Simplified Logic for Tree-Structure Segmented DEM Encoders

Christian Venerus, *Member, IEEE*, Jason Remple, and Ian Galton, *Fellow, IEEE*

***Abstract*—Segmented dynamic element matching (DEM) encoders can be used in high-resolution digital-to-analog converters (DACs) to reduce nonlinear distortion that would otherwise arise from component mismatches while avoiding the high circuit complexity of non-segmented DEM encoders. This paper presents tree-structure segmented DEM encoders that have several advantages over prior segmented DEM encoders: they do not require digital adders, they have latencies that are linear rather than parabolic functions of the number of bits of DAC resolution, and they have lower hardware complexity than previously published architectures.**

***Index Terms*—Analog-to-digital conversion, digital-to-analog conversion, dynamic element matching (DEM), encoder, segmentation, tree-structure.**

I. Introduction

A DYNAMIC element matching (DEM) digital-to-analog converter (DAC) consists of an all-digital block called a *DEM encoder*, followed by an array of N 1-bit DACs whose outputs are added together to form the DEM DAC's continuous-time output signal $y(t)$. The DEM encoder scrambles the usage of the 1-bit DACs such that the error caused by component mismatches has a noise-like structure, is free of nonlinear distortion and spurious tones, and has either a white or shaped power spectral density [1]–[3].

In unity-weighted DEM DACs, all the 1-bit DACs are nominally identical. Unfortunately, in such DEM DACs, the number of 1-bit DACs, and therefore the hardware complexity, increases exponentially with the number of bits of DAC resolution, B.

Segmented DEM DACs can be used in place of unity-weighted DEM DACs in high-resolution applications to reduce circuit complexity at the expense of 1-bit DAC array power consumption [4]–[7]. A segmented DEM DAC consists of an all-digital DEM encoder, called a segmented DEM encoder, followed by an array of N 1-bit DACs, some of which have different nominal step-sizes. For greater than about 5 bits of resolution, they offer reduced complexity compared to unity-weighted DEM DACs [7]. Nevertheless, as shown in this paper, the complexity of previously published segmented DEM encoders retains a square or exponential dependence on B and a propagation delay proportional to B^2.

This paper presents adder-free, reduced-hardware-complexity segmented DEM encoders whose latencies are linear functions of the number of bits used to represent the DEM encoders' input sequences. Moreover, the proposed DEM encoders consist of digital sub-blocks, called *segmenting* and *non-segmenting switching blocks*, whose implementations are independent of the DEM DACs' resolutions and 1-bit DAC step-sizes, thereby simplifying the design process.

Manuscript received November 25, 2015; revised March 8, 2016; accepted March 23, 2016. Date of publication April 1, 2016; date of current version October 27, 2016. This work was supported in part by the National Science Foundation under Award 1343389 and in part by the corporate members of the Center for Wireless Communications, University of California at San Diego. This paper was recommended by Associate Editor B. Sahoo.

C. Venerus is with Qualcomm Technologies, Inc., San Diego, CA 92121 USA.

J. Remple is with Microchip Technology Inc., San Diego, CA 92121 USA.

I. Galton is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: galton@ece.ucsd.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2016.2549618

II. DEM DACs' Overview

Each sample of the f_s-rate DEM DAC's input sequence, $x[n]$, for $n = 0, 1, 2, \ldots$, is interpreted as a numerical value in the set

$$\left\{-\frac{M}{2}\Delta,\ -\left(\frac{M}{2}-1\right)\Delta,\ -\left(\frac{M}{2}-2\right)\Delta, \ldots,\ \frac{M}{2}\Delta\right\} \tag{1}$$

where $M+1$ is the number of DAC input sequence levels and Δ is the minimum *step-size* of $x[n]$.

