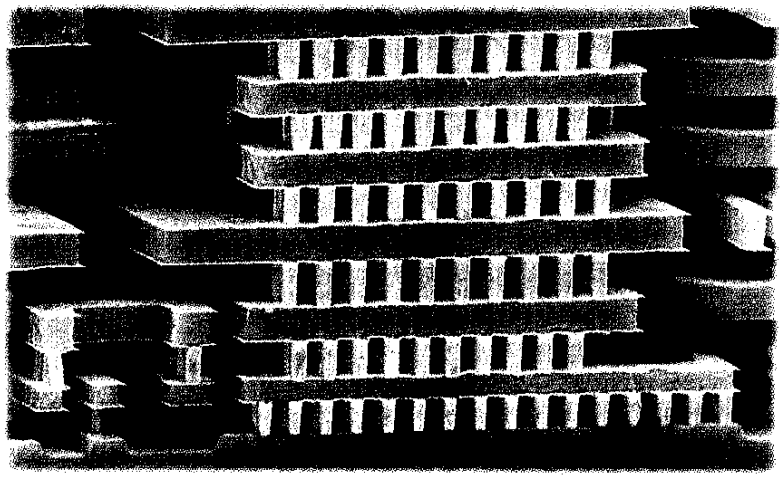


© Index Stock

FPGA Signal Processing Using **Sigma-Delta Modulation**

Innovative Combinations of Techiques and Hardware for System Designers

Chris Dick and Fred Harris

While ΣΔM techniques [3], [7] are applied widely in analog conversion sub-systems, both analog-to-digital (ADC) and digital-to-analog (DAC) converters, these methods have enjoyed much less exposure in the broader application domain, where flexible and configurable solutions, traditionally supplied via a software DSP (soft-DSP), are required. And this limited level of exposure is easy to understand. Most, if not all, of the efficiencies and optimizations afforded by ΣΔM are hardware-oriented and, therefore, cannot be capitalized upon in the fixed-precision predefined datapath found in a soft-DSP processor. This limitation, of course, does not exist in a *field programmable gate array* (FPGA) DSP solution. With FPGAs the designer has complete control of the silicon, able to implement any desired datapath and employ optimal word precisions in the system with the objective of producing a design that satisfies the specifications in the most economically sensitive manner.

While implementation of a digital ΣΔ *application-specific integrated circuit* (ASIC) is, of course, possible, economic constraints make the implementation of a building block that would provide the flexibility, and be generic enough to cover a broad market cross-section, impractical. FPGA-based hardware provides a solution to this problem. FPGAs are off-the-shelf commodity items that provide a silicon feature set ideal for constructing high-performance DSP systems. These devices maintain the flexibility of software-based solutions, while providing levels of performance that match, and often exceed ASIC solutions.

There is a rich and expanding body of literature devoted to the efficient and effective implementation of digital signal processors using FPGA-based hardware. More often than not, the most successful of these techniques involves a paradigm shift away from the methods that provide good solutions in software programmable DSP systems.

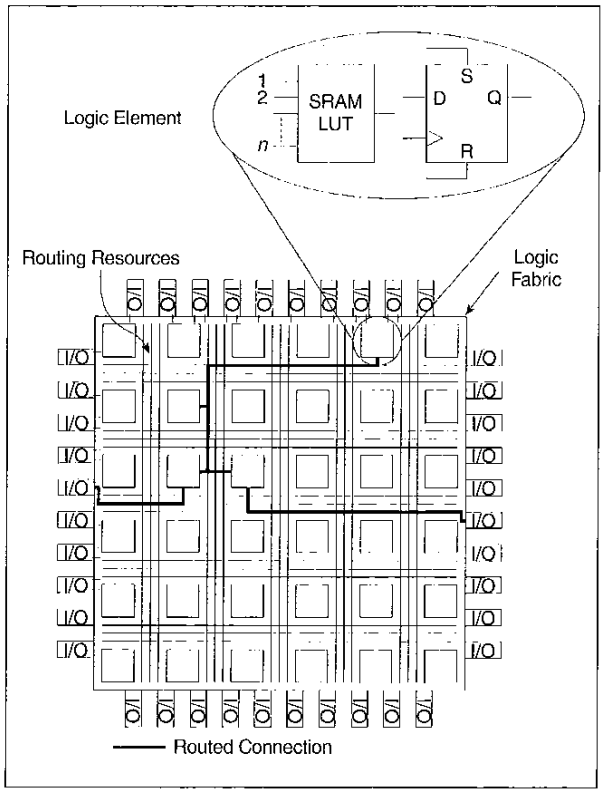
This article reports on the rich set of design opportunities that are available to the signal processing system designer through innovative combinations of ΣΔM techniques and FPGA signal processing hardware. The applications considered include narrow-band filters, both single-rate and multi-rate; dc canceler; and ΣΔM hybrid digital-analog control loops for simplifying carrier recovery, timing recovery, *automatic gain control* (AGC) loops in a digital communication receiver.

