Tree-Structured DEM DACs with Arbitrary Numbers of Levels

Nevena Rakuljic, *Member, IEEE*, and Ian Galton, *Senior Member, IEEE*

***Abstract*—Unity-weighted tree-structured dynamic element matching (DEM) DACs are widely used in delta-sigma ($\Delta\Sigma$) data converters to ensure that mismatches among nominally identical analog components give rise to shaped noise instead of nonlinear distortion. Tree-structured DEM DACs offer an advantage over other published DEM DACs in that the shaped noise from component mismatches can be made free of spurious tones. However, previously published unity-weighted tree-structured DEM DACs have the disadvantage that they require a power-of-two number of nominally identical 1-bit DACs. When applied to a $\Delta\Sigma$ data converter with a non-power-of-two number of quantization steps, this requires the DEM DAC to have a larger input range than needed by the $\Delta\Sigma$ data converter which wastes power and circuit area. This paper presents a generalized tree-structured DEM encoder applicable to DEM DACs with any number of 1-bit DACs, thereby avoiding this limitation.**

***Index Terms*—Digital-to-analog conversion, dynamic element matching (DEM), encoder.**

I. Introduction

A TYPICAL unity-weighted dynamic element matching (DEM) digital-to-analog converter (DAC) consists of a DEM encoder followed by a bank of N nominally identical 1-bit DACs, the outputs of which are summed to form the output of the DEM DAC. The DEM encoder maps the input sequence into N 1-bit sequences, each of which drives one of the 1-bit DACs. As described in the next section, the DEM encoder exploits flexibility in its choice of output bits each sample period to cause the error arising from mismatches among the 1-bit DACs to have a noise-like structure that is free of nonlinear distortion and spectrally shaped as appropriate for the application.

Unity-weighted DEM DACs are widely used in oversampling $\Delta\Sigma$ data converters, i.e., $\Delta\Sigma$ ADCs and $\Delta\Sigma$ DACs, to prevent component mismatches from degrading data converter precision [1]–[30]. The DACs within a typical $\Delta\Sigma$ data converter need only convert digital signals with a small number of levels, but they must not add significant error within the $\Delta\Sigma$ data converter's relatively narrow signal band. Therefore, DEM is used to spectrally shape the error introduced by the DACs so as to suppress the error within the signal band.

Manuscript received January 13, 2009. First published June 02, 2009; current version published February 10, 2010. This work was supported by the National Science Foundation under Award 0515286, by the UCSD Center for Wireless Communications and by the University of California Discovery Program. This paper was recommended by Associate Editor G. Manganaro.

The authors are with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA (e-mail: galton@ece.ucsd.edu).

Digital Object Identifier 10.1109/TCSI.2009.2023931

Many types of unity-weighted DEM encoders have been published to date, one of which is the tree-structured DEM encoder [5], [9]–[12], [31], [32]. To the knowledge of the authors the tree-structured DEM encoder is the only of these in which the error caused by 1-bit DAC mismatches has been made spectrally shaped and free of spurious tones. This is a significant advantage in high-performance $\Delta\Sigma$ data converters, which tend to be used in applications which are highly sensitive to spurious tones. However, previously published tree structured DEM DACs have the disadvantage that they require a power-of-two number of 1-bit DACs. When applied to a $\Delta\Sigma$ data converter with a non-power-of-two number of quantization steps, this requires the DEM DAC to have a larger input range than needed by the $\Delta\Sigma$ data converter which wastes power and circuit area. This paper presents a generalized tree-structured DEM encoder applicable to DEM DACs with any number of 1-bit DACs, thereby avoiding this limitation.

II. Unity-Weighted DEM DAC Overview

The purpose of a DAC is to convert a sequence of input values, $x[n]$, $n = 0, 1, 2, \ldots$, represented as a sequence of digital codewords updated at times nT, where T is the duration of each sample period, into an analog waveform. In this paper, for a DAC with $N+1$ levels each codeword is interpreted by design convention to have a numerical value in the range

$$\left\{-\frac{N}{2}\Delta,\ -\left(\frac{N}{2}-1\right)\Delta,\ -\left(\frac{N}{2}-2\right)\Delta, \ldots,\ \frac{N}{2}\Delta\right\} \quad (1)$$

where Δ is the minimum *step-size* of $x[n]$. Ideally, the output of the DAC during the nth sample period, i.e. during the time interval $nT \le t < (n+1)T$, is an analog pulse given by

$$y(t) = a(t - nT)x[n] \quad (2)$$

where $a(t)$ is called the *unit output pulse* and is zero outside of $0 \le t < T$.

A general architecture for such a DAC is shown in Fig. 1. It consists of an all-digital *encoder* followed by a bank of N 1-bit DACs, the outputs of which are summed to form the DAC output waveform, $y(t)$. The encoder maps the sequence of input codewords into N 1-bit sequences denoted as $c_i[n]$, $i = 1, 2, \ldots, N$, each of which takes on a value of 0 or 1 for each n. The encoder chooses its N output bits once per sample period under the constraint

$$x[n] = \Delta \sum_{i=1}^{N} \left(c_i[n] - \frac{1}{2}\right). \quad (3)$$

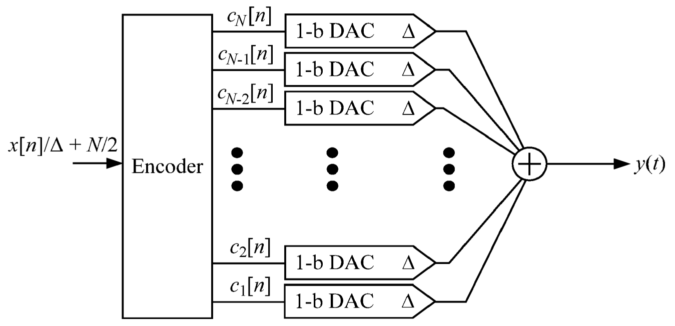

Fig. 1. General DAC Architecture.

The output of the ith 1-bit DAC during the nth sample period is given by

$$y_i(t) = \begin{cases} \frac{\Delta}{2}a(t-nT) + e_{hi}(t-nT) & \text{if } c_i[n]=1 \\ -\frac{\Delta}{2}a(t-nT) + e_{li}(t-nT) & \text{if } c_i[n]=0 \end{cases} \tag{4}$$

where $e_{hi}(t)$ and $e_{li}(t)$ are *mismatch error pulses* that result from inadvertent process variations during IC fabrication. The only assumption made about $e_{hi}(t)$ and $e_{li}(t)$ in this paper is that they are zero outside of $0 \le t < T$. The output of the overall DAC is given by

$$y(t) = \sum_{i=1}^{N} y_i(t). \tag{5}$$

The output of the ith 1-bit DAC during the nth sample period as given by (4) can equivalently be written as

$$y_i(t) = \alpha_i(t-nT)\left(c_i[n] - \frac{1}{2}\right)\Delta + \beta_i(t-nT) \tag{6}$$

where

$$\alpha_i(t) = a(t) + \frac{e_{hi}(t) - e_{li}(t)}{\Delta} \quad \text{and} \quad \beta_i(t) = \frac{e_{hi}(t) + e_{li}(t)}{2}. \tag{7}$$

This can be verified by substituting (7) into (6) to obtain (4).

Substituting (6) and (7) into (5) yields

$$y(t) = a(t-nT)x[n] + \beta(t-nT) + \varepsilon(t) \tag{8}$$

during the nth sample period, where

$$\beta(t) = \sum_{i=1}^{N} \beta_i(t) \tag{9}$$

and

$$\varepsilon(t) = \sum_{i=1}^{N}\left(c_i[n] - \frac{1}{2}\right)\left(e_{hi}(t-nT) - e_{li}(t-nT)\right). \tag{10}$$

The mismatch error pulses are responsible for the terms $\beta(t-nT)$ and $\varepsilon(t)$ in (8). The first of these terms is a fixed pulse that repeats each sample period, independent of $x[n]$. Consequently, it results only in spurious tones at multiples of the sample frequency which do not degrade the signal-to-noise ratio (SNR) or the in-band spurious-free-dynamic-range (SFDR) of the DAC. In sampled DACs such as those implemented with switched capacitor circuits, it aliases down to a fixed offset, in which case it does not introduce any tones. In contrast, the $\varepsilon(t)$ term represents the dynamic error caused by the mismatch error pulses so it has the potential to degrade both the SNR and SFDR of the DAC.