Let $c_i[n]$ be the two-level (0 or 1) input sequence to the ith 1-bit DAC, and let $\lfloor w \rfloor$ be the largest integer not greater than w for any real number w. The output of the ith 1-bit DAC is

$$y_i(t) = x_i\left[\lfloor f_s t \rfloor\right]\Delta_i + e_i(t) \tag{2}$$

where

$$x_i[n] = c_i[n] - \frac{1}{2} \tag{3}$$

$\Delta_i = K_i\Delta$, K_i is the *weight* of the 1-bit DAC, and $e_i(t)$ is called *mismatch error* because it represents the 1-bit DAC's pulse shape, timing, and amplitude errors caused by non-ideal circuit behavior. The 1-bit DAC weights are assigned such that $K_1 = 1, K_i \geq K_{i-1}$ for $i = 2, 3, \ldots, N$, and

$$M = \sum_{i=1}^{N} K_i. \tag{4}$$

In the return-to-zero 1-bit DACs normally used in DEM DACs, $e_i(t)$ is well modeled as

$$e_i(t) = \begin{cases} e_{1i}(t), & \text{if } c_i\left[\lfloor f_s t \rfloor\right] = 1 \\ e_{0i}(t), & \text{if } c_i\left[\lfloor f_s t \rfloor\right] = 0 \end{cases} \tag{5}$$

where $e_{1i}(t)$ and $e_{0i}(t)$ are periodic error functions of period $T_s = 1/f_s$ that represent the error introduced by the ith 1-bit

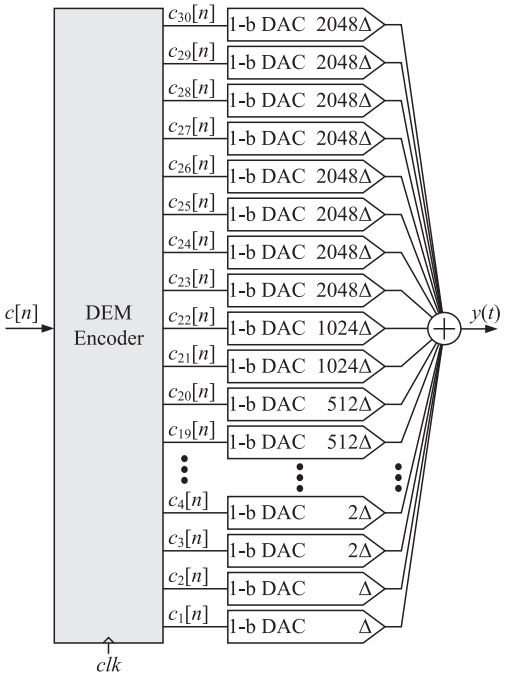

Fig. 1. Example of a DEM DAC for $N = 30$. The implementation details of the all-digital DEM encoder are shown in Fig. 2.

DAC over each clock interval for each of the two possible values of $c_i[n]$.

The DEM encoder sets each of its 1-bit output sequences to 0 or 1 such that

$$x[n] = \sum_{i=1}^{N} x_i[n]\Delta_i \tag{6}$$

which ensures that, in the absence of mismatch error, the DAC output during the nth sample interval is

$$y(t) = x\left[\lfloor f_s t \rfloor\right]. \tag{7}$$

When implementing the DEM encoder, it is often convenient to map the sequence of DEM DAC input codewords, $x[n]$, to a sequence of non-negative integers, $c[n]$, related to $x[n]$ by

$$c[n] = \frac{x[n]}{\Delta} + c_{\mathrm{OS}} \tag{8}$$

where c_{OS} is a constant that is greater than or equal to $M/2$ [7]. Therefore, without loss of generality, $c[n]$ can be considered to be the DEM encoder's input sequence.

An example of a 14-bit DEM DAC implementation for $N = 30$, $M = 20\,478$, and 1-bit DAC weights

$$\begin{aligned} K_{2i-1} = K_{2i} &= 2^{i-1} && \text{for } i = 1, \ldots, 11 \\ K_i &= 2048 && \text{for } i = 23, \ldots, 30 \end{aligned} \tag{9}$$

is shown in Fig. 1. The DEM encoder consists of 29 digital *switching blocks*, labeled $S_{k,r}$ for $k = 1, 2, \ldots, 14$, and $r = 1, 2, \ldots, 15$, configured in a tree structure as shown in Fig. 2 [7]. The 11 switching blocks that are shaded in the figure are called *segmenting switching blocks*, and the other 18 switching blocks are called *non-segmenting switching blocks*.