FPGA Architecture

There is a rich range of FPGAs provided by many semiconductor vendors including Xilinx, Altera, Atmel, and AT&T. The architectural approaches are as diverse as there are manufacturers, but some generalizations can be made. Most of the devices are basically organized as an ar-

1053-5888/00/$10.00©2000IEEE

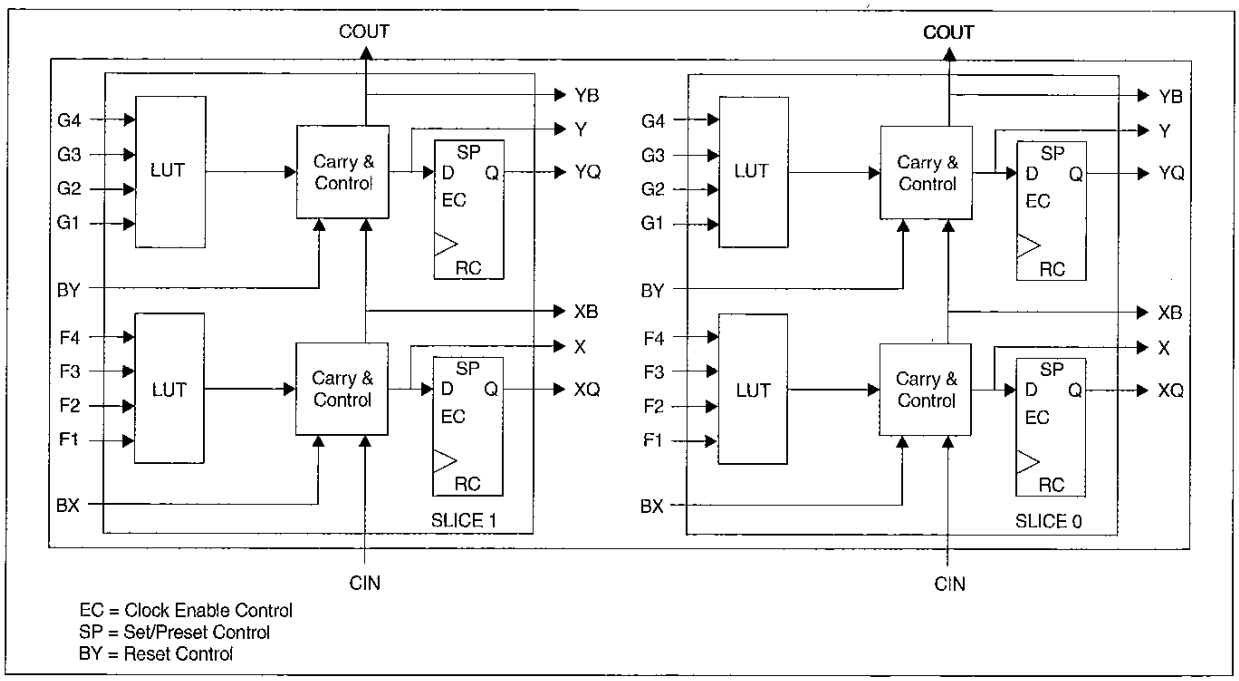
ray of logic elements and programmable routing resources used to provide the connectivity between the logic elements, FPGA I/O pins, and other resources such as on-chip memory. The structure and complexity of the logic elements, as well as the organization and functionality supported by the interconnection hierarchy, distinguish the devices from each other. Other features such as block memory and delay-locked-loop technology are also significant factors that influence the complexity and performance of an algorithm implemented using FPGAs.



