A PARALLEL DIGITAL ARCHITECTURE FOR DELTA-SIGMA MODULATION

Dan P. Scholnik
scholnik@nrl.navy.mil

Naval Research Laboratory, Washington, DC

***Abstract*— A major drawback of delta-sigma modulation is the high oversampling ratios required, especially for single-bit quantization. Accordingly, much of the research in the area has focused on lowering the sampling rate through various parallelization approaches. However, this research has been overwhelmingly concentrated on continuous and discrete-time analog modulator implementations for A/D converters, and not on reducing the critical path in a digital implementation for D/A conversion. In this paper the popular time-interleaved modulator is paired with a vector quantizer implementation of a finite-length modulator to form a parallel implementation of a delta-sigma DAC with a reduced critical path.**

1 INTRODUCTION

Delta-sigma ($\Delta\Sigma$) modulation has become the method of choice for high-resolution A/D and D/A conversion. By using feedback to shape the errors resulting from a high-speed, low-resolution (even single-bit) quantizer, better signal-to-noise ratios and linearity can be achieved than with conventional converters. Unfortunately, the oversampling required results in low available bandwidths for a fixed clock rate. Since many applications await high-resolution, high-bandwidth converters, there is great interest in increasing the effective bandwidth (or lowering the required sampling rate) of $\Delta\Sigma$ converters. To this end, two largely orthogonal approaches have been taken. The first is to increase the resolution of the quantizer in the loop, which reduces the oversampling requirement. The other, and the focus of this paper, is to parallelize the problem and increase the effective clock rate by computing many outputs in parallel rather than increasing the actual clock rate. Techniques that fall under this category include frequency channelization, code-division multiplexing [1], and time interleaving [2, 3, 4, 5]. Typically these have been applied to analog modulators in ADCs. The DAC counterparts of the frequency-channelized and code-division-multiplexed architectures require modulating and/or summing the (analog) outputs of several lower-speed $\Delta\Sigma$ modulators, complicating the system and introducing mismatch errors. The time-interleaved approach taken here is convenient in that it can be implemented entirely digitally.

The focus of this paper is to extend the time-interleaved approach to parallelize a digital $\Delta\Sigma$ modulator for DAC applications. The existing time-interleaved architectures cited all focus on discrete-time analog implementations for A/D applications. However, they do not fully solve the throughout problem in digital modulator implementations in that the critical computation path around the loop grows linearly with the number of outputs computed in parallel. In digital applications this results in a reduced clock rate, offsetting the gains from parallelization. In the sequel, it is shown that by considering the time-interleaved modulator to be a special case of vector $\Delta\Sigma$ modulation and by deriving a suitable vector quantizer implementation, the critical path can be reduced.

This work was supported by the Office of Naval Research.

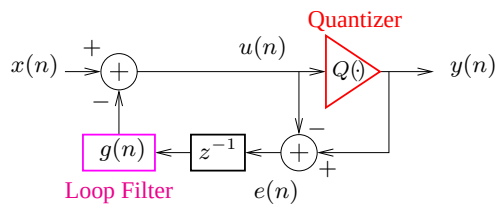

Figure 1: Error-feedback $\Delta\Sigma$ architecture.

2 PARALLELIZING THE LOOP FILTER

We start by reviewing the conventional $\Delta\Sigma$ modulator as it is typically realized in DSP. The *error-feedback* $\Delta\Sigma$ modulator topology is shown in Fig. 1, although the following analysis generally applies to other topologies that include a single filter somewhere in the loop. In the error feedback topology the error from the low-resolution quantizer is explicitly modeled as an additive noise source $e(n)$ which is fed back, filtered by a narrowband one-step-ahead predictor with impulse response $g(n)$, and subtracted from the input signal $x(n)$. The input is a real bandlimited sequence and the output $y(n) = Q\big(u(n)\big)$ is a real quantized sequence at the same rate. Input and output are related by

$$y(n) = x(n) + (h \circledast e)(n), \tag{1}$$

where sequence $e(n) = y(n) - u(n)$ is the error introduced by the quantizer and $h(n) \triangleq \delta(n) - g(n-1)$. The z-transform of $h(n)$, $H(z) = 1 - z^{-1}G(z)$, is known as the *noise transfer function* (NTF) since $Y(z) = X(z) + H(z)E(z)$. To be computable, there must be a net delay around the feedback loop, which is satisfied when $g(n)$ is causal. Subject to this structural constraint and some stability constraint on the peak gain, the NTF is typically designed to minimize the in-band quantization errors assuming they are white.