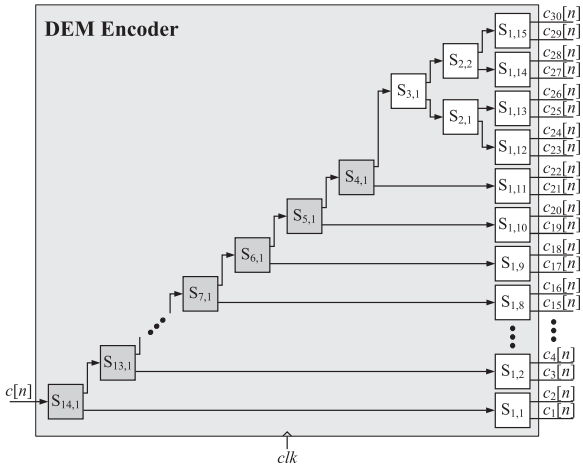

Fig. 2. DEM encoder in the DEM DAC of Fig. 1.

The top and bottom outputs of each segmenting switching block, $S_{k,1}$, are

$$\frac{1}{2}\left(c_{k,1}[n] - 1 - s_{k,1}[n]\right) \quad \text{and} \quad 1 + s_{k,1}[n] \tag{10}$$

respectively, where $c_{k,1}[n]$ is the switching block input sequence, and $s_{k,1}[n]$, called a *switching sequence*, is 0 when $c_{k,1}[n]$ is odd and 1 or -1 otherwise. Similarly, the top and bottom outputs of each non-segmenting switching block $S_{k,r}$ are

$$\frac{1}{2}\left(c_{k,r}[n] - s_{k,r}[n]\right) \quad \text{and} \quad \frac{1}{2}\left(c_{k,r}[n] + s_{k,r}[n]\right) \tag{11}$$

respectively, where $c_{k,r}[n]$ is the switching block input sequence, and the switching sequence $s_{k,r}[n]$ in this case is 0 when $c_{k,r}[n]$ is even and 1 or -1 otherwise.

As shown in [7] the DEM DAC's output has the form

$$y(t) = \alpha(t)x\left[\lfloor f_s t \rfloor\right] + \beta(t) + e_{\mathrm{DAC}}(t) \tag{12}$$

where $\alpha(t)$ and $\beta(t)$ are the linear combinations of $e_{1i}(t)$ and $e_{0i}(t)$, hence, they are T_s-periodic functions, and $e_{\mathrm{DAC}}(t)$, called *DAC noise*, is a function of the 1-bit DAC mismatch errors and the switching sequences. Although the 1-bit DAC mismatch errors cause $\alpha(t)$ and $\beta(t)$ to deviate from their ideal values of 1 and 0, respectively, these deviations do not introduce nonlinear distortion, thus they are tolerable in most applications.

Provided that the switching sequences are noise-like sequences that are zero mean, free of spurious tones, and uncorrelated with $x[n]$, $e_{\mathrm{DAC}}(t)$ has a noise-like structure that is free of spurious tones, and the DEM DAC does not introduce nonlinear distortion [7].

III. Simplified Logic for Tree-Structure Segmented DEM Encoders

The hardware complexity of a given tree-structure DEM encoder depends on its numbers of segmenting and non-segmenting switching blocks, and the switching blocks'

hardware complexities. It generally increases with the number of unity-weighted 1-bit DACs [7]. Hence, bounds on the hardware complexity of DEM encoders are derived below considering the two complexity extremes: unity-weighted DEM DACs in which $N = 2^B$ and all 1-bit DACs have unity weight, i.e., $K_N = 1$, and fully-segmented DEM DACs in which $N = 2B$, $K_{2i} = K_{2i-1} = 2^{i-1}$ for $i = 1, 2, \ldots, N/2$ [7].

Switching blocks directly connected to 1-bit DACs, e.g., the $S_{1,r}$ switching blocks in Fig. 2, have 1-bit outputs by design. It follows from (10) and (11) that each switching block input sequence requires one more bit than at least one of the corresponding switching block output sequences when represented as an unsigned binary word. According to (1), (4), and (8), $c[n]$ requires $B + 1$ bits if represented as an unsigned binary word for both unity-weighted and fully-segmented DEM DACs.