▲ *Fig. 1. Generic FPGA architecture.*

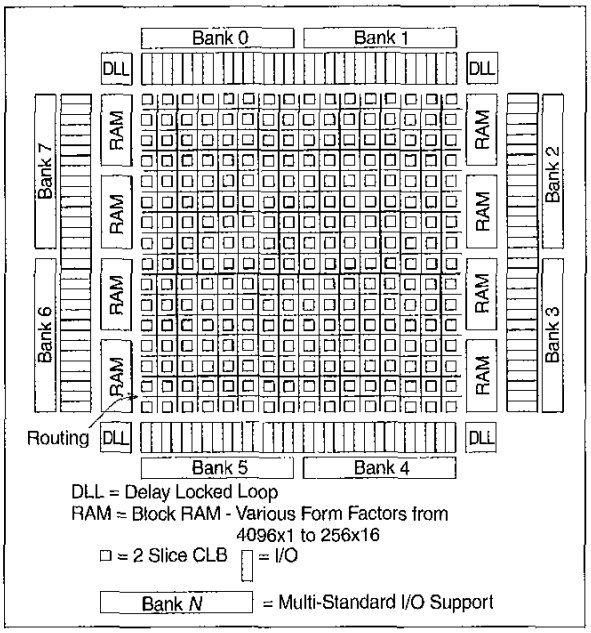
A logic element usually consists of one or more RAM-based n-input look-up tables (where n is between three and six) and one to several flip-flops. There may also be additional hardware support in each element to enable high-speed arithmetic operations. This generic FPGA architecture is shown in Fig. 1. Also illustrated in the figure (as bold lines) are several connections between logic elements and the device input/output (I/O) ports. Application-specific circuitry is supported in the device by downloading a bit stream into SRAM-based configuration memory. This personalization database defines the functionality of the logic elements, as well as the internal routing. Different applications are supported on the same FPGA hardware platform by configuring the FPGA(s) with appropriate bit streams.

As a specific example, consider the Xilinx Virtex series of FPGAs [12]. The logic elements, called *slices*, essentially comprise two four-input look-up tables (LUTs), two flip-flops, several multiplexers, and some additional silicon support that allows the efficient implementation of carry-chains for building high-speed adders, subtracters, and shift registers. Two slices form a configurable logic block (CLB) as shown in Fig. 2. The CLB is the basic tile used to build the logic matrix. Some FPGAs, like the Xilinx Virtex families, supply on-chip block RAM. Fig. 3 shows the CLB matrix that defines a Virtex FPGA. Current-generation Virtex silicon provides a family of devices offering 768 to 19,200 logic slices, and from eight to 160 variable-form factor block memories.



▲ *Fig. 2. Simplified Virtex CLB architecture.*

Xilinx XC4000 and Virtex [12] devices also allow the designer to use the logic element LUTs as memory—either ROM or RAM. Constructing memory with this distributed approach can yield access bandwidths in the many tens of gigabytes per second range.