Unfortunately, short of eliminating the mismatch error pulses, it is not possible to make $\varepsilon(t)$ zero. However, the encoder does have some control over the structure of $\varepsilon(t)$ because for each DAC input value except $x[n] = -N\Delta/2$ and $x[n] = N\Delta/2$ the encoder can choose among multiple sets of bits $\{c_1[n], c_2[n], \ldots, c_N[n]\}$ that satisfy (3). For example, for any value of n at which $x[n] = -(N/2-1)\Delta$ the encoder can satisfy (3) by setting $c_i[n] = 1$ for any single value of i in the set $\{1, 2, \ldots, N\}$ and setting $c_j[n] = 0$ for all $j \neq i$. Encoders that dynamically exploit this flexibility to impart desirable properties to $\varepsilon(t)$ are called DEM encoders.

Since $\varepsilon(t)$ cannot be eliminated, its presence would be most tolerable if a DEM encoder could cause it to be a random process that is uncorrelated with $x[n]$, free of spurious tones, and spectrally shaped as appropriate for the application at all times regardless of $x[n]$ and the mismatch error pulses. A necessary condition for $\varepsilon(t)$ to have these properties is that its expectation during each sample period must be independent of $x[n]$. If this necessary condition were not satisfied, the expectation of $\varepsilon(t)$ would be a deterministic function of $x[n]$.

Unfortunately, $\varepsilon(t)$ does not satisfy this necessary condition. To see this, suppose that $x[n] = -N\Delta/2$ or $x[n] = N\Delta/2$ at some sample time n. Then to satisfy (3), the DEM encoder must set $c_i[n] = 0$ or $c_i[n] = 1$, respectively, for all $i = 1, 2, \ldots, N$. It follows from (3) and (10) that for either of these cases $\varepsilon(t)$ is deterministic, so it is equal to its expectation and is given by

$$\varepsilon(t) = p(t-nT)x[n] \tag{11}$$

during the nth sample interval where

$$p(t) = \frac{1}{N\Delta}\sum_{i=1}^{N}\left[e_{hi}(t) - e_{li}(t)\right]. \tag{12}$$

Thus, the expectation of $\varepsilon(t)$ depends on $x[n]$ for any mismatch error pulses that do not cause $p(t)$ to be zero.

The above reasoning implies that at least a component of $\varepsilon(t)$ must be a deterministic function of $x[n]$. If the deterministic function were nonlinear, then the effect of the mismatch error pulses would be to cause the DAC to introduce nonlinear distortion, which is unacceptable in most applications. Hence, the best possible outcome would be for the deterministic function to be linear. Given that (11) holds for the minimum and maximum values of $x[n]$, the only possible form for $\varepsilon(t)$ in which the deterministic function is linear is

$$\varepsilon(t) = p(t-nT)x[n] + e_{DAC}(t) \tag{13}$$

during the nth sample period, where $p(t)$ is given by (12), and $e_{DAC}(t)$ is a random process whose expectation is zero regardless of $x[n]$ and the mismatch error pulses. By definition $e_{DAC}(t) = 0$ during any sample interval in which $x[n] = -N\Delta/2$ or $x[n] = N\Delta/2$. Therefore, if the expectation of $e_{DAC}(t)$ were not zero over all sample intervals it would have the form of a nonlinear deterministic function of $x[n]$ plus a zero-mean random process.

It follows from (7) and (12) that $p(t) = \alpha(t) - a(t)$, where

$$\alpha(t) = \frac{1}{N}\sum_{i=1}^{N}\alpha_i(t). \tag{14}$$

Therefore, the analysis presented above implies that a necessary condition for a DEM DAC to avoid introducing nonlinear distortion is that its output during the nth sample interval is

$$y(t) = \alpha(t-nT)x[n] + \beta(t-nT) + e_{DAC}(t) \tag{15}$$

for each n, where $\alpha(t)$ is given by (14), $\beta(t)$ is given by (9), and $e_{DAC}(t)$ is a random process whose expectation is zero regardless of $x[n]$ and the mismatch error pulses. This is also a sufficient condition for $e_{DAC}(t)$ to be uncorrelated with $x[n]$. The objectives of DEM are to achieve this condition and, as described above, to further ensure that $e_{DAC}(t)$ is free of spurious tones, and spectrally shaped as appropriate for the application regardless of $x[n]$ and the mismatch error pulses.

The three components of the DAC's output signal in (15) are referred to as the *signal pulse sequence*, the *offset pulse sequence*, and the *DAC noise*, respectively [33]. The mismatch error pulses cause $\alpha(t)$ to deviate somewhat from the ideal unit output pulse, $a(t)$, but in most applications this is not a serious problem because it has little effect on the SNR or SFDR of the overall DAC. As described above, the offset pulse sequence does not degrade the SNR or the in-band SFDR of the overall DAC, and the objective of DEM is to render the DAC noise tolerable for the given application.

III. Decomposition of Arbitrary DEM Encoders Into Tree Structures

A. Preliminary Definitions

When considering the behavior of a DAC in the context of a signal processing system such as a $\Delta\Sigma$ data converter, it is convenient to interpret the sequence of input codewords to have values given by (1) as described above. However, when considering the operation of the DEM encoder, it is convenient to consider each codeword to represent the number of 1-bit DACs whose input bits must be set high during that sample interval, i.e.

$$c[n] = \frac{x[n]}{\Delta} + \frac{N}{2}. \tag{16}$$

Therefore (3) is equivalent to

$$c[n] = \sum_{i=1}^{N} c_i[n] \tag{17}$$

and (15) can be written as

$$y(t) = \alpha(t-nT)c[n]\Delta + \beta_c(t-nT) + e_{DAC}(t) \tag{18}$$

during the nth sample period, where

$$\beta_c(t) = \beta(t) - \alpha(t)\frac{N}{2}\Delta. \tag{19}$$

In order to simplify the subsequent analysis, the following definition from [34] is used.

DAC$^{(u,w)}$ Definition: For any integers u and w that satisfy $1 \le u \le w \le N$, DAC$^{(u,w)}$ consists of an encoder followed by the uth through wth 1-bit DACs of the DAC shown in Fig. 1. The encoder maps a digital input sequence given by

$$c^{(u,w)}[n] = \sum_{i=u}^{w} c_i[n] \tag{20}$$

to the same 1-bit sequences $c_u[n], c_{u+1}[n], \ldots, c_w[n]$ generated by the encoder shown in Fig. 1. The output of DAC$^{(u,w)}$ during the nth sample period is

$$y^{(u,w)}(t) = \sum_{i=u}^{w} y_i(t). \tag{21}$$

□

Following an analysis almost identical to that presented in the previous section (6) and (21) imply that

$$y^{(u,w)}(t) = \alpha^{(u,w)}(t-nT)c^{(u,w)}[n]\Delta + \beta_c^{(u,w)}(t-nT) + e_{DAC}^{(u,w)}(t) \tag{22}$$

where

$$\alpha^{(u,w)}(t) = \frac{1}{w-u+1}\sum_{i=u}^{w}\alpha_i(t) \tag{23}$$

and

$$\beta_c^{(u,w)}(t) = -\frac{\Delta}{2}(w-u+1)\alpha^{(u,w)}(t) + \sum_{i=u}^{w}\beta_i(t). \tag{24}$$

Note that for the special case of $u = w$, DAC$^{(u,u)}$ denotes the uth 1-bit DAC, and that $c^{(u,u)}[n] = c_u[n]$. Furthermore, a comparison of (6) and (22) implies that

$$e_{DAC}^{(u,u)}(t) = 0. \tag{25}$$

This is reasonable given that a 1-bit DAC has only two input levels; the mismatch error pulses give rise to a pulse shape error and an offset pulse, but no DAC noise as defined in the previous section.