It can be thus verified by traversing a tree-structure DEM encoder from its outputs to its input that each switching block in the kth *layer*, i.e., each $S_{k,r}$ switching block in Fig. 2, has a $k + 1$-bit input sequence. Moreover, any tree-structure DEM encoder's critical timing path consecutively traverses B switching blocks, one in each layer k. The encoder's latency can thus be shown to be proportional to $B^2 + 3B$ if ripple carry adders are used, and at least to

$$1 + \sum_{k=1}^{B} \log_2(k+1) = 1 + \log_2\left[(B+1)!\right] \tag{13}$$

for faster but more complex adder architectures such as prefix adders [9].[1]

As each $S_{k,r}$ switching block has a $k + 1$-bit input sequence, the number of digital gates required for it to implement (10) and (11) using binary adders is at least proportional to $k + 1$ [8], [9]. In a unity-weighted DEM DAC, the DEM encoder consists of only non-segmenting switching blocks arranged in a binary tree [3]. At each layer k there are 2^{B-k} switching blocks so the number of digital gates in the DEM encoder is proportional to

$$\sum_{k=1}^{B} (k+1)2^{B-k} = 3 \cdot 2^B - (B+3). \tag{14}$$

In a fully-segmented DEM DAC, the DEM encoder consists only of one segmenting switching block per layer up to layer $k = 2$ and B non-segmenting switching blocks at layer $k = 1$. The number of digital gates necessary to implement the DEM encoder is proportional to

$$2 \cdot B + \sum_{k=2}^{B} (k+1) = \frac{B^2 + 7B - 4}{2}. \tag{15}$$

In the next sections, an encoding scheme for the switching blocks' input and output signals is presented that removes the need for adders, thereby reducing the DEM encoder hardware complexity and latency. The proposed DEM encoder is described in the context of the DEM DAC example of Fig. 2 to simplify the explanation.

[1]These latency expressions represent upper bounds in that they were derived by adding the latencies of all the switching blocks; hence, their derivations implicitly assume that all the input bits of each switching block are valid before that switching block begins its calculations.

A. Adder-Free Segmenting Switching Blocks

The bottom output of each segmenting switching block, $S_{k,1}$, for $k = 4, \ldots, 14$ is a three-level sequence $c_{1,15-k}[n]$ restricted to the set of values {0, 1, 2} according to (10). It can be represented by a 2-bit binary number interpreted according to the *extra-LSB encoding* scheme in [10]: each of the two bits, $c^{(0)}_{1,15-k}[n]$ and $c^{(1)}_{1,15-k}[n]$, can have a value of one or zero, and the numerical value associated with $c_{1,15-k}[n]$ is interpreted as

$$c_{1,15-k}[n] = 1 + s_{k,1}[n] = c^{(0)}_{1,15-k}[n] + c^{(1)}_{1,15-k}[n]. \tag{16}$$

Whenever $c_{k,1}[n]$ is even, $c_{1,15-k}[n]$ is either 0 or 2, depending on the value of $s_{k,1}[n]$. If $c_{1,15-k}[n] = 0$, $c^{(0)}_{1,15-k}[n]$ and $c^{(1)}_{1,15-k}[n]$ are both 0. If $c_{1,15-k}[n] = 2$, $c^{(0)}_{1,15-k}[n]$, and $c^{(1)}_{1,15-k}[n]$ are both set to 1. Whenever $c_{k,1}[n]$ is odd, $c_{1,15-k}[n]$ is 1 and the bottom output bits $c^{(0)}_{1,15-k}[n]$ and $c^{(1)}_{1,15-k}[n]$ are set to 1 and 0, respectively. Hence, the signal processing performed by $S_{k,1}$ segmenting switching blocks to generate their bottom outputs does not require explicit adders and is performed simply by setting high or low the LSBs of $c_{1,15-k}[n]$ depending on the parity of $c_{k,1}[n]$ and the value of $s_{k,1}[n]$.