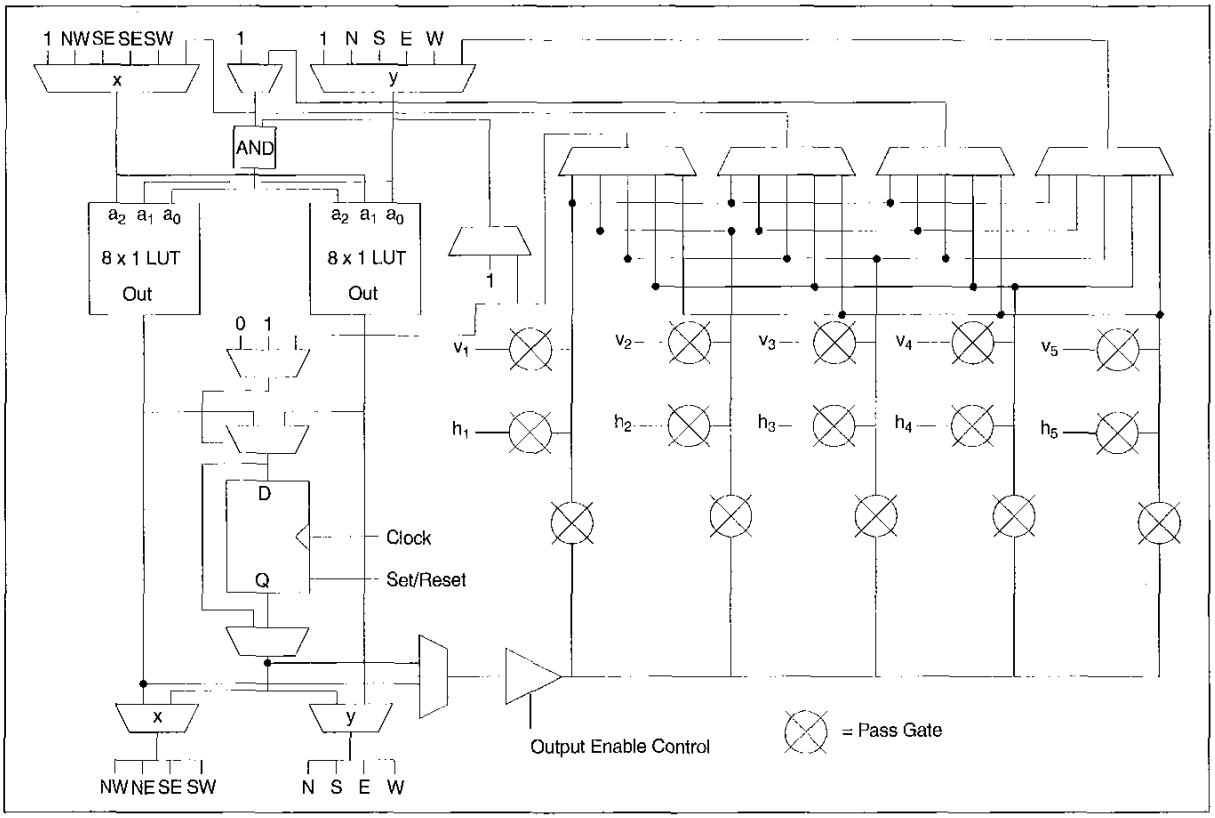


▲ *Fig. 3. Xilinx Virtex logic cell array architecture.*

Typical clock frequencies for current-generation devices are in the multiple tens of megahertz (100 to 200) range.

In contrast to the logic-slice architecture used in Xilinx Virtex devices, the logic block architecture employed in the Atmel AT40K [1] FPGA is shown in Fig. 4. Like the Xilinx device, combinational logic is realized using look-up tables. In this case, two three-input LUTs and a single flip-flop are available in each logic cell. The pass gates in a cell form part of the signal routing network, and are used for connecting signals to the multiple horizontal and vertical bus planes. In addition to the orthogonal routing resources, indicated as *N, S, E,* and *W* in Fig. 4, a diagonal group of interconnects (NW, NE, SE, and SW), associated with each cell *x* output, are available to provide efficient connections to a neighboring cell's *x* bus inputs.

The objective of the FPGA/DSP architect is to formulate algorithmic solutions for applications that best utilize FPGA resources to achieve the required functionality. This is a three-dimensional optimization problem in power, complexity, and bandwidth. The remainder of this article describes some novel FPGA solutions to several signal processing problems. The results are important in an industrial context because they enable either smaller, and hence, more economic, solutions to important problems, or allow more arithmetic compute power to be realized with a given area of silicon.



▲ *Fig. 4. Atmel AT40K logic cell architecture.*

ΣΔ Modulators, FIR Filters, and FPGAs

This section describes a method employing *sigma-delta* modulation (ΣΔM) techniques for implementing area-efficient finite impulse response (FIR) filters using FPGA hardware. Before treating the FPGA filter design, a brief review of ΣΔ modulation encoding is presented.

ΣΔ *Modulation*

ΣΔM is a source coding technique most prominently employed in ADCs and DACs. In this context, hybrid analog and digital circuits are used in the realization. Fig. 5 shows a single-loop ΣΔ modulator. Provided the input signal is *busy* enough, the linearized discrete time model of Fig. 6 can be used to illustrate the principle. In this figure, the 1-bit quantizer is modeled by an additive white noise source with variance $\sigma_e^2 = \Delta^2 / 12$, where Δ represents the quantization interval. The z-transform of the system is

$$Y(z) = \frac{H(z)}{1+H(z)} X(z) + \frac{1}{1+H(z)} Q(z) \tag{1}$$

$$= H_s(z)X(z) + H_n(z)Q(z), \tag{2}$$

where

$$H(z) = \frac{1}{z-1}, \tag{3}$$

and $H_s(z)$ and $H_n(z)$ are the signal and noise transfer functions (NTF) respectively. In a good ΣΔ modulator, $H_s(\omega)$ will have a flat frequency response in the interval $|f| \le B$. In contrast, $H_n(\omega)$ will have a high attenuation in the frequency band $|f| \le B$ and a "don't-care" region in the interval $B < |f| < f_s/2$. For the single loop ΣΔ in Fig. 6 $H_s(z) = z^{-1}$ and $H_n(z) = 1 - z^{-1}$. Thus, the input signal is not distorted in any way by the network and simply experiences a pure delay from input to output. The performance of the system is determined by the noise transfer function $H_n(z)$, which is given by

$$|H_n(f)| = 4\left|\sin\frac{\pi f}{f_s}\right|, \tag{4}$$

and is shown in Fig. 7. The in-band quantization noise variance is

$$\sigma_n^2 = \int_{-B}^{+B} |H_n(f)|^2 S_q(f)\,df, \tag{5}$$

where $S_q(f) = \sigma_q^2 / f_s$ is the power spectral density of the quantization noise. Observe that for a non-shaped noise (or white) spectrum, increasing the sampling rate by a factor of two, while keeping the bandwidth B fixed, reduces the quantization noise by 3 dB. For a first-order ΣΔM, it can be shown that