B. Decomposition Analysis

It follows from (5), (16), (17), (20), and (21) that any DAC of the form shown in Fig. 1 can be redrawn in the equivalent form shown in Fig. 2 for any $g \in \{1, 2, \ldots, N-1\}$. The equivalent form consists of a digital block labeled $S^{(1,g,N)}$, called a *switching block*, and two sub-DACs, DAC$^{(1,g)}$ and

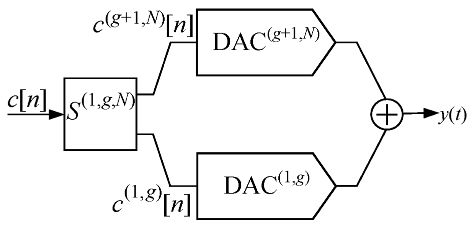

Fig. 2. Equivalent form of DAC in Fig. 1.

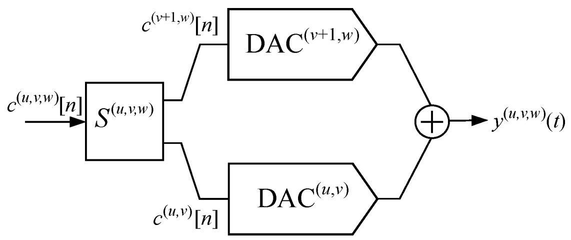

Fig. 3. Equivalent form of DAC$^{(u,w)}$.

DAC$^{(g+1,N)}$, the outputs of which are summed to form the overall DAC output.

The $S^{(1,g,N)}$ switching block converts the $c[n]$ sequence into the input sequences to DAC$^{(1,g)}$ and DAC$^{(g+1,N)}$, i.e., $c^{(1,g)}[n]$ and $c^{(g+1,N)}[n]$, respectively. It follows from (20) that the bottom and top output sequences from the $S^{(1,g,N)}$ switching block must be

$$c^{(1,g)}[n] = \sum_{i=1}^{g} c_i[n], \quad \text{and} \quad c^{(g+1,N)}[n] = \sum_{i=g+1}^{N} c_i[n], \tag{26}$$

respectively. Equivalently, these sequences can be rewritten as

$$c^{(1,g)}[n] = \frac{g}{N}c[n] + s^{(1,g,N)}[n], \\ \text{and} \\ c^{(g+1,N)}[n] = \frac{N-g}{N}c[n] - s^{(1,g,N)}[n], \tag{27}$$

respectively, where $s^{(1,g,N)}[n]$ is called a *switching sequence* and is given by

$$s^{(1,g,N)}[n] = \frac{N-g}{N}\sum_{i=1}^{g} c_i[n] - \frac{g}{N}\sum_{i=g+1}^{N} c_i[n]. \tag{28}$$

This can be verified by substituting (28) into (27) to obtain (26). It follows that the $S^{(1,g,N)}$ switching block can be viewed as a device that somehow generates $s^{(1,g,N)}[n]$ and uses it with (27) to obtain the switching block's two output sequences as functions of its input sequence.

It is next shown that the $s^{(1,g,N)}[n]$ switching sequence plays a key role in determining the behavior of the DAC noise. Given that

$$y(t) = y^{(1,g)}(t) + y^{(g+1,N)}(t), \tag{29}$$

(22) implies that during the nth sample period

$$\begin{aligned} y(t) = {} & \alpha^{(1,g)}(t-nT)c^{(1,g)}[n]\Delta \\ & + \beta_c^{(1,g)}(t-nT) + e_{DAC}^{(1,g)}(t) \\ & + \alpha^{(g+1,N)}(t-nT)c^{(g+1,N)}[n]\Delta \\ & + \beta_c^{(g+1,N)}(t-nT) + e_{DAC}^{(g+1,N)}(t). \end{aligned} \tag{30}$$

With (23), (24), and (27) this can be rewritten as

$$\begin{aligned} y(t) = {} & \alpha(t-nT)c[n]\Delta + \beta_c(t-nT) \\ & + s^{(1,g,N)}[n]\Delta^{(1,g,N)}(t-nT) + e_{DAC}^{(1,g)}(t) + e_{DAC}^{(g+1,N)}(t) \end{aligned} \tag{31}$$

where

$$\Delta^{(1,g,N)}(t) = \left[\alpha^{(1,g)}(t) - \alpha^{(g+1,N)}(t)\right]\Delta. \tag{32}$$

Comparison to (18) indicates that

$$\begin{aligned} e_{DAC}(t) = {} & s^{(1,g,N)}[n]\Delta^{(1,g,N)}(t-nT) \\ & + e_{DAC}^{(1,g)}(t) + e_{DAC}^{(g+1,N)}(t) \end{aligned} \tag{33}$$

during the nth sample period.

The above analysis trivially can be generalized to any DAC$^{(u,w)}$, where $1 \le u < w \le N$. Specifically, as illustrated in Fig. 3, DAC$^{(u,w)}$ can be decomposed into an $S^{(u,v,w)}$ switching block and two sub-DACs, DAC$^{(u,v)}$ and DAC$^{(v+1,w)}$, for any $v \in \{u, u+1, \ldots, w-1\}$. It follows from almost identical reasoning as used to obtain (27), (28), and (33) that the bottom and top outputs of the $S^{(u,v,w)}$ switching block are

$$c^{(u,v)}[n] = \frac{v-u+1}{w-u+1}c^{(u,w)}[n] + s^{(u,v,w)}[n], \\ \text{and} \\ c^{(v+1,w)}[n] = \frac{w-v}{w-u+1}c^{(u,w)}[n] - s^{(u,v,w)}[n], \tag{34}$$

respectively, where $s^{(u,v,w)}[n]$ is a switching sequence and is given by

$$s^{(u,v,w)}[n] = \frac{w-v}{w-u+1}\sum_{i=u}^{v} c_i[n] - \frac{v-u+1}{w-u+1}\sum_{i=v+1}^{w} c_i[n]. \tag{35}$$

During the nth sample period (22) holds with

$$\begin{aligned} e_{DAC}^{(u,w)}(t) = {} & s^{(u,v,w)}[n]\Delta^{(u,v,w)}(t-nT) \\ & + e_{DAC}^{(u,v)}(t) + e_{DAC}^{(v+1,w)}(t) \end{aligned} \tag{36}$$

where

$$\Delta^{(u,v,w)}(t) = \left[\alpha^{(u,v)}(t) - \alpha^{(v+1,w)}(t)\right]\Delta. \tag{37}$$

Fig. 4 shows the signal processing performed by switching block $S^{(u,v,w)}$, where

$$G^{(u,v,w)} = \frac{w-v}{w-u+1}.$$

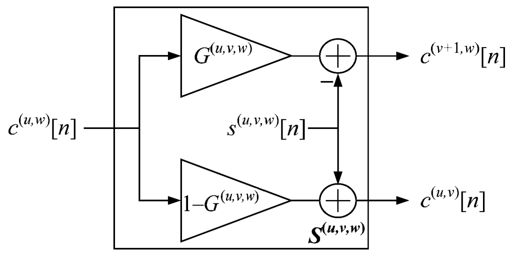

Fig. 4. Signal processing performed by $S^{(u,v,w)}$.