Unfortunately, this scheme is not applicable to the inputs and top outputs of the segmenting switching blocks, i.e., to the $c_{k,1}[n]$ and $c_{k-1,1}[n]$ sequences, where $k = 4, \ldots, 14$. Instead, a different encoding convention, called *negative-extra-LSB encoding*, has been devised and used for these sequences. The negative-extra-LSB encoding of $c_{k,1}[n]$ consists of $k + 1$ bits that are denoted $c^{(i)}_{k,1}[n]$ $(i = 1, \ldots, k)$ and $c^{(-)}_{k,1}[n]$, each of which takes on a value of one or zero. The numerical value of $c_{k,1}[n]$ is interpreted as

$$c_{k,1}[n] = \sum_{i=1}^{k} 2^{i-1} c^{(i)}_{k,1}[n] - c^{(-)}_{k,1}[n], \tag{17}$$

so that the extra LSB, $c^{(-)}_{k,1}[n]$, has an effective weight of -1.

A conventional unsigned binary encoded number can be converted to a negative-extra-LSB encoded number by appending an extra zeroth bit and setting it low.

Whenever $c_{k,1}[n]$ is even, $c^{(1)}_{k,1}[n]$ and $c^{(-)}_{k,1}[n]$ are either both zero or both one. If $s_{k,1}[n] = -1$, the top output, $c_{k-1,1}[n]$, of the $S_{k,1}$ switching block is

$$\frac{1}{2}\left(c_{k,1}[n] - 1 - s_{k,1}[n]\right) = \frac{c_{k,1}[n]}{2}, \tag{18}$$

which is generated by right shifting by 1 bit the $k - 1$ MSBs of $c_{k,1}[n]$ and setting $c^{(-)}_{k-1,1}[n] = 0$. If instead $s_{k,1}[n] = 1$, $c_{k-1,1}[n]$ is generated by right shifting by 1 bit the $k - 1$ MSBs of $c_{k,1}[n]$ and setting $c^{(-)}_{k-1,1}[n] = 1$ so that its numerical value is

$$\frac{1}{2}\left(c_{k,1}[n] - 1 - s_{k,1}[n]\right) = \frac{c_{k,1}[n] - 2}{2} = \frac{c_{k,1}[n]}{2} - 1. \tag{19}$$

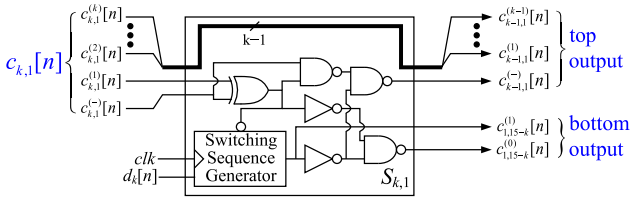

Fig. 3. RTL implementation for adder-free segmenting switching blocks.

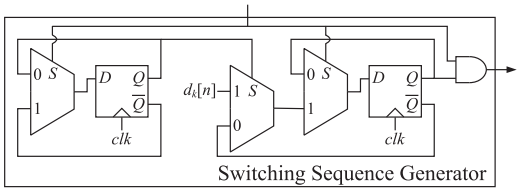

Fig. 4. RTL implementation of the switching sequence generator in the adder-free switching blocks of Figs. 3 and 5.

For any real number w, let the fractional part of w be defined as $\{w\} = w - \lfloor w \rfloor$. Whenever $c_{k,1}[n]$ is odd, $s_{k,1}[n]=0$ and the top output of the $S_{k,1}$ switching block is set to

$$\frac{c_{k,1}[n]-1}{2} = \frac{c_{k,1}[n]}{2} - \left\{\frac{c_{k,1}[n]}{2}\right\} = \left\lfloor \frac{c_{k,1}[n]}{2} \right\rfloor \tag{20}$$

by right shifting by 1 bit the $k-1$ MSBs of $c_{k,1}[n]$ and setting $c^{(-)}_{k-1,1}[n] = 0$.

Therefore, the negative-extra-LSB encoding enables each $S_{k,1}$ segmenting switching block to perform the additions and subtractions in (10) without affecting the $k-1$ MSBs of $c_{k,1}[n]$, which can be simply routed to the next switching block.