While the required dynamic range of a system fixes the number of bits required to represent the data, it also affects the expense of subsequent arithmetic operations, in particular, multiplications.

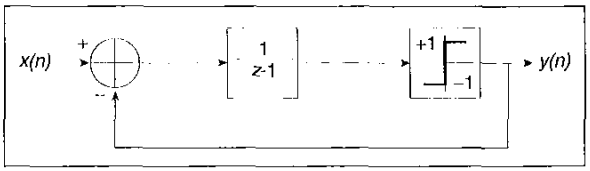
$$\sigma_n^2 \approx \frac{1}{3}\pi^2\sigma_q^2\left(\frac{2B}{f_s}\right)^3 \tag{6}$$

for $f_s >> 2B$. Under these conditions, doubling the sampling frequency reduces the noise power by 9 dB, of which 3 dB is due to the reduction in $S_q(f)$, and a further 6 dB is due to the filter characteristic $H_n(f)$. The noise power is reduced by increasing the sampling rate to spread the quantization noise over a large bandwidth, and then by shaping the power spectrum using an appropriate filter.

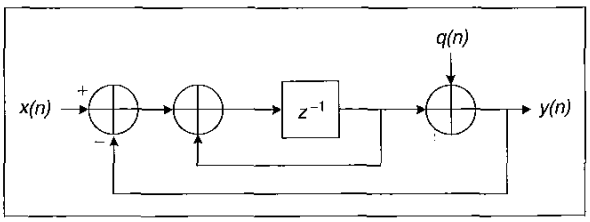
Reduced Complexity Filters Using ΣΔ *Modulation Techniques*

ΣΔM techniques can be employed for realizing area-efficient narrowband filters in FPGAs. These filters are utilized in many applications including narrow-band communication receivers, multichannel RF surveillance systems, and solutions to some spectrum management problems.

A uniform quantizer operating at the Nyquist rate is the standard solution to the problem of representing data within a specified dynamic range. Each additional bit of resolution in the quantizer provides an increase in dynamic range of approximately 6 dB. A signal with 60 dB of dynamic range requires 10 bits, while 16 bits can represent data with a dynamic range of 96 dB. While the re-

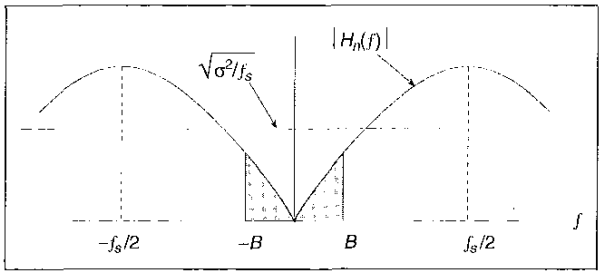


▲ *Fig. 5. Single loop ΣΔ modulator.*

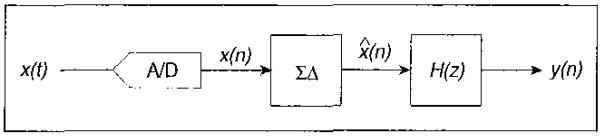


▲ *Fig. 6. Linearized model of ΣΔ modulator.*

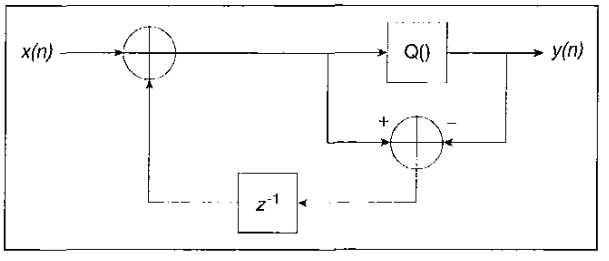
quired dynamic range of a system fixes the number of bits required to represent the data, it also affects the expense of subsequent arithmetic operations, in particular, multiplications. In any hardware implementation, and of course, this includes FPGA-based DSP processors, there are strong economic imperatives to minimize the number and complexity of the arithmetic components employed in the datapath. The proposal investigated in this section is to employ noise-shaping techniques to reduce the precision of the input data samples so that the complexity of the multiply-accumulate (MAC) units in the filter can be minimized. Of course, the preprocessing must not compromise the integrity of the signal in the band of interest. The net result is a reduction in the amount of FPGA logic resources required to realize the specified filter.