The above analysis shows that i) any DAC of the form shown in Fig. 1 can be decomposed as shown in Fig. 2, ii) if $g \geq 2$ then DAC$^{(1,g)}$ can be decomposed as shown in Fig. 3 for $u = 1$, $w = g$, and any $v = v_1$ where $v_1 \in \{1, 2, \ldots, g-1\}$, and iii) if $g \leq N-1$ then DAC$^{(g+1,N)}$ can be decomposed as shown in Fig. 3 for $u = g+1$, $w = N$, and any $v = v_2$ where $v_2 \in \{g+1, g+2, \ldots, N-1\}$. Using results (ii) and (iii) above, the last two terms in (33) can each be expanded via (36) to obtain

$$\begin{aligned} e_{DAC}(t) = {} & s^{(1,g,N)}[n]\Delta^{(1,g,N)}(t-nT) \\ & + s^{(1,v_1,g)}[n]\Delta^{(1,v_1,g)}(t-nT) \\ & + e_{DAC}^{(1,v_1)}(t) + e_{DAC}^{(v_1+1,g)}(t) \\ & + s^{(g+1,v_2,N)}[n]\Delta^{(g+1,v_2,N)}(t-nT) \\ & + e_{DAC}^{(g+1,v_2)}(t) + e_{DAC}^{(v_2+1,N)}(t). \end{aligned} \tag{38}$$

This decomposition process can be continued recursively, each time replacing a sub-DAC of the form DAC$^{(u,w)}$ in which $w \geq u+1$ by a new switching block $S^{(u,v,w)}$ and two new sub-DACs, DAC$^{(u,v)}$ and DAC$^{(v+1,w)}$. The recursive decomposition can be continued until the DAC of Fig. 1 has been transformed into a tree of switching blocks that drives N sub-DACs of the form DAC$^{(u,u)}$ for $u = 1, 2, \ldots, N$. By definition DAC$^{(u,u)}$ is just the uth 1-bit DAC, so the above analysis indicates that any encoder (DEM or otherwise) which satisfies (17) is equivalent to a tree of switching blocks with switching sequences given by (35). Furthermore, since each recursion allows one of the $e_{DAC}^{(u,w)}(t)$ terms in the $e_{DAC}(t)$ expression obtained from the previous recursion step to be expanded via (36), and $e_{DAC}^{(u,u)}(t) = 0$ for all u, it follows that

$$e_{DAC}(t) = \sum_{u,v,w} s^{(u,v,w)}[n]\Delta^{(u,v,w)}(t-nT) \tag{39}$$

during the nth sample period where the sum in (39) is taken over all values of u, v, and w used during the recursive decomposition process. By definition, $\Delta^{(u,v,w)}(t)$ for each u, v, and w is a fixed pulse that is zero outside of $0 \leq t < T$, so (39)) implies that the statistical properties of $e_{DAC}(t)$ are determined by the switching sequences.

At each step in the decomposition process, whenever $w-u \geq 2$ more than one choice exists for v. Therefore, a given encoder can be decomposed into several equivalent trees of switching blocks. Each such tree of switching blocks is called a *tree-structured encoder*. Fig. 5(a) and (b) show examples of tree-structured encoders obtained by decomposing DACs of the form shown in Fig. 1 with $N = 12$ and $N = 9$ 1-bit DACs, respectively.

The tree-structured encoders shown in Fig. 5 each contain $N-1$ switching blocks, and the following argument indicates that this result holds in general, i.e., that all tree-structured encoders contain exactly $N-1$ switching blocks. As described above, each of the recursion steps used to generate a given tree-structured encoder replaces a DAC of the form DAC$^{(u,w)}$ where $1 \leq u < w \leq N$ by a switching block and two new sub-DACs, DAC$^{(u,v)}$ and DAC$^{(v+1,w)}$. This places a dividing line between the vth and $(v+1)$th 1-bit DACs, and assigns all of the 1-bit DACs in DAC$^{(u,w)}$ below this dividing line to DAC$^{(u,v)}$ and all those above the dividing line to DAC$^{(v+1,w)}$. The recursion process ends when all sub-DACs are of the form DAC$^{(u,u)}$ for $1 \leq u \leq N$, i.e., when a dividing line has been placed between every pair of 1-bit DACs. A bank of N 1-bit DACs can contain up to $N-1$ such dividing lines, so exactly $N-1$ recursion steps are required to transform the encoder shown in Fig. 1 into a tree-structured encoder. Hence, the tree-structured encoder contains $N-1$ switching blocks.

The analysis presented above starts with an arbitrary encoder that satisfies (17) and shows that there exist multiple equivalent tree-structured encoders with switching sequences specified in terms of the 1-bit output sequences from the original encoder. Therefore, it implies that each tree-structured encoder is completely general in that with the appropriate choice of switching sequences it can mimic any given encoder that satisfies (17). Furthermore, (39) implies that the switching sequences specify the dynamics of the DAC noise. Hence, the derivation uses the tree-structured encoder as an analysis tool.

IV. Synthesis of Unity-Weighted Tree-Structured DEM Encoders

The results of the previous section are extended in this section to provide a method with which to synthesize tree-structured DEM encoders that have desired DAC noise properties. The synthesis method involves choosing one of the possible trees of switching blocks derived in the previous section, and then designing switching sequences that result in DAC noise with desired properties under the constraint that (17) is satisfied.

As in the previous section, first consider the $S^{(1,g,N)}$ switching block. It follows from (27) that the outputs of the switching block satisfy

$$c^{(1,g)}[n] + c^{(g+1,N)}[n] = c[n] \tag{40}$$

and that for each n the value of $s^{(1,g,N)}[n]$ determines how $c[n]$ is distributed between $c^{(1,g)}[n]$ and $c^{(g+1,N)}[n]$. Given that $c_i[n] \in \{0,1\}$ for each n, (26) implies that

$$c^{(1,g)}[n] \in \{0, 1, \ldots, g\} \text{ and } c^{(g+1,N)}[n] \in \{0, 1, \ldots, N-g\}. \tag{41}$$

Therefore, for any value of n at which $c[n] = 0$ the only way to satisfy (40) and (41) is to have $c^{(1,g)}[n] = 0$ and $c^{(g+1,N)}[n] =$

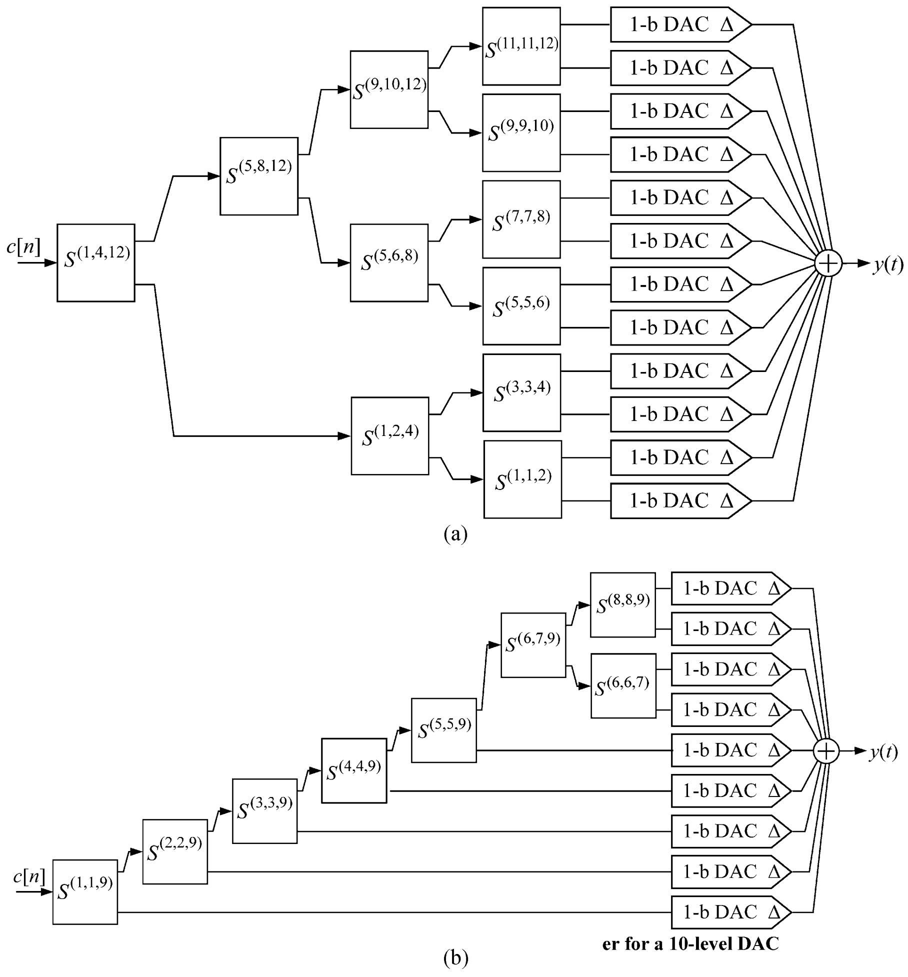

Fig. 5. (a) A type of tree-structured DEM encoder for a 13-level DAC. (b) A type of tree-structured DEM encoder for a 10-level DAC.

