A REDUCED-COMPLEXITY MISMATCH-SHAPING DAC FOR DELTA-SIGMA DATA CONVERTERS

Henrik T. Jensen *Ian Galton*

Department of Electrical & Computer Engineering
UNIVERSITY OF CALIFORNIA, SAN DIEGO, La Jolla, CA 92093-0407, USA.
htjensen@ece.ucsd.edu galton@ece.ucsd.edu

ABSTRACT

Although mismatch-shaping multi-bit DACs offer many advantages over single-bit DACs in delta-sigma data converters, most of the presently known mismatch-shaping DAC architectures involve a large amount of digital processing. This paper presents digital structures that implement previously published tree-structured mismatch-shaping DAC algorithms yet offer significant reductions in the digital processing.

1. INTRODUCTION

As is well-known, static mismatches in the analog output levels cause conventional multi-bit DACs employed in $\Delta\Sigma$ data converters to introduce *DAC noise* that resides in the signal-band and thus drastically reduces the conversion precision and achievable spurious-free dynamic range (SFDR). A number of investigators have recently proposed multi-bit *mismatch-shaping* DAC architectures specifically for use in $\Delta\Sigma$ data converters [1], [2], [3], [4]. These DAC architectures employ a bank of *unit DAC-elements* whose outputs are summed to form a multi-bit DAC. An all-digital encoder dynamically selects the unit DAC elements such that the DAC noise is modulated to reside primarily outside of the signal-band. To eliminate or attenuate signal-dependent spurious tones (spurs) in the DAC-noise, some of the mismatch-shaping algorithms allow for the inclusion of a *dithering scheme.*

Many efforts continue to be devoted to simplifying the digital processing required to implement the mismatch-shaping algorithms. An architecture that promises to be particularly hardware efficient and amenable to dithering was first proposed in [5] and analyzed in detail in [4]. A recent publication [6] reported an architecture that allows for very hardware efficient implementations of mismatch-shaping algorithms, demonstrating the performance of first- and second-order versions with simulation data. This paper shows that the architecture in [6] is a special case of that proposed in [4] and extends the first-order algorithm to include a low-complexity dithering scheme.

2. REVIEW OF THE DAC ARCHITECTURE

To briefly review the mismatch-shaping DAC architecture proposed in [4], consider the 3-bit version shown in Figure 1. The input, $x[n]$, is a sequence of unsigned 3-bit integers. The DAC consists of 3 layers of digital devices, each device referred to as a *switching block*, followed by 8 one-bit DACs, each referred to as a *unit DAC-element*, and an analog summing node which generates the output $y[n]$. Each switching block is labeled $S_{k,r}$, where k denotes the layer number and r denotes the position of the switching block within the layer. The unit DAC-elements each operate according to

$$y_i[n] = \begin{cases} 1 + e_{h_i}, & \text{if } x_i[n] = 1; \\ e_{l_i}, & \text{if } x_i[n] = 0; \end{cases}$$

where $x_i[n]$ and $y_i[n]$ denote the input and output, respectively, of the i^{th} unit DAC-element, and e_{h_i} and e_{l_i} are errors in the analog output levels. The errors are assumed to be time-invariant, but otherwise arbitrary, and are referred to as *static DAC-element errors.*

When employing a conventional multi-bit DAC in a $\Delta\Sigma$ data converter, the static DAC-element errors give rise to DAC noise that severely limits the achievable conversion precision. As an example of this, consider the data shown in Figure 2 resulting from simulations of a second-order $\Delta\Sigma$ADC driven by a sinusoidal input and employing a 3-bit conventional DAC in the feedback path. The static DAC-element errors were chosen randomly from a Gaussian distribution with a standard deviation of 1%. Figure 2a shows the power spectral density (PSD) of the output of the $\Delta\Sigma$ADC plotted against a normalized logarithmic frequency scale. The PSD is given in dB relative to Δ^2, where Δ is the quantizer step size. At high frequencies, the PSD exhibits the well-known second-order shaping of the quantization noise, whereas at low frequencies the DAC noise is dominant. Figure 2b shows a close-up of the PSD of the DAC noise, which is seen to consist of a random component and spurs spread throughout the spectrum. Operating this particular $\Delta\Sigma$ADC at an oversampling ratio of 64 with an ideal DAC yields a signal-band noise power of -88 dB rel. to Δ^2, corresponding to a conversion precision of 15.2 bits. The conversion precision of the $\Delta\Sigma$ADC employing the non-ideal DAC is merely 12.6 bits, a reduction of 2.6 bits.