The complete register-transfer level (RTL) view for adder-free segmenting switching blocks is shown in Fig. 3, wherein the switching sequence generator is implemented as shown in Fig. 4 and sequences $d_k[n]$ for $k=4,5,\ldots,14$ well approximate white random processes that each take on values of 0 and 1 with equal probability and are independent of each other and $x[n]$.

It can be verified from (16) through (19) that the digital logic in Figs. 3 and 4 sets $s_{k,1}[n]$ in (10) to be -1 or 1 depending on whether $d_k[n]$ is 0 or 1, respectively, every other time $c_{k,1}[n]$ is even. Each subsequent time $c_{k,1}[n]$ is even, it sets $s_{k,1}[n]$ to the negative of what $s_{k,1}[n]$ was the previous time $c_{k,1}[n]$ was even. Whenever $c_{k,1}[n]$ is odd, $s_{k,1}[n]$ is instead set to 0.

As only the parity of $c_{k,1}[n]$ and $d_k[n]$ are used to determine both outputs of $S_{k,1}$ adder-free segmenting switching blocks, only the two LSBs of $c_{k,1}[n]$ and the 1-bit $d_k[n]$ sequence are involved in the computation of each switching block output sequences. Hence each adder-free segmenting switching block can be implemented using the logic shown in Fig. 3, with identical hardware complexity and latency, regardless of the number of bits of its input sequence, $c_{k,1}[n]$, and layer k to which it belongs.

B. Adder-Free Non-Segmenting Switching Blocks

An adder-free implementation of non-segmenting switching blocks $S_{k,r}$ for $k=2,3$, $r=1,2$, and $k=1$, $r=1,\ldots,15$, was presented in [10] and is shown with notation compliant to [7] in Fig. 5. The switching sequence generator in Fig. 5 is

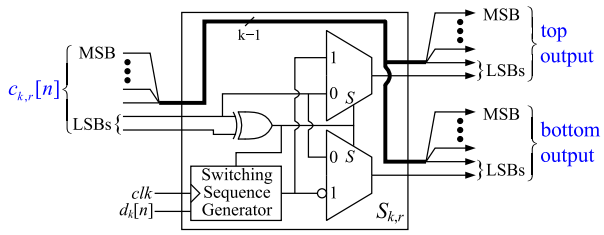

Fig. 5. RTL implementation for adder-free non-segmenting switching blocks.

identical to that used in adder-free segmenting switching blocks and shown in Fig. 4. Sequences $d_k[n]$ for $k=1,2,3$ well approximate white random processes that each take on values of 0 and 1 with equal probability, and are independent of each other, $x[n]$, and sequences $d_k[n]$ for $k=4,5,\ldots,14$.

The extra-LSB encoding implemented for the bottom outputs of segmenting switching blocks $S_{k,1}$, for $k=4,\ldots,14$, is used to represent the input and output sequences of each non-segmenting switching block. The extra-LSB encoding [10] of the $c_{k,r}[n]$ sequence consists of $k+1$ bits denoted $c^{(i)}_{k,r}[n]$, for $i=0,\ldots,k$. Each bit can have a value of one or zero, and the numerical value associated with $c_{k,r}[n]$ is

$$c_{k,r}[n] = \sum_{i=1}^{k} 2^{i-1} c^{(i)}_{k,r}[n] + c^{(0)}_{k,r}[n]. \tag{21}$$

From (11), each non-segmenting switching block adds or subtracts 1 from its input only when its input is odd. For odd inputs, only one between $c^{(0)}_{k,r}[n]$ and $c^{(1)}_{k,r}[n]$ can be set at 1. Hence, the extra-LSB encoding allows each $S_{k,r}$ switching block to perform the additions and subtractions in (11) by setting high or low both LSBs of its output signals without affecting the $k-1$ MSBs of $c_{k,r}[n]$, which are routed to the next non-segmenting switching blocks as shown in Fig. 5.

C. Hardware-Efficient DEM Encoders

A hardware-efficient version of the DEM encoder in Fig. 2 that avoids conventional adders can be implemented using the switching blocks shown in Figs. 3 and 5, and mapping the negative-extra-LSB representation of $c_{3,1}[n]$ as output of $S_{4,1}$ to its extra-LSB representation as input to $S_{3,1}$. The DEM DAC input sequence $x[n]$ is also to be mapped to the sequence $c[n]$ according to (8).