The combination of the nonlinear quantizer and the feedback loop complicates digital implementations of the modulator. In general the present output depends on the previous output through feedback, and thus pipelining techniques used in strictly feedforward systems don't apply. Techniques developed for restructuring IIR filters to improve throughput depend heavily on the linear nature of the filters, and cannot be applied because of the quantizer. Thus throughput is limited by the *critical path*, which is the longest computational distance between two delay registers. In Fig. 1 the critical path consists of two additions, the quantizer, and the internal critical path of

the loop filter, all of which must be computed in a single sample interval.

To improve throughput, we can consider computing N modulator outputs in parallel. Let

$$\mathbf{x}(n) \triangleq \left[x(nN), x(nN+1), \ldots, x(nN+N-1)\right]^T$$

be the $N \times 1$ vector of the next N inputs, and similarly define $\mathbf{y}(n)$, $\mathbf{u}(n)$, and $\mathbf{e}(n)$. This leads to the vector $\Delta\Sigma$ modulator architecture of Fig. 2, in which the scalar signals of Fig. 1 have been replaced with their vector representations, the scalar quantizer has become an elementwise vector quantizer, and the loop filter now has the matrix impulse response

$$\mathbf{g}(n) = \begin{pmatrix} g_{N-1}(n-1) & g_{N-2}(n-1) & \ldots & g_0(n-1) \\ g_0(n) & g_{N-1}(n-1) & \ldots & g_1(n-1) \\ \vdots & \vdots & \ddots & \vdots \\ g_{N-2}(n) & g_{N-3}(n) & \ldots & g_{N-1}(n-1) \end{pmatrix}$$

where $g_k(n) \triangleq g(nN+k)$ is the kth polyphase component of the sequence $g(n)$. Save a reversal of the vector index and the absence of explicit serial-to-parallel converters, this is exactly the block filtering representation given in [6] and the overall modulator architecture is just the error-feedback version of the time-interleaved modulator of [2]. It is also a special case of the vector $\Delta\Sigma$ modulator introduced in [7] with a loop filter that is *pseudo-circulant*, which is the matrix structure required to ensure that the block filtering is LTI. The vector output relationship is

$$\mathbf{y}(n) = \mathbf{x}(n) + (\mathbf{h} \circledast \mathbf{e})(n), \tag{2}$$

where $\mathbf{h}(n) \triangleq \delta(n)\mathbf{I} - \mathbf{g}(n)$ and matrix-vector convolution is defined as $(\mathbf{h} \circledast \mathbf{e})(n) \triangleq \sum_k \mathbf{h}(k)\mathbf{e}(n-k)$.

What at first appears to be a successful parallelization of the $\Delta\Sigma$ modulator is, in terms of a digital implementation, just a rearrangement of the calculation. Recall the computability requirement in the scalar modulator that there be a net delay around the loop, a condition that is no longer satisfied in the vector loop. The explicit delay of the scalar modulator has been folded into the causal matrix impulse response $\mathbf{g}(n)$. The equivalent new computability constraint would be $\mathbf{g}(0) = \mathbf{0}$, so that $\mathbf{y}(n)$ depends only on $\mathbf{y}(n-1), \mathbf{y}(n-2), \ldots$ making a parallel implementation straightforward. Simple inspection reveals, however, that $\mathbf{g}(0)$ is not zero but the lower-triangular Toeplitz matrix

$$\mathbf{g}_0 \triangleq \mathbf{g}(0) = \begin{pmatrix} 0 & 0 & \ldots & 0 \\ g(0) & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g(N-2) & g(N-3) & \ldots & 0 \end{pmatrix}. \tag{3}$$

Because there is no net delay in the impulse response components below the diagonal, all elements of the output vector cannot be computed at the same time. Rather, they must be computed sequentially, from the first vector element y_1 to the last y_N, as each depends on the preceding output elements. The critical path is the same as for the scalar system, and the result is a scalar system masquerading as a vector one.[1]

[1] In [4] it is suggested that pipelining and clock staggering can partially overcome this in VLSI, although this is generally not applicable to FPGAs.