The performance of the $\Delta\Sigma$ADC can be restored by modulating the DAC noise away from the signal-band. To achieve this, the switching blocks $S_{k,r}$ perform signal processing as shown in Figure 3. The switching block has one $k+1$-bit input, $x_{k,r}[n]$, and two k-bit outputs, $x_{k-1,2r-1}[n]$ and $x_{k-1,2r}[n]$. The i^{th} bit of the input is referred to by $x_{k,r}^{(i)}[n]$. The outputs are formed according to

$$x_{k-1,2r-1}[n] = \frac{1}{2}(x_{k,r}[n] + s_{k,r}[n]) \qquad (1)$$

and

$$x_{k-1,2r}[n] = \frac{1}{2}(x_{k,r}[n] - s_{k,r}[n]), \quad (2)$$

respectively, where

$$s_{k,r}[n] = x_{k-1,2r-1}[n] - x_{k-1,2r}[n]$$

is a *switching sequence* generated within $S_{k,r}$.

2.1 First Order Algorithms

As shown in [4], the basic algorithm for generating $s_{k,r}[n]$ sequences that result in first order mismatch-shaping is shown in Figure 4. The sequence $o_{k,r}[n]$ is such that

$$o_{k,r}[n] = \begin{cases} 0, & \text{if } x_{k,r}[n] \text{ is even;} \\ 1, & \text{if } x_{k,r}[n] \text{ is odd.} \end{cases} \quad (3)$$

The hard limiter operates according to

$$v_{k,r}[n] = \begin{cases} 1, & \text{if } u_{k,r}[n] \geq 0; \\ -1, & \text{if } u_{k,r}[n] < 0; \end{cases} \quad (4)$$

where $v_{k,r}[n]$ and $u_{k,r}[n]$ denote the input and output, respectively. When employing this *deterministic* hard limiter, a number of spurs typically remain in the signal-band and may prohibitively degrade the SFDR of the $\Delta\Sigma$ADC. To eliminate tonal behavior, a dithering scheme can be included by defining the operation of the hard limiter according to

$$v_{k,r}[n] = \begin{cases} 1, & \text{if } u_{k,r}[n] = 1; \\ -1, & \text{if } u_{k,r}[n] = -1; \\ r_k[n], & \text{if } u_{k,r}[n] = 0; \end{cases} \quad (5)$$

where $r_k[n]$ is a pseudo-random ± 1 sequence that is white, independent of $x[n]$, and uncorrelated with the $r_k[n]$ sequences in the other layers. The sequence $r_k[n]$ is shared by all $S_{k,r}$ within the same layer. Notice that the algorithm of Figure 4 is such that

$$s_{k,r}[n] = \begin{cases} 0 & \text{if } x_{k,r}[n] \text{ is even;} \\ -1 \text{ or } 1 & \text{if } x_{k,r}[n] \text{ is odd;} \end{cases} \quad (6)$$

independent of which version of the hard limiter is used.

2.2 Algorithm Implementations

Figure 5 shows the implementation of the switching block of Figure 3, and is essentially the *node logic* presented in [6]. Although considered here in the context of first-order mismatch-shaping, this implementation of $S_{k,r}$ is valid for *any* mismatch-shaping algorithm that involves $s_k[n]$ sequences according to (6), as will be shown in the following section.