0, which implies that $s^{(1,g,N)}[n]$ must be zero. Similarly, for any value of n at which $c[n] = N$ the only way to satisfy (40) and (41)) is to have $c^{(1,g)}[n] = g$ and $c^{(g+1,N)}[n] = N - g$, which implies that $s^{(1,g,N)}[n]$ must be zero.

For each other possible value of $c[n]$, i.e., each value in the set $\{1, 2, \ldots, N-1\}$, there exist at least two valid choices of $s^{(1,g,N)}[n]$ because there are at least two different choices of $c^{(1,g)}[n]$ and $c^{(g+1,N)}[n]$ that satisfy (40) and (41). These two valid choices of $s^{(1,g,N)}[n]$ are

$$s^{(1,g,N)}[n] = \left\langle \frac{N-g}{N} c[n] \right\rangle \text{ and } s^{(1,g,N)}[n] = \left\langle \frac{N-g}{N} c[n] \right\rangle - 1 \tag{42}$$

where $\langle b \rangle = b - \lfloor b \rfloor$ denotes the fractional part of b and $\lfloor b \rfloor$ denotes the largest integer less than or equal to b. This can be verified by substituting the left and right equations of (42) into (27). Substituting the left equation of (42) into (27) yields

$$c^{(1,g)}[n] = c[n] - \left\lfloor \frac{N-g}{N} c[n] \right\rfloor, \text{ and}$$
$$c^{(g+1,N)}[n] = \left\lfloor \frac{N-g}{N} c[n] \right\rfloor. \tag{43}$$

and substituting the right equation of (42) into (27) yields

$$c^{(1,g)}[n] = c[n] - \left\lfloor \frac{N-g}{N} c[n] \right\rfloor - 1, \text{ and}$$
$$c^{(g+1,N)}[n] = \left\lfloor \frac{N-g}{N} c[n] \right\rfloor + 1. \tag{44}$$

In both cases (41) is satisfied whenever $c[n] \in \{1, 2, \ldots, N-1\}$ as required. Thus for each n at which $c[n] \in \{1, 2, \ldots, N-1\}$ there must be sets of bits $\{c_1[n], c_2[n], \ldots, c_N[n]\}$ that cause (28) to take on the values implied by (42).

The above analysis trivially can be generalized to any switching block $S^{(u,v,w)}$, where $1 \le u < w \le N$ with only changes in the notation. Specifically, following identical reasoning as above indicates that valid choices of $s^{(u,v,w)}[n]$ are

$$s^{(u,v,w)}[n] = \begin{cases} 0, & \text{if } c^{(u,w)}[n] \in \{0, w-u+1\}, \\ m \text{ or } m-1, & \text{otherwise} \end{cases} \tag{45}$$

where

$$m = \left\langle \frac{w-v}{w-u+1} c^{(u,w)}[n] \right\rangle.$$

Thus for each n there must be sets of bits $\{c_u[n], c_{u+1}[n], \ldots, c_w[n]\}$ that cause (35) to take on the values implied by (45).

Conversely, if a tree-structured encoder is designed in which all the switching sequences satisfy (45), then the N 1-bit output sequences from the encoder will satisfy (17) and the DAC noise will satisfy (39).

Note that (45) is not a general expression because it does not represent all possible values that can be assumed by (35). This implies that tree-structured encoders with switching sequences that satisfy (45) are not capable of mimicking any conceivable encoder that satisfies (17). Nevertheless, as shown below and demonstrated in the next section, the switching sequence values implied by (45) are sufficient to achieve desirable DAC noise properties.

As shown in Section II, a necessary condition for a DEM DAC to avoid introducing nonlinear distortion is for the expectation of $e_{DAC}(t)$ to be zero for all t, regardless of the mismatch error pulses and $x[n]$. A tree-structured DAC with switching sequences that satisfy (45) can achieve this necessary condition because for each n, regardless of the value of $c[n]$, every switching sequence that satisfies (45) has a value that is zero or is either of two known non-zero values, one of which is positive and the other negative. Therefore each switching block can exercise its choice of possible switching sequence values to ensure that the expectation of each switching sequence is zero. In this case, (39) implies that $e_{DAC}(t)$ satisfies the necessary condition.

Furthermore, for each n at which (45) allows the switching block to have a choice of two non-zero values with opposite signs, the choice can be made without regard to the switching block's input sequence, and therefore without regard to $c[n]$. This makes it possible to use switching sequences that are spectrally shaped random processes which are uncorrelated with $c[n]$. An example of such a DEM DAC is presented in the next section.

V. A Design Example

The design of a 13-level DEM DAC with a tree-structured DEM encoder for use in a second-order $\Delta\Sigma$ ADC is described in this section. The objective of the DEM DAC for this application is to cause the DAC noise to be a random process with an expectation of zero, to be uncorrelated with $c[n]$, to be free of spurious tones, and to be highpass shaped, all regardless of the mismatch error pulses. The tree-structured DEM encoder described in Section III and shown in Fig. 5(a) with switching blocks that perform the signal processing operations shown in Fig. 4 is used as the starting point for the design. The design tasks are to choose appropriate switching sequences and devise digital logic that generates the switching sequences.

The PSD of a DAC waveform can be estimated in the laboratory using a spectrum analyzer, or, analogously, in simulation using periodogram analysis [35]. Therefore, the spectral properties of the switching sequences are derived below in terms of their periodograms. The length-L periodogram of $s^{(u,v,w)}[n]$ is given by

$$I_{s^{(u,v,w)},L}(\omega) = \frac{1}{L}\left|\sum_{n=0}^{L-1} s^{(u,v,w)}[n]e^{-j\omega n}\right|^2, \tag{46}$$

which can be written equivalently as

$$I_{s^{(u,v,w)},L}(\omega) = \frac{1}{L}\sum_{n=0}^{L-1}\sum_{m=0}^{L-1} s^{(u,v,w)}[n]s^{(u,v,w)}[m]e^{-j\omega(n-m)}. \tag{47}$$

It is well known that in certain cases the expectation of the periodogram converges to the true PSD function in the limit as $L \to \infty$, but in a DAC application this is not a requirement, or even relevant to the measured performance.

First consider the $s^{(1,4,12)}[n]$ switching sequence. Evaluation of (45) for all possible values of $c[n]$ yields

$$s^{(1,4,12)}[n] = \begin{cases} 0 & \text{if } c[n] \in \{0,12\} \\ 0 \text{ or } -1 & \text{if } c[n] \in \{3,6,9\} \\ \frac{1}{3} \text{ or } -\frac{2}{3} & \text{if } c[n] \in \{2,5,8,11\} \\ \frac{2}{3} \text{ or } -\frac{1}{3} & \text{if } c[n] \in \{1,4,7,10\} \end{cases}. \tag{48}$$

It follows from (48) that for the expectation of $s^{(1,4,12)}[n]$ to be zero regardless of $c[n]$ (and therefore to be uncorrelated with $c[n]$), the probability distribution of $s^{(1,4,12)}[n]$ must satisfy

$$\mathrm{P}\left(s^{(1,4,12)}[n] = 0\right) = 1 \quad \text{and} \quad \mathrm{P}\left(s^{(1,4,12)}[n] = -1\right) = 0$$

when $c[n] \in \{0,3,6,9,12\}$,

$$\mathrm{P}\left(s^{(1,4,12)}[n] = \frac{1}{3}\right) = \frac{2}{3} \text{ and } \mathrm{P}\left(s^{(1,4,12)}[n] = -\frac{2}{3}\right) = \frac{1}{3}$$

when $c[n] \in \{2,5,8,11\}$, and

$$\mathrm{P}\left(s^{(1,4,12)}[n] = \frac{2}{3}\right) = \frac{1}{3} \text{ and } \mathrm{P}\left(s^{(1,4,12)}[n] = -\frac{1}{3}\right) = \frac{2}{3} \tag{49}$$

when $c[n] \in \{1,4,7,10\}$, where $\mathrm{P}(s^{(1,4,12)}[n] = \lambda)$ denotes the probability that $s^{(1,4,12)}[n]$ is equal to λ.