A necessary condition for any DEM encoder to ensure that the DAC noise has a noise-like structure regardless of the 1-bit DAC error pulses is that $x[n]$ be restricted to avoid the smallest K_N-2 values and the largest K_N-2 values of (1) [7]. In the DEM DAC example of Fig. 1, the range of values $c[n]$ may take on according to (8) is thus restricted between $K_N-1=2047$ and $M-(K_N-1)=18\,431$.

Further reducing the interval of values for $c[n]$ by one unit imparts a very modest reduction in its total range. However, it ensures that $x[n]/\Delta$ can be represented by a $B=14$-bit two's complement binary word, as typically done in practical implementations, and mapped to $c[n]$ following (8) for $c_{OS} = 2^{13}+2048$.

The offset c_{OS} is added to the DEM DAC input sequence in two steps. First, it can be verified by recursively applying (10)

for $k = 4, 5, \ldots, 14$ that the addition of a 2048 offset to the nth DEM encoder input sample does not affect the operation of switching blocks $S_{k,1}$ for $k = 4, 5, \ldots, 14$. The offset, however, does affect the operation of $S_{3,1}$ by adding a one to its input sequence $c_{3,1}[n]$.

As the top output of $S_{4,1}$, $c_{3,1}[n]$ is represented by design in negative-extra-LSB notation so that

$$c_{3,1}[n] + 1 = \sum_{i=1}^{3} 2^{i-1} c_{3,1}^{(i)}[n] - c_{3,1}^{(-)}[n] + 1. \tag{22}$$

As an input to $S_{3,1}$, $c_{3,1}[n] + 1$ is represented in extra-LSB notation as

$$c_{3,1}[n] + 1 = \sum_{i=1}^{3} 2^{i-1} c_{3,1}^{(i)}[n] + c_{3,1}^{(0)}[n] \tag{23}$$

according to (21). An offset of 2048 is added to the DEM encoder input sequence, and the top output of $S_{4,1}$ is assigned as an input to $S_{3,1}$ by setting

$$c_{3,1}^{(0)}[n] = 1 - c_{3,1}^{(-)}[n] \tag{24}$$

so that (22) equates (23). The three MSBs of $c_{3,1}[n]$ as output of $S_{4,1}$ are routed to their corresponding bits at the input of $S_{3,1}$, whereas from (24), $c_{3,1}^{(0)}[n]$ is the inverted version of $c_{3,1}^{(-)}[n]$.

Finally, $x[n]/\Delta$ is typically represented by a 14-bit two's complement binary number as mentioned. Its bits are denoted as $x^{(i)}[n]$ $(i = 1, \ldots, 14)$, and its numerical value is interpreted to be

$$\frac{x[n]}{\Delta} = \sum_{i=1}^{13} 2^{i-1} x^{(i)}[n] - 2^{13} x^{(14)}[n]. \tag{25}$$

The remaining offset of 2^{13} in (8) is added to $x[n]/\Delta$ by simply inverting the MSB of $x[n]/\Delta$, $x^{(14)}[n]$, in (25).

The sequence of operations outlined maps the DEM DAC input sequence $x[n]$ into the sequence $c[n] = x[n]/\Delta + 2^{13} + 2048$, which takes on values within the range of $\{2048, \ldots, 18\,431\}$. The mapping is performed without requiring any extra hardware except for two inverters. The complete RTL view of the hardware-efficient DEM encoder is shown in Fig. 6, where all segmenting and non-segmenting switching blocks are implemented as in Figs. 3 and 5, respectively.

The results presented in [7] and [10] imply that the $s_{k,r}[n]$ sequences for $k = 1, 2, \ldots, 14$ and $r = 1, 2, \ldots, 15$ are zero-mean sequences that are free of spurious tones, have a highpass spectral shape with a zero-frequency null, and are uncorrelated with each other and $x[n]$. The results in [7] can be used to prove that $e_{\mathrm{DAC}}(t)$ also has these properties and therefore the DEM DAC does not introduce nonlinear distortion.