As apparent from Figure 5, only the two least significant bits (LSBs) of the input, $x_{k,r}^{(0)}[n]$ and $x_{k,r}^{(1)}[n]$, are affected by the processing, whereas the remaining $k-1$ bits directly form the $k-1$ MSBs of the two $S_{k,r}$ outputs. The LSBs of the two outputs, $x_{k-1,2r-1}^{(0)}[n]$ and $x_{k-1,2r}^{(0)}[n]$, are formed by the outputs of the upper and lower OR-gates, respectively. As indicated by the logic, and as explained in detail the following section, the numerical interpretation of $x_{k,r}[n]$ is such that $o_{k,r}[n]$ can be formed by simply XOR-ing the two LSBs of $x_{k,r}[n]$. Figure 5 indicates the presence of additional logic, referred to as *sequencing logic* (SL). The SL circuitry determines which of the two versions of the mismatch-shaping algorithm is implemented. Figure 6 shows the SL circuitry when implementing the version of the algorithm employing a hard limiter that operates according to (4). The signal 'CLK' denotes a clock signal that advances the D flip-flop when the input is odd. It follows from Figure 4 and Figure 5 that the implementation of the mismatch-shaping algorithm for the 3-bit example DAC requires merely 7 D flip-flops and 49 logic gates.

Figure 7 shows typical performance of the mismatch-shaping DAC when the sequencing logic is implemented according to Figure 6. The figure shows the PSD of the DAC noise when employing the unit DAC elements used for the data in Figure 2b and driving the $\Delta\Sigma$ADC by a sinusoidal input. As suggested previously, first order mismatch-shaping is provided, but a considerable number of spurs reside in the signal-band.

A dithering scheme corresponding to employing a hard limiter that operates according to (5) can be included when the sequencing logic is implemented according to Figure 8. In this figure, $r_k[n]$ denotes a white bit-sequence, uncorrelated with the $r_k[n]$ sequences of the other layers. A count of the total hardware usage for the 3-bit example DAC results in 14 D flip-flops, 70 logic gates, plus modest logic required to generate 3 white, uncorrelated bit-sequences [7]. Figure 9 shows the PSD of the DAC noise when employing the unit DAC elements used for the data in Figure 2b. The achieved conversion precision is 14.8 bits, a considerable improvement over the performance of the $\Delta\Sigma$ADC employing the conventional DAC.

3. COMPLEXITY-REDUCING PROPERTIES

This section presents a claim and proof that the DAC architecture presented in [6] represents a special case of the DAC presented in [4]. Also, the hardware efficient implementation of $S_{k,r}$ depicted in Figure 5 is verified.

First, a formal definition of the numerical value of the input and outputs of $S_{k,r}$ is required.

Definition: The numerical value of the $k+1$-bit sequence $x_{k,r}[n]$ is interpreted according to

$$x_{k,r}[n] = \sum_{i=1}^{k} 2^{i-1} x_{k,r}^{(i)}[n] + x_{k,r}^{(0)}[n]. \quad (7)$$

Thus, $x_{k,r}[n]$ is interpreted as the sum of a conventional k-bit unsigned integer and an "extra LSB", $x_{k,r}^{(0)}[n]$. This interpretation of $x_{k,r}[n]$ was first used in [8].

Claim 1: When employing *any* mismatch-shaping algorithm for which $s_{k,r}[n]$ satisfies (6) and for which $x_{k,r}[n]$ is interpreted as in (7), the signal processing performed by $S_{k,r}$ only affects $x_{k,r}^{(1)}[n]$ and $x_{k,r}^{(0)}[n]$.

Proof: From the similarity of (1) and (2), it suffices to show the claim for only one output, e.g., $x_{k-1,2r-1}[n]$.

In particular, it will be shown that bits $x^{(1)}_{k-1,2r-1}[n]$ through $x^{(k-1)}_{k-1,2r-1}[n]$ are copies of bits $x^{(2)}_{k,r}[n]$ through $x^{(k)}_{k,r}[n]$, respectively, and that $x^{(0)}_{k-1,2r-1}[n]$ is a binary function of $x^{(0)}_{k,r}[n]$, $x^{(1)}_{k,r}[n]$, and $s_{k,r}[n]$.