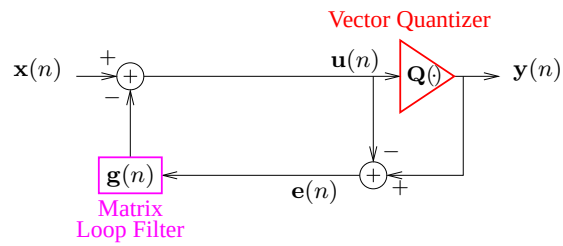

Figure 2: Vector $\Delta\Sigma$ architecture.

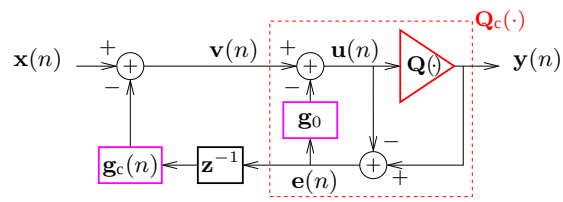

Figure 3: Rearranged vector $\Delta\Sigma$ architecture. The box contains the memoryless part of the system computation.

The next step to obtaining a true parallel implementation is to separate the loop filter impulse response into strictly causal and $n = 0$ components, so that $\mathbf{g}(n) = \delta(n)\mathbf{g}_0 + \mathbf{g}_c(n-1)$. Substituting into (2) and rearranging yields

$$\mathbf{y}(n) = \mathbf{x}(n) - (\mathbf{g}_c \circledast \mathbf{e})(n-1) + \mathbf{h}_0\mathbf{e}(n), \tag{4}$$

where $\mathbf{h}_0 \triangleq \mathbf{I} - \mathbf{g}_0$. A realization of this is shown in Fig. 3 with the memoryless part of the computation grouped within the box to separate it from the net-delay loop filter computation. The memoryless system within the box forms a new vector quantizer, with the output relation

$$\mathbf{y} = \mathbf{Q}_c(\mathbf{v}) = \mathbf{v} + \mathbf{h}_0\mathbf{e} \tag{5}$$

or, componentwise for $k = 1, \ldots, N$, using (3) and Fig. 3,

$$\begin{aligned} y_k = [\mathbf{Q}_c(\mathbf{v})]_k &= Q\left(v_k - \sum_{n=1}^{k-1} g(n-1)e_{k-n}\right) \\ &= v_k + \sum_{n=0}^{k-1} h(n)e_{k-n}. \end{aligned} \tag{6}$$

The similarity between the defining relationship (1) of a scalar $\Delta\Sigma$ modulator and (6) suggests defining the latter as the input/output relationship of a *finite-length* $\Delta\Sigma$ *modulator* with input, output, and error N-vectors $\mathbf{v}$, $\mathbf{y}$, and $\mathbf{e}$ respectively.

The architecture of Fig. 3 is inconvenient as it feeds back the internal quantizer error $\mathbf{e}$ rather than the error $\mathbf{e}_c \triangleq \mathbf{Q}_c(\mathbf{v}) - \mathbf{v}$ due to the finite-length $\Delta\Sigma$ vector quantizer. By (5) the two errors are related through a coordinate transformation, $\mathbf{e}_c = \mathbf{h}_0\mathbf{e}$. Defining $\mathbf{h}_c(n) \triangleq \delta(n)\mathbf{I} - \mathbf{g}_c(n-1)\mathbf{h}_0^{-1}$ and rewriting (2) yields

$$\mathbf{y}(n) = \mathbf{x}(n) + (\mathbf{h}_c \circledast \mathbf{e}_c)(n), \tag{7}$$

which is the error-feedback output equation for the vector $\Delta\Sigma$ modulator with loop-filter impulse response $\mathbf{g}_c(n)\mathbf{h}_0^{-1}$ shown in Fig. 4. Comparing Figs. 2 and 4 reveals that the goal of removing zero-delay paths around the loop has been achieved, at the price of a more

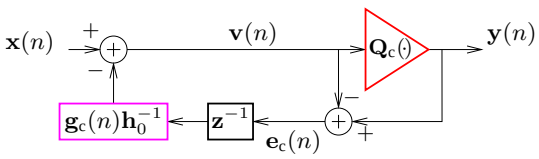

Figure 4: Equivalent causal vector $\Delta\Sigma$ architecture.

complex vector quantizer. To realize the promised speed benefits a parallel implementation for this quantizer must be found, which is the topic of the next section.