By representing the input and output signals of each switching block according to (17) and (21), the DEM encoder in Fig. 6 does not require explicit adders, reducing the power consumption, required area, and propagation delay of both segmenting and non-segmenting switching blocks. For example, a conventional ripple–carry adder implementation of the DEM encoder in Fig. 2 would require 1303 digital gates, whereas the DEM encoder in Fig. 6 consists of 325 digital gates.

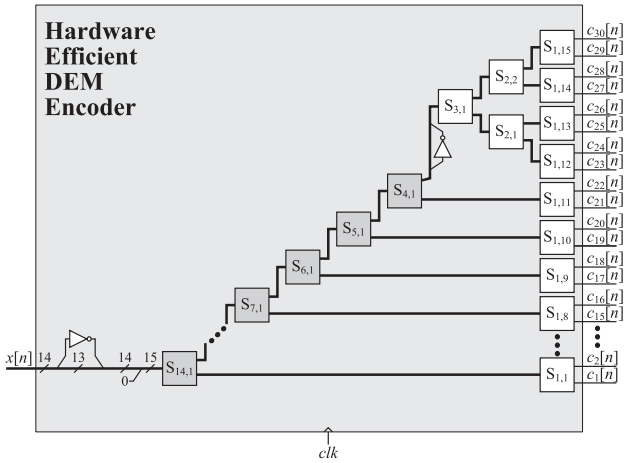

Fig. 6. Implementation of the DEM encoder of Fig. 2 without the use of any adder. Segmenting and non-segmenting switching blocks are implemented, as shown in Figs. 3–5.

As all switching blocks have similar hardware complexity, the number of digital gates that make up the proposed DEM encoder is, in general, simply proportional to the number of switching blocks in the encoder itself

$$\sum_{k=1}^{B} 2^{B-k} = 2^B - 1, \tag{26}$$

and $2B - 1$, for unity-weighted and fully-segmented DEM DACs, respectively.

As explained at the beginning of Section III, the DEM encoder's critical timing path traverses B switching blocks, and the number of digital gates in each switching block is independent of the layer k to which the switching block belongs. Hence, the latency of the proposed DEM encoder is a linear function of the number of bits, B, of DAC resolution.

REFERENCES

[1] B. H. Leung and S. Sutarja, "Multi-bit sigma–delta A/D converter incorporating a novel class of dynamic element matching techniques," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, no. 1, pp. 35–51, Jan. 1992.

[2] R. W. Adams and T. W. Kwan, "Data-Directed Scrambler for Multi-Bit Noise Shaping D/A Converters," U.S. Patent 5 404 142, Apr. 4, 1995.

[3] I. Galton, "Spectral shaping of circuit errors in digital-to-analog converters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, no. 10, pp. 808–817, Nov. 1997.

[4] R. Adams and K. Q. Nguyen, "A 113-dB SNR oversampling DAC with segmented noise-shaped scrambling," *IEEE J. Solid-State Circuits*, vol. 33, no. 12, pp. 1871–1878, Dec. 1998.

[5] B. Jewett, J. Liu, and K. Poulton, "A 1.2 GS/s 15b DAC for precision signal generation," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2005, 587, pp. 110–111.

[6] K. L. Chan, J. Zhu, and I. Galton, "Dynamic element matching to prevent nonlinear distortion from pulse-shape mismatches in high-resolution DACs," *IEEE J. Solid-State Circuits*, vol. 43, no. 9, pp. 2067–2078, Sep. 2008.

[7] K. L. Chan, N. Rakuljic, and I. Galton, "Segmented dynamic element matching for high-resolution digital-to-analog conversion," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3383–3392, Dec. 2008.

[8] R. B. Johnson, "The complexity of a VLSI adder," *Inf. Process. Lett.*, vol. 11, no. 2, pp. 92–93, Oct. 1980.

[9] S. Winograd, "On the time required to perform addition," *J. Assoc. Comput. Mach.*, vol. 12, no. 2, pp. 277–285, Apr. 1965.

[10] J. Welz, I. Galton, and E. Fogleman, "Simplified logic for first-order and second-order mismatch-shaping digital-to-analog converters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 11, pp. 1014–1027, Nov. 2001.