First notice that it follows from the discussion of the switching block signal processing in [4] that if $s_{k,r}[n]$ satisfies (6) then $S_{k,r}$ adheres to the *number conservation rule*. This ensures that, in Figure 1,

$$x_1[n] + x_2[n] + \cdots + x_8[n] = x[n],$$

i.e., $y[n] = x[n]$ if the unit DAC-elements exhibit ideal behavior.

It follows from (1) and (7) that

$$x_{k-1,2r-1}[n] = \frac{1}{2}\Big(\sum_{i=1}^{k} 2^{i-1} x^{(i)}_{k,r}[n] + x^{(0)}_{k,r}[n] + s_{k,r}[n]\Big)$$

$$= \sum_{i=2}^{k} 2^{i-2} x^{(i)}_{k,r}[n] + f_{k,r}[n], \qquad (8)$$

where

$$f_{k,r}[n] = \frac{1}{2}(x^{(1)}_{k,r}[n] + x^{(0)}_{k,r}[n] + s_{k,r}[n]).$$

If $x_{k,r}[n]$ is even, then $s_{k,r}[n] = 0$, and

$$f_{k,r}[n] = \begin{cases} 0 & \text{if } x^{(0)}_{k,r}[n] = x^{(1)}_{k,r}[n] = 0, \\ 1 & \text{if } x^{(0)}_{k,r}[n] = x^{(1)}_{k,r}[n] = 1. \end{cases} \qquad (9)$$

Alternatively, if $x_{k,r}[n]$ is odd, then

$$f_{k,r}[n] = \begin{cases} 0 & \text{if } s_{k,r}[n] = -1, \\ 1 & \text{if } s_{k,r}[n] = 1. \end{cases} \qquad (10)$$

Thus, it follows from (7) and the summation term in (8) that

$$x^{(i)}_{k-1,2r-1}[n] = x^{(i+1)}_{k,r}[n], \quad i = 1 \ldots k-1, \qquad (11)$$

and, by assigning $x^{(0)}_{k-1,2r-1}[n]$ to $f_{k,r}[n]$t in (8), it follows from (9) and (10) that $x^{(0)}_{k-1,2r-1}[n]$ is a binary function of $x^{(0)}_{k,r}[n]$, $x^{(1)}_{k,r}[n]$, and $s_{k,r}[n]$. ■

By assigning the input bits $x^{(1)}[n]$ through $x^{(3)}[n]$ to $x^{(1)}_{3,1}[n]$ through $x^{(3)}_{3,1}[n]$, respectively, and assigning a zero to $x^{(0)}_{3,1}[n]$ as indicated in Figure 1, it follows from (11) that $x^{(j)}[n]$ is not used in the processing until layer $4-j$. Thus, for this special case of the architecture proposed in [4], bit $x^{(j)}[n]$ need not be introduced until layer $4-j$. This corresponds to the figures shown in [6]. As an aside, notice that by applying a white $s_{k,r}[n]$ sequence, this structure can be used to *whiten* the DAC noise, rather than impose spectral shaping. An application for this type of DAC is direct digital synthesis.

Claim 2: The logic shown in Figure 5 implements the signal processing performed by $S_{k,r}$ of Figure 3 for both versions of the sequencing logic.

Proof: Again, only $x_{k-1,2r-1}[n]$ need be considered. Clearly, Figure 5 implements (11). To show that $x^{(0)}_{k-1,2r-1}[n]$ is formed appropriately, notice first that it follows from (3) and (7) that $o_{k,r}[n]$ can be formed by XOR-ing $x^{(1)}_{k,r}[n]$ and $x^{(0)}_{k,r}[n]$. It is easy to verify from Figure 6 and Figure 8 that $q_{k,r}[n] = 1$ corresponds to $s_{k,r}[n] = 1$ in Figure 4, and that $q_{k,r}[n] = 0$ corresponds to $s_{k,r}[n] = -1$. For the case where $x^{(0)}_{k,r}[n] = x^{(1)}_{k,r}[n]$, it follows from (9) that $x^{(0)}_{k-1,2r-1}[n]$ can be formed by AND-ing $x^{(0)}_{k,r}[n]$ and $x^{(1)}_{k,r}[n]$. This function is implemented by the upper AND-gate in Figure 5. For the case where $x^{(0)}_{k,r}[n] \neq x^{(1)}_{k,r}[n]$, it follows from (10) that $x^{(0)}_{k-1,2r-1}[n] = q_{k,r}[n]$. This function is implemented by the XOR-gate and the middle AND-gate. ■

4. CONCLUSION

Highly hardware efficient digital structures that implement the first order mismatch-shaping algorithms first proposed in [5] have been presented and a 3-bit example considered in detail.

