, "A 16-channel low-power nonuniform spaced filter bank core for digital hearing aids," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 9, pp. 853–857, Sep. 2006.

[17] B. Paul, S. Fujita, and M. Okajima, "Rom-based logic (rbl) design: A low-power 16 bit multiplier," *IEEE J. Solid-State Circuits*, vol. 44, no. 11, pp. 2935–2942, Nov. 2009.

[18] M. D. Ercegovac and T. Lang, "Multiplication," in *Digital Arithmetic*. San Francisco: Morgan Kaufmann, 2004, pp. 181–245.

[19] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4^{th} ed. USA: Addison-Wesley Publishing Company, 2010.

[20] "Dual dsp plus micro for audio applications," Feb. 2003, TDA7503 Datasheet, STMicroelectronics.

[21] C. Xu, X. Dong, N. Jouppi, and Y. Xie, "Design implications of memristor-based rram cross-point structures," in *Design, Automation Test in Europe Conf. Exhibition (DATE)*, Mar. 2011, pp. 1–6.