3 PARALLELIZING THE QUANTIZER

The vector quantizer implementation suggested by (6) still depends on a sequential computation across the vector indices, and thus has a critical path that is linear in N. In this section a nonsequential implementation of the vector quantizer is derived that has a nearly constant critical path and is more suitable for a real-time system.

Start by rewriting the output relation (5) in terms of only the quantizer input $\mathbf{v}$ and output $\mathbf{y}$. Directly from Fig. 3 we obtain $\mathbf{u} = \mathbf{v} + \mathbf{g}_0\mathbf{u} - \mathbf{g}_0\mathbf{y}$. Solving for $\mathbf{u}$ results in

$$\begin{aligned}\mathbf{y} = \mathbf{Q}_c(\mathbf{v}) &= \mathbf{Q}(\mathbf{u}) \\ &= \mathbf{Q}\big(\mathbf{h}_0^{-1}\mathbf{v} - (\mathbf{h}_0^{-1} - \mathbf{I})\mathbf{y}\big).\end{aligned} \tag{8}$$

Since $\mathbf{h}_0$ is lower triangular and Toeplitz with ones on the diagonal, so is $\mathbf{h}_0^{-1}$. Thus (8) can be written recursively as

$$y_k = [\mathbf{Q}_c(\mathbf{v})]_k = Q\left([\mathbf{h}_0^{-1}\mathbf{v}]_k - \sum_{n=1}^{k-1}[\mathbf{h}_0^{-1}]_{k,n}y_n\right). \tag{9}$$

Now, two observations. First, for each vector index k, y_k depends only on $y_1, \ldots, y_{k-1}$ and $\mathbf{v}$. Second, the set of all possible output vectors $\mathbf{y}$ forms a finite constellation of M^N points for an M-level scalar quantizer.

For simplicity a binary scalar quantizer is now assumed with constellation $\{-1, 1\}$ such that $Q(u) = \mathrm{sgn}(u)$ and the scalar decision regions are the negative and positive half lines. The decision regions for general N-space can be derived through a simple inductive argument based on (9). First, the decision regions for y_1 are just the half spaces corresponding to $v_1 > 0$ and $v_1 < 0$. The decision regions for $y_1, \ldots, y_k$ are then formed by splitting each of the decision regions for $y_1, \ldots, y_{k-1}$ by the region-dependent offset hyperplane in $\mathbf{v}$ defined by $[\mathbf{h}_0^{-1}\mathbf{v}]_k = \sum_{n=1}^{k-1}[\mathbf{h}_0^{-1}]_{k,n}y_n$. If the decision regions for $y_1, \ldots, y_{k-1}$ are each the intersection of $k-1$ half spaces, then the decision regions for $y_1, \ldots, y_k$ will be the intersection of k half spaces. By induction, the overall decision regions are each the intersection of N half spaces. This structure was previously observed in [8] for a restricted class of impulse responses $g(n)$.

For a visual example consider $N = 2$, where (9) reduces to

$$y_1 = \mathrm{sgn}(v_1) \tag{10}$$

$$y_2 = \begin{cases}\mathrm{sgn}\big(v_2 + [\mathbf{h}_0^{-1}]_{2,1}v_1 + [\mathbf{h}_0^{-1}]_{2,1}\big), & y_1 = -1 \\ \mathrm{sgn}\big(v_2 + [\mathbf{h}_0^{-1}]_{2,1}v_1 - [\mathbf{h}_0^{-1}]_{2,1}\big), & y_1 = 1\end{cases} \tag{11}$$

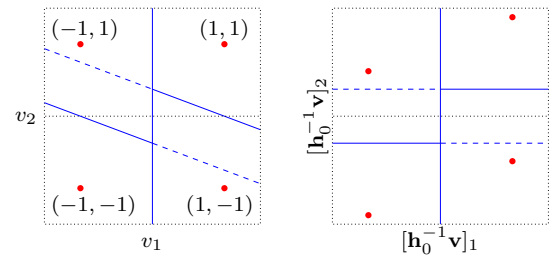

Figure 5: Decision regions for the two-dimensional $\Delta\Sigma$ quantizer before (left) and after (right) coordinate transformation.

and the decision boundaries are

$$\begin{aligned}v_1 &= 0 \\ v_2 &= -[\mathbf{h}_0^{-1}]_{2,1}v_1 - [\mathbf{h}_0^{-1}]_{2,1}, \qquad v_1 < 0 \\ v_2 &= -[\mathbf{h}_0^{-1}]_{2,1}v_1 + [\mathbf{h}_0^{-1}]_{2,1}, \qquad v_1 > 0\end{aligned}$$

as shown by the solid lines of Fig. 5 (left). If we instead consider the decision regions in terms of the transformed variable $\mathbf{h}_0^{-1}\mathbf{v}$, then each decision boundary becomes perpendicular to one of the axes as shown in Fig. 5 (right). This last result holds in any dimension N.