Claims and proofs of hardware-reducing properties of the algorithms have been presented and used for the 3-bit example to devise an implementation of the basic mismatch-shaping algorithm using merely 7 flip-flops and 49 logic gates.

The inclusion of a dithering scheme to eliminate spurs in the mismatch-shaping has been presented and was shown to result in a total hardware count of 14 D flip-flops, 70 logic gates, plus logic to generate 3 white, uncorrelated bit-sequences.

REFERENCES

1. R. W. Adams, T. W. Kwan, "Data-directed scrambler for multi-bit noise shaping D/A converters," U. S. Patent No. 5,404,142, Apr. 4, 1995.
2. R. Schreier, B. Zhang, "Noise-shaped multibit D/A converter employing unit elements," *Electronic Letters*, vol. 31, no. 20, pp. 1712-1713, Sept. 1995
3. R. T. Baird, T. S. Fiez, "Linearity enhancement of multibit $\Delta\Sigma$ A/D and D/A converters using data weighted averaging," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 12, pp. 753-762, Dec. 1995.
4. I. Galton, "Spectral shaping of circuit errors in digital-to-analog converters," *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, no. 10, pp. 808-817, Oct. 1997.
5. I. Galton, "A hardware-efficient noise-shaping D/A converter," *Proceedings of the IEEE International Symposium on Circuits and Systems*, May, 1996.
6. A. Keady, C. Lyden, "Tree structure for mismatch noise-shaping multibit DAC," *Electronic Letters*, vol. 33, no. 17, pp. 1431-1432, Aug. 1997.
7. E. J. McCluskey, "Logic Design Principles," Englewood Cliffs, NJ: Prentice Hall, Inc., 1986.
8. H. T. Jensen, I. Galton, "A hardware-efficient DAC for direct digital synthesis," *Proceedings of the IEEE International Symposium on Circuits and Systems*, May, 1996.

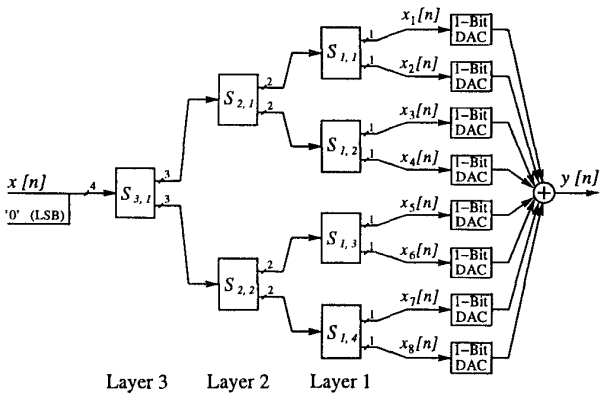

Figure 1: The example 3-bit version of the mismatch-shaping DAC.

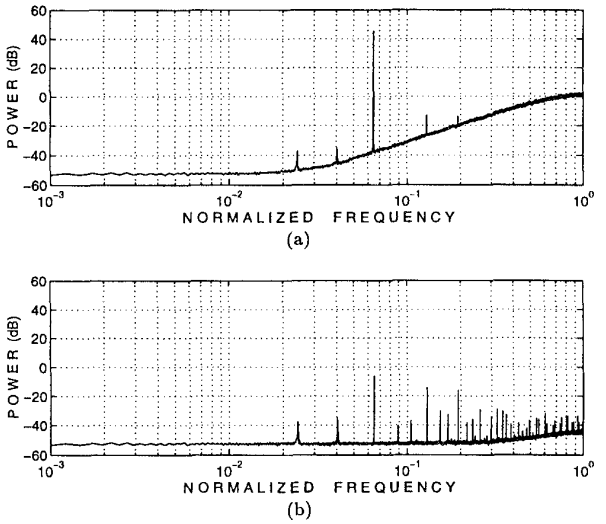

Figure 2: Simulated PSDs of (a) the example 2nd order $\Delta\Sigma$ADC output, and (b) the DAC noise of the example 3-bit conventional DAC.