As a special case, first suppose that $c[n] \in \{2,5,8,11\}$ for all n. Then one way to satisfy (49) is to let $(s^{(1,4,12)}[3k], s^{(1,4,12)}[3k+1], s^{(1,4,12)}[3k+2])$ be an independent sequence of random variable triples that take on values of (1/3, 1/3, −2/3), (1/3, −2/3, 1/3) and (−2/3, 1/3, 1/3) with equal probability. The sum of terms in each triple is zero, so it follows from (46) that $I_{s^{(u,v,w)},L}(0) < 1/L$ which implies that the expectation of $I_{s^{(u,v,w)},L}(\omega)$ goes to zero at $\omega = 0$ as $L \to \infty$. Although the expectation of $I_{s^{(u,v,w)},L}(\omega)$ does not go to zero as $L \to \infty$when $\omega \neq 0$, (47) and the independence of the triples imply that the expectation of $I_{s^{(u,v,w)},L}(\omega)$ is uniformly bounded for all L and ω. Hence, the sequence is highpass shaped and is free of spurious tones as desired.

A digital logic block that generates $s^{(1,4,12)}[n]$ for this special case is shown in Fig. 6. Three flip-flops preloaded with bit values of 1, 1, and 0, respectively, are configured as a re-circulating shift register clocked once per DAC sample interval. Thus, the Q output of the left-most flip-flop is the periodic sequence 1, 1, 0, 1, 1, 0, . . ., and the Q outputs of the middle and right-most flip flops are the same sequence except delayed by one and two DAC sample intervals, respectively. A three-to-one MUX selects as its output one of the three flip-flop outputs based on a pseudo-random number that is updated once every three DAC sample intervals. Each pseudo-random number is chosen independently

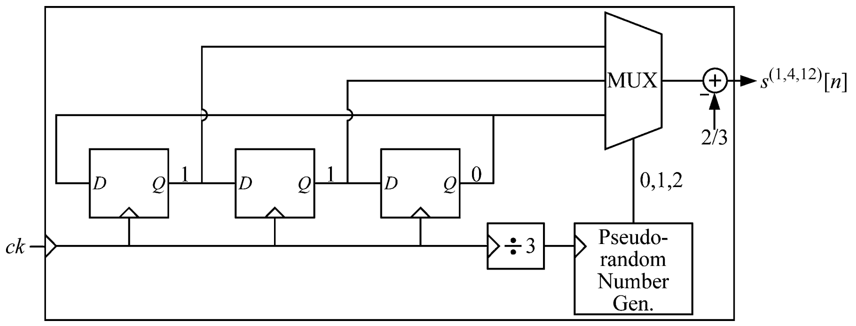

Fig. 6. Register transfer level circuitry that generates $s^{(1,4,12)}[n]$ for $c[n] \in \{2, 5, 8, 11\}$.

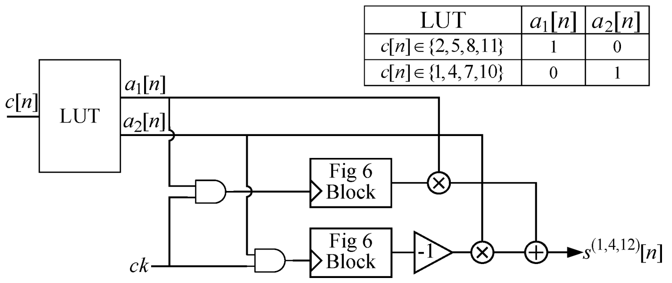
LUT | $a_1[n]$ | $a_2[n]$
$c[n] \in \{2,5,8,11\}$ | 1 | 0
$c[n] \in \{1,4,7,10\}$ | 0 | 1
$c[n]$
LUT
$a_1[n]$
$a_2[n]$
Fig 6 Block
Fig 6 Block
-1
ck
$s^{(1,4,12)}[n]$

Fig. 7. Register transfer level circuitry that generates $s^{(u,v,w)}[n]$ for $(u, v, w) = (1, 4, 12)$.

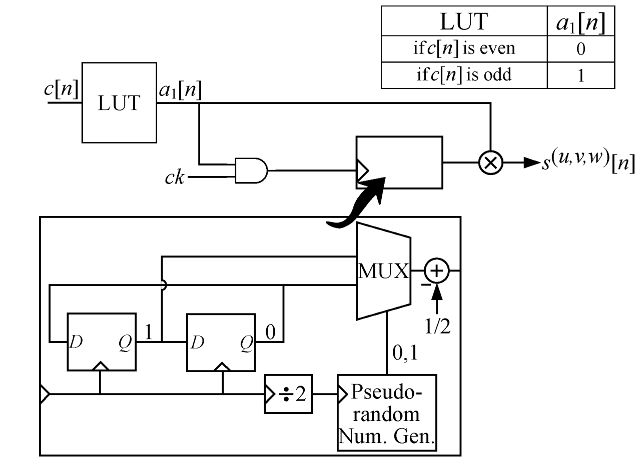

Fig. 8. Register transfer level circuitry that generates $s^{(u,v,w)}[n]$ for $(u, v, w) \neq (1, 4, 12)$.

and takes on values of 0, 1, and 2 with equal probability. A value of 2/3 is subtracted from the output of the multiplexer to cause the final sequence to take on values of 1/3, 1/3, and −2/3, as required.

The other special cases can be handled similarly. If $c[n] \in \{1, 4, 7, 10\}$ for all n, the same strategy can be used except with triples that take on values of (−1/3, −1/3, 2/3), (−1/3, 2/3, −1/3) and (2/3, −1/3, −1/3) with equal probability. It is straightforward to verify that the digital logic block shown in Fig. 6 with its output multiplied by −1 can be used to generate $s^{(1,4,12)}[n]$ for this special case. Alternatively, if $c[n] \in \{0, 3, 6, 9, 12\}$ for all n, then $s^{(1,4,12)}[n] = 0$ can be used. This satisfies (49) and has the benefit that it does not contribute at all to the DAC noise.

In general, any $c[n]$ sequence can be constructed by interlacing subsequences corresponding to the three special cases described above, and the $s^{(1,4,12)}[n]$ switching sequence can be formed by correspondingly interlacing the switching sequences described above for each subsequence. Since each of the interlaced switching sequences is dc-free, highpass shaped, and free of spurious tones, $s^{(1,4,12)}[n]$ inherits these properties.

A digital logic block that generates the $s^{(1,4,12)}[n]$ is shown in Fig. 7. It generates switching sequences for the three special cases described above and combines them to implement the interlacing operation. Therefore, the structure of Fig. 4 with $G^{(1,4,12)} = 2/3$ and the logic block shown in Figs. 6 and 7 make up the $S^{(1,4,12)}$ switching block.