Having determined the vector-quantizer geometry, we now derive a parallel computational structure to implement the quantizer. There are (at least) two related approaches. The first is to consider each possible value of $\mathbf{y}$ independently to determine if it is the correct output. The second is to quantize each element of $\mathbf{y}$ independently. Both are described in turn.

The first approach follows directly from (8), as one (and only one) of the 2^N possible values of $\mathbf{y}$ satisfies the equality. Thus we can test all possible values in parallel, and choose only the matching value of $\mathbf{y}$. Enumerate all the possible values of $\mathbf{y}$ as the columns of

$$\mathbf{Y} = \begin{pmatrix} -1 & -1 & -1 & & 1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ -1 & -1 & 1 & & 1 \\ -1 & 1 & -1 & & 1 \end{pmatrix}.$$

All of the decision boundary offsets can be precomputed as $(\mathbf{h}_0^{-1} - \mathbf{I})\mathbf{Y}$. While symbolically convenient, this expression is redundant; due to the lower-triangular nature of $(\mathbf{h}_0^{-1} - \mathbf{I})$ the top row of the product contains only zeros, the second row only two unique values, the third row four unique values, and so on. Now, according to (8), we subtract each column of $(\mathbf{h}_0^{-1} - \mathbf{I})\mathbf{Y}$ from the transformed input $\mathbf{h}_0^{-1}\mathbf{v}$ and quantize each result. This is equivalent to performing N comparisons, in agreement with our characterization of the decision regions as the intersection of N half spaces. The quantized vectors are then compared to the corresponding columns of $\mathbf{Y}$. In only one case will they match and this is the correct quantized output. By taking advantage of the aforementioned redundancies, the total number of comparisons (scalar quantizations) performed can be reduced from $N2^N$ to $2^N - 1$, in addition to the 2^N comparisons of the resulting quantized vectors with the columns of $\mathbf{Y}$. Although the number of subtract-quantize-compare blocks is exponential in N, the size of each block grows only linearly with N.

In the second vector-quantizer realization, each output element y_k is computed independently. Recall that by construction the decision regions for $y_1, \ldots, y_k$ are found by splitting each of the decision

regions for $y_1, \ldots, y_{k-1}$ with an offset hyperplane. In terms of the transformed variable $\mathbf{h}_0^{-1}\mathbf{v}$ the hyperplanes are all perpendicular to the kth coordinate axis and thus test only the kth coordinate. In the first dimension there is just the single test given by (10); label the result $a_{1,1}$ so that $y_1 = a_{1,1}$. In the second dimension there are two tests (11), the results of which we label $a_{2,1}$ and $a_{2,2}$. The table to determine y_2 from these tests is

$a_{1,1}$	$a_{2,1}$	$a_{2,2}$	y_1	y_2
-1	-1	X	-1	-1
-1	1	X	-1	1
1	X	-1	1	-1
1	X	1	1	1

In the kth dimension there are 2^{k-1} such tests to perform. To determine y_k we need these tests plus the tests from dimensions 1 through $k-1$, a total of $2^k - 1$ binary values. For small values of N (up to four or so), the output $\mathbf{y}$ can be evaluated directly by a single lookup table with the $\{a_{k,m}\}$ as the input. For larger N it may need to be calculated in stages. In contrast to the first method, here we have only N processing units (each lookup-table output column), but they grow exponentially in size with N. In fact, the test outputs $\{a_{k,m}\}$ are just the unique tests from the first method and so the difference between the two lies in how the set of tests is decoded to find $\mathbf{y}$.