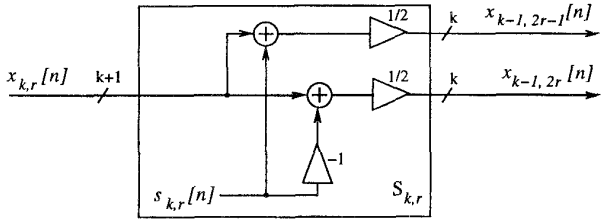

Figure 3: The switching block $S_{k,r}$.

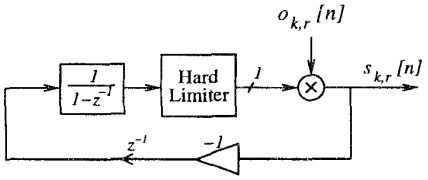

Figure 4: Generation of the switching sequence $s_{k,r}[n]$.

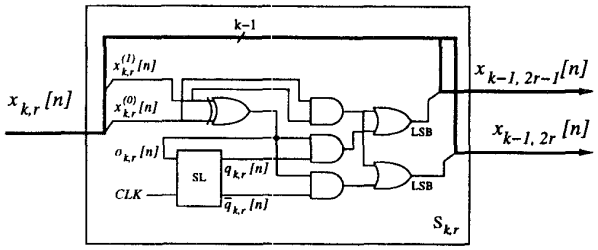

Figure 5: Implementation of the switching block $S_{k,r}$.

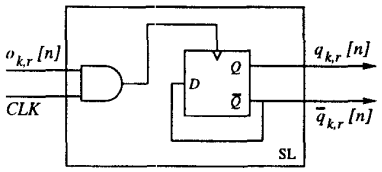

Figure 6: Implementation of SL corresponding to Figure 4 with a hard-limiter that operates according to (4).

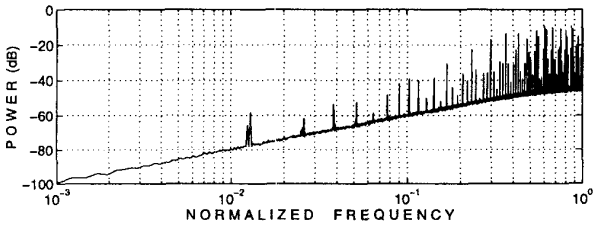

Figure 7: Simulated PSD of the DAC noise resulting from implementing SL as shown in Figure 6.

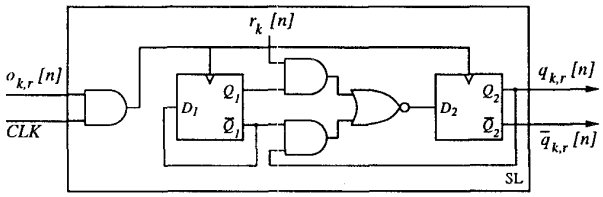

Figure 8: Implementation of SL corresponding to Figure 4 with a hard-limiter that operates according to (5).

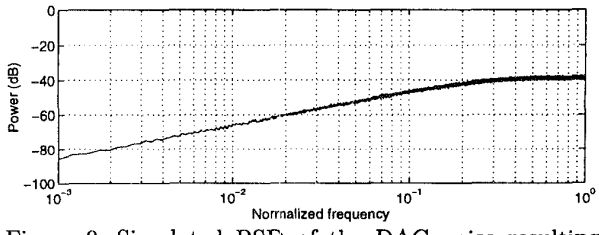

Figure 9: Simulated PSD of the DAC noise resulting from implementing SL as shown in Figure 8.