Applying the same procedure to design the remaining switching sequences yields

$$s^{(u,v,w)}[n] = \begin{cases} 0, & \text{if } c^{(u,w)}[n] \text{ is even,} \\ \lambda^{(u,v,w)}[n], & \text{if } c^{(u,w)}[n] \text{ is odd,} \end{cases} \tag{50}$$

for each $(u, v, w) \neq (1, 4, 12)$, where $\lambda^{(u,v,w)}[n]$ is a highpass shaped random sequence each sample of which takes on values of ±1/2 with equal probability. A digital block that implements (50) is shown in Fig. 8. Each of the corresponding switching blocks consists of the structure of Fig. 4 with $G^{(u,v,w)} = 1/2$ and the digital block shown in Fig. 8. Although the notation is slightly different to allow for the generalizations presented in this paper, it is straightforward to verify that for this special case the switching block is equivalent to that presented in [36] for unity-weighted DACs with a power-of-two number of 1-bit DACs.

When used in a $\Delta\Sigma$ ADC, any input-output latency imposed by the DEM encoder adds to the delay around the feedback paths within the $\Delta\Sigma$ ADC so it must be considered when designing the $\Delta\Sigma$ ADC. Although a tree-structured DEM encoder contains clocked components (e.g., the components shown in Figs. 7 and 8), these components are not in the data path; they are only used to generate the switching sequences, so they only need to be fast enough to generate switching sequence samples at the sample-rate of the DEM DAC. In contrast, the latency of the DEM encoder is determined by how fast the operations shown in Fig. 4 occur within each switching block, and the largest number of cascaded switching blocks within the DEM encoder.

In general, the largest number of cascaded switching blocks within a tree-structured DEM encoder for a given number, N, of 1-bit DACs depends on the choices made during the recursion

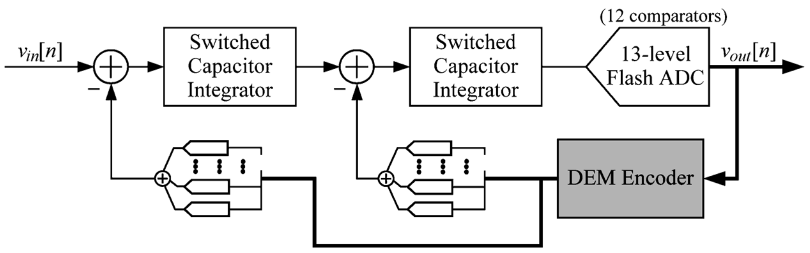

Fig. 9. Block diagram of the second order ΔΣ ADC.

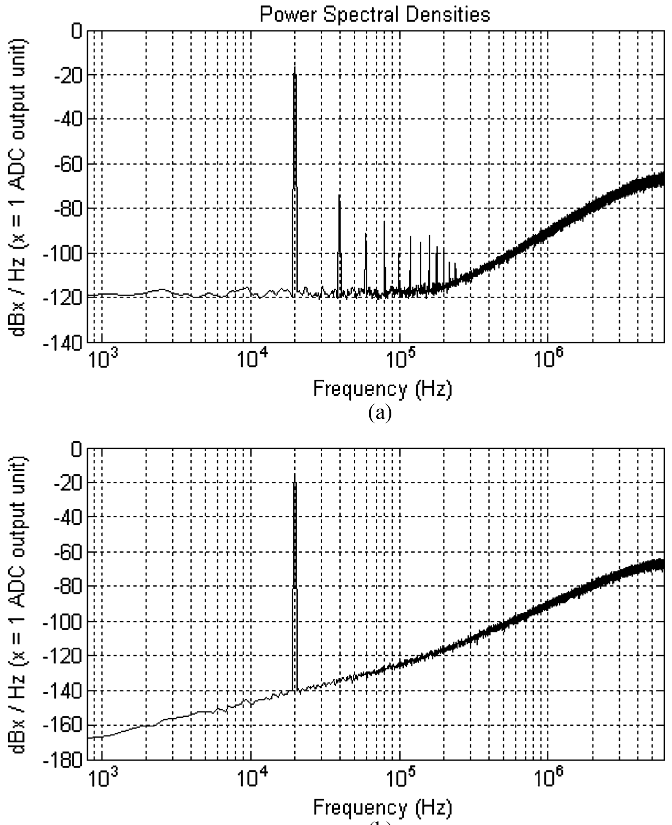

Fig. 10. (a) Power spectral density at the output of the second order ΔΣ ADC in Fig. 9 without the DEM Encoder. (b) Power spectral density at the output of the second order ΔΣ ADC in Fig. 9 with the tree-structured DEM encoder shown in Fig. 5.

process described in Section III. It is straightforward to verify that the minimum value of this number is the smallest integer greater than or equal to $\log_2 N$ and that this minimum value is achieved by at least one of the possible tree-structures. The DEM encoders shown in Fig. 5 are such examples.

If necessary, the latency of the DEM encoder can be reduced at the expense of increased complexity. One approach is to represent $c[n]$ as a thermometer code and flatten the tree structure into an equivalent single layer of transmission gates as described in [38]. Other approaches involve modifying the binary number formats used by the individual switching blocks to reduce latency as described in [36].

Fig. 9 shows the block diagram of a second order ΔΣ ADC that contains two 13-level DEM DACs of the type designed above. As is common practice in such ADCs, both DEM DACs share the same DEM encoder to save circuit area [13]. Fig. 10(a) and (b) show output spectra from a computer simulation of the ΔΣ ADC with ideal components except for 1-bit DAC mismatches. The simulated 1-bit DAC mismatches were chosen from a Gaussian distribution with a standard deviation of 1%. The simulated input sequence is the sum of a full-scale sinusoid and a small amount of white noise to act as dither [37]. Fig. 10(a) shows the output spectrum from the ΔΣ ADC simulated without the DEM encoder. As expected, the 1-bit DAC mismatches introduce significant distortion in this case. Fig. 10(b) shows the output spectrum from the ΔΣ ADC simulated with the DEM encoder described above. As expected, the 1-bit DAC mismatches give rise to highpass shaped DAC noise free of spurious tones.

VI. Conclusion

A generalized tree-structured DEM encoder that can drive any number of 1-bit DACs is presented in this paper. It removes the limitation of prior work which requires the number of 1-bit DACs to be a power of two. The analysis section of the paper proves that tree-structured encoders with appropriately chosen switching sequences can mimic the behavior of any DAC encoder. The synthesis section presents a way to design switching sequences for any tree-structured DEM encoder such that the DAC noise arising from mismatches is uncorrelated with the DAC's input sequence, spectrally shaped, and free of spurious tones. The last section of the paper demonstrates the key points of the paper in the context of a second order ΔΣ ADC.

References

[1] L. R. Carley and J. Kenny, "A 16-bit 4th order noise-Shaping D/A converter," in *1988 IEEE Custom Integrated Circuits Conf.*, May 1988.

[2] L. R. Carley, "A noise-shaping coder topology for 15 + bit converters," *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 267–273, Apr. 1989.

[3] B. H. Leung and S. Sutarja, "Multi-bit ΣΔ A/D converter incorporating a novel class of dynamic element shaping," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 39, pp. 35–51, Jan. 1992.

[4] F. Chen and B. H. Leung, "A high resolution multibit sigma-delta modulator with individual level averaging," *IEEE J. Solid-State Circuits*, vol. 30, pp. 453–460, Apr. 1995.

[5] M. J. Story, "Digital to Analogue Converter Adapted to Select Input Sources Based on a Preselected Algorithm Once Per Cycle of a Sampling Signal," U.S. Patent 5,138,317, Aug. 11, 1992.

[6] W. Redman-White and D. J. L. Bourner, "Improved dynamic linearity in multi-level ΔΣ(converters by spectral dispersion of D/A distortion products," in *IEEE Eur. Conf. Circuit Theory and Design*, Sep. 1989.

[7] H. S. Jackson, "Circuit and Method of Cancelling Nonlinearity Error Associated With Component Mismatches in a Data Converter," U.S. Patent 5,221,926, Jun. 22, 1993.

[8] R. T. Baird and T. S. Fiez, "Improved ΔΣ DAC linearity using data weighted averaging," in *IEEE Int. Symp. Circuits and Systems*, May 1995.