Summing up, the procedure for a parallel implementation of the vector quantizer consists of three steps:

1. Multiply input $\mathbf{v}$ by $\mathbf{h}_0^{-1}$
2. Perform the $2^N - 1$ unique comparisons
3. Decode the results using the first or second method

Figure 6 shows the vector quantizer represented in this way. By rearranging Fig. 6 to push the multiplication by $\mathbf{h}_0^{-1}$ back through the system we arrive at the equivalent system of Fig. 7. Here the input $\mathbf{x}(n)$ is transformed by $\mathbf{h}_0^{-1}$ prior to the feedback loop, in order that it not contribute to the critical path. The fed-back output is also multiplied by $\mathbf{h}_0^{-1}$, but since $\mathbf{y}(n)$ is binary, this multiply can be efficiently merged into the decoder with no critical path penalty. The third change is that the loop-filter response is premultiplied rather than post-multiplied by $\mathbf{h}_0^{-1}$. The net result of all these changes is to move the high-resolution matrix multiply outside the loop. The critical path of this system consists of the block filtering, two additions, a comparison, and the decoder (a lookup table or similar). For comparison, with $N = 4$ the critical path of Fig. 3 consists of the block filtering, three multiplications (one for each trip around the loop within the dashed box), four comparisons (one for each element of $\mathbf{y}$), and between eight and eleven additions depending on the number of nonzero terms in $\mathbf{g}_0$.

4 CONCLUSIONS

In this paper the popular block-filtering approach to parallelization of the scalar $\Delta\Sigma$ modulator is modeled as a special case of vector $\Delta\Sigma$ modulation. The usual sequential implementation of the resulting vector quantizer [2, 4] leads to a long critical path and an inefficient digital realization. To reduce the critical path, two related parallel implementations are derived, each comprising only a matrix transformation (which can be moved outside the loop), a parallel set of scalar comparisons, and one or more decoder logic blocks. Due to the exponential growth of the number of required comparisons, this approach does not offer unlimited parallelism. For modest N, however, it provides a method of parallelization well suited (for example) for the upcoming generations of FPGAs with large, fast lookup tables. Although specifically derived for the scalar $\Delta\Sigma$ problem, this quantizer implementation has application to the more general class of vector $\Delta\Sigma$ modulators.

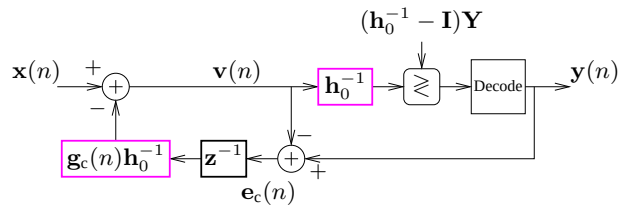

Figure 6: Vector $\Delta\Sigma$ modulator with parallel vector quantizer implementation.

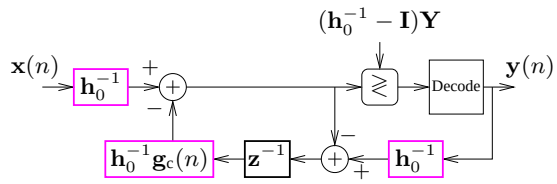

Figure 7: Equivalent vector modulator rearranged to reduce computation in the loop.

References

[1] I. Galton and H. T. Jensen, "Oversampling parallel delta-sigma modulator A/D conversion," *IEEE Trans. Circuits and Systems II*, vol. 43, no. 12, pp. 801–810, Dec. 1996.

[2] R. Khoini-Poorfard, L. B. Lim, and D. A Johns, "Time-interleaved oversampling A/D converters: Theory and practice," *IEEE Trans. Circuits and Systems II*, vol. 44, no. 8, pp. 634–645, Aug. 1997.

[3] A. Eshraghi and T. Fiez, "An area efficient time-interleaved parallel delta-sigma A/D converter," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS)*, 1998.

[4] M. Kozak, M. Karaman, and I. Kale, "Efficient architectures for time-interleaved oversampling delta-sigma converters," *IEEE Trans. Circuits and Systems II*, vol. 47, no. 8, pp. 802–810, Aug. 2000.

[5] X. Wang, W. Qin, and X. Ling, "Cascaded parallel oversampling sigma-delta modulators," *IEEE Trans. Circuits and Systems II*, vol. 47, no. 2, pp. 156–161, Feb. 2000.

[6] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[7] D. P. Scholnik and J. O. Coleman, "Vector delta-sigma modulation with integral shaping of hardware-mismatch errors," in *IEEE 2000 Int'l Symp. on Circuits and Systems (ISCAS 2000)*, Geneva, Switzerland, May 2000.

[8] N. Thao, "Vector quantization analysis of $\Sigma\Delta$ modulation," *IEEE Trans. Signal Processing*, vol. 44, no. 4, pp. 808–817, Apr. 1996.