[9] R. T. Baird and T. S. Fiez, "Linearity enhancement of $\Delta\Sigma$ A/D and D/A converters using data weighted averaging," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, pp. 753–762, Dec. 1995.

[10] 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. 1995.

[11] R. Schreier and B. Zhang, "Noise-shaped multibit D/A converter employing unit elements," *Electron. Lett.*, vol. 31, pp. 1712–1713, Sep. 1995.

[12] I. Galton, "Spectral shaping of circuit errors in digital-to-analog converters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, pp. 808–817, Oct. 1997.

[13] E. Fogleman, I. Galton, W. Huff, and H. Jensen, "A 3.3-V single-poly CMOS audio ADC delta-sigma modulator with 98-dB peak SINAD and 105-dB peak SFDR," *IEEE J. Solid-State Circuits*, vol. 35, no. 3, pp. 297–307, Mar. 2000.

[14] E. Fogleman, J. Welz, and I. Galton, "An audio ADC Delta-Sigma modulator with 100-dB peak SINAD and 102-dB DR using a second-order mismatch-shaping DAC," *IEEE J. Solid-State Circuits*, vol. 36, no. 3, pp. 339–348, Mar. 2001.

[15] I. Galton, "Delta-sigma data conversion in wireless transceivers," *IEEE Trans. Microw. Theory Tech.*, vol. 50, no. 1, pp. 302–316, Jan. 2002.

[16] J. Grilo, I. Galton, K. Wang, and R. G. Montemayor, "A 12-mW ADC delta-sigma modulator with 80 dB of dynamic range integrated in a single-chip Bluetooth transceiver," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 271–278, Mar. 2002.

[17] T. Shui, R. Schreier, and F. Hudson, "Mismatch shaping for a current-mode multibit delta-sigma DAC," *IEEE J. Solid-State Circuits*, vol. 34, no. 3, pp. 331–338, Mar. 1999.

[18] I. Fujimori, A. Nogi, and T. Sugimoto, "A multibit delta-sigma audio DAC with 120-dB dynamic range," *IEEE J. Solid-State Circuits*, vol. 35, no. 8, pp. 1066–1073, Aug. 2000.

[19] R. E. Radke, A. Eshraghi, and T. F. Fiez, "A 14-bit current-mode $\Sigma\Delta$ DAC based upon rotated data weighted averaging," *IEEE J. Solid-State Circuits*, vol. 35, no. 8, pp. 1074–1084, Aug. 2000.

[20] E. Tuijl, J. Homberg, D. Reefman, C. Bastiaansen, and L. Dussen, "A 128fs, multi-bit $\Sigma\Delta$ CMOS audio DAC with real-time DEM and 115 dB SFDR," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2004.

[21] M. Clara, W. Klatzer, A. Wiesbauser, and D. Straeussnigg, "A 350MHz low-OSR $\Sigma\Delta$ current-steering DAC with active termination in $0.13\,\mu$m CMOS," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2005.

[22] Z. Zhang and G. Temes, "A segmented data-weighted-averaging technique," in *IEEE Int. Symp. Circuits and Systems*, May 2007.

[23] T. S. Kaplan, J. F. Jensen, C. H. Fields, and M. F. Chang, "A 2-GS/s 3-bit $\Sigma\Delta$-modulated DAC with tunable bandpass mismatch shaping," *IEEE J. Solid-State Circuits*, vol. 40, no. 3, pp. 603–610, Mar. 2005.

[24] S. Reekmans, J. D. Maeyer, P. Rombouts, and L. Weyten, "Quadrature mismatch shaping for digital-to-analog converters," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 12, pp. 2529–2538, Dec. 2006.

[25] H. Hsieh and L. Lin, "A first-order tree-structured DAC with reduced signal-band noise," *IEEE Trans. Circuits Syst. II: Expr. Briefs*, vol. 54, no. 5, pp. 392–396, May 2007.

[26] M. Vadipour, "Techniques for preventing tonal behavior of data weighted averaging algorithm in $\Sigma\Delta$ modulators," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 47, pp. 1137–1144, Nov. 2000.

[27] J. Arias, P. Kiss, V. Boccuzzi, L. Quintanilla, L. Enriquez, J. Vicente, D. Bishal, J.S. Pablo, and J. Barbolla, "Nonlinearity correction for multibit $\Delta\Sigma$ DACs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 6, pp. 1033–1041, Jun. 2005.

[28] A. A. Hamoui and K. W. Martin, "High-order multibit modulators and pseudo data-weighted-averaging in low-oversampling $\Delta\Sigma$ ADCs for broad-band applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 1, pp. 72–85, Jan. 2004.

[29] D. H. Lee and T. H. Kuo, "Advancing data weighted averaging technique for multi-bit sigma-delta," *IEEE Trans. Circuits Syst. II: Expr. Briefs*, vol. 54, no. 10, pp. 838–842, Oct. 2007.

[30] R. Schreier and G. C. Temes, *Understanding Delta-Sigma Data Converters*. New York: Wiley, 2005.

[31] R. Adams, K. Nguyen, and K. Sweetland, "A 113-dB SNR oversampling DAC with segmented noise-shaped scrambling," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1871–1878, Dec. 1998.

[32] K. Vleugels, S. Rabii, and B. A. Wooley, "A 2.5V sigma-delta modulator for broadband communications applications," *IEEE J. Solid-State Circuits*, vol. 36, pp. 1887–1899, Dec. 2001.

[33] 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, pp. 2067–2078, Sep. 2008.

[34] 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, pp. 3383–3392, Dec. 2008.

[35] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[36] J. Welz and I. Galton, "Simplified logic for first-order and second-order mismatch-shaping digital-to-analog converters," *IEEE Trans. Circuits Syst. II, Expr. Briefs*, vol. 48, no. 11, pp. 1014–1027, Nov. 2001.

[37] I. Galton, "Granular quantization noise in a class of delta-sigma modulators," *IEEE Trans. Inf. Theory*, vol. 40, no. 3, pp. 848–859, May 1994.

[38] E. Siragusa and I. Galton, "A digitally enhanced 1.8 V 15 b 40 MS/s CMOS pipelined ADC," *IEEE J. Solid-State Circuits*, vol. 39, no. 12, pp. 2126–2138, Dec. 2004.

Nevena Rakuljic (M'06) received the B.S. and M.S. degrees from the University of California at San Diego in 2006 and 2008, respectively, where she is currently working toward her Ph.D. degree.

For the last two years she has been a part of the Integrated Signal Processing Group where she has worked on dynamic element matching for multibit DACs and digital correction of higher order non-linearities in pipelined ADCs.

Ian Galton (M'92–SM'09) received the Sc.B. degree from Brown University, Providence, RI, in 1984, and the M.S. and Ph.D. degrees from the California Institute of Technology, Pasadena, in 1989 and 1992, respectively, all in electrical engineering.

Since 1996, he has been a Professor of electrical engineering with the University of California at San Diego, La Jolla, where he teaches and conducts research in the field of mixed-signal integrated circuits and systems for communications. Prior to 1996, he was with University of California, Irvine, and, prior to 1989, he was with Acuson and Mead Data Central. His research involves the invention, analysis, and integrated circuit implementation of critical communication system blocks such as data converters, frequency synthesizers, and clock recovery systems. In addition to his academic research, he regularly consults at several semiconductor companies and teaches industry-oriented short courses on the design of mixed-signal integrated circuits.

Dr. Galton has served on a corporate Board of Directors, on several corporateTechnical Advisory Boards, as the Editor-in-Chief of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: ANALOG AND DIGITAL SIGNAL PROCESSING, as a member of the IEEE Solid-State Circuits Society Administrative Committee, as a member of the IEEE Circuits and Systems Society Board of Governors, as a member of the IEEE International Solid-State Circuits Conference Technical Program Committee, and as a member of the IEEE Solid-State Circuits Society Distinguished Lecturer Program.