▲ *Fig. 7. Single loop sigma-delta modulator noise transfer function.*



▲ *Fig. 8. Reduced complexity FIR filtering employing a ΣΔ preprocessor.*



▲ *Fig. 9. Error feedback sigma-delta modulator.*

Consider the structure shown in Fig. 8. Instead of applying the quantized data $x(n)$ from the ADC directly to the filter, it will be preprocessed by a ΣΔ modulator.

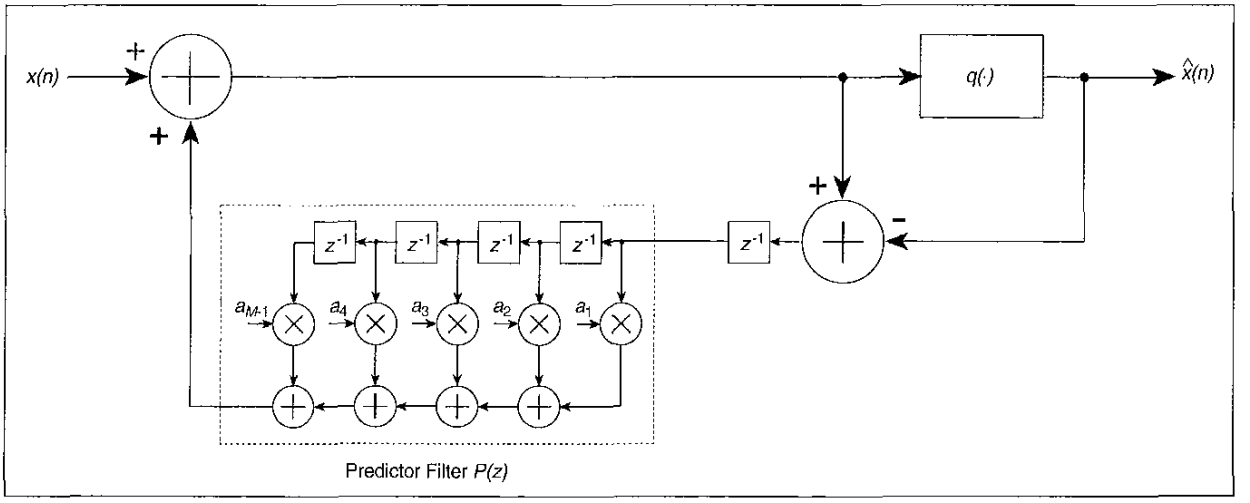
The requantized input samples $\hat{x}(n)$ are now represented using fewer bits per sample, permitting the subsequent filter $H(z)$ to employ reduced precision multipliers in the mechanization. The filter coefficients are still kept to a high precision.

The ΣΔ data requantizer is based on a single loop error feedback ΣΔ modulator [3] shown in Fig. 9. In this configuration, the difference between the quantizer input and output sample is a measure of the quantization error, which is fed back and combined with the next input sample. The error-feedback ΣΔ modulator operates on a highly oversampled input and uses the unit delay z^{-1} as a predictor. With this basic error feedback modulator, only a small fraction of the bandwidth can be occupied by the required signal. In addition, the circuit only operates at baseband. A larger fraction of the Nyquist bandwidth can be made available and the modulator can be tuned if a more sophisticated error predictor is employed. This requires replacing the unit delay with a prediction filter $P(z)$. This generalized modulator is shown in Fig. 10.

The operation of the requantizer can be understood by considering the transform domain description of the circuit. This is expressed in (7) as

$$\hat{X}(z) = X(z) + \underline{Q}(z)(1 - P(z)z^{-1}), \tag{7}$$

where $\underline{Q}(z)$ is the z-transform of the equivalent noise source added by the quantizer $q(\cdot)$, $P(z)$ is the transfer function of the error predictor filter, and $X(z)$ and $\hat{X}(z)$ are the transforms of the system input and output, respectively. $P(z)$ is designed to have unity gain and leading phase shift in the bandwidth of interest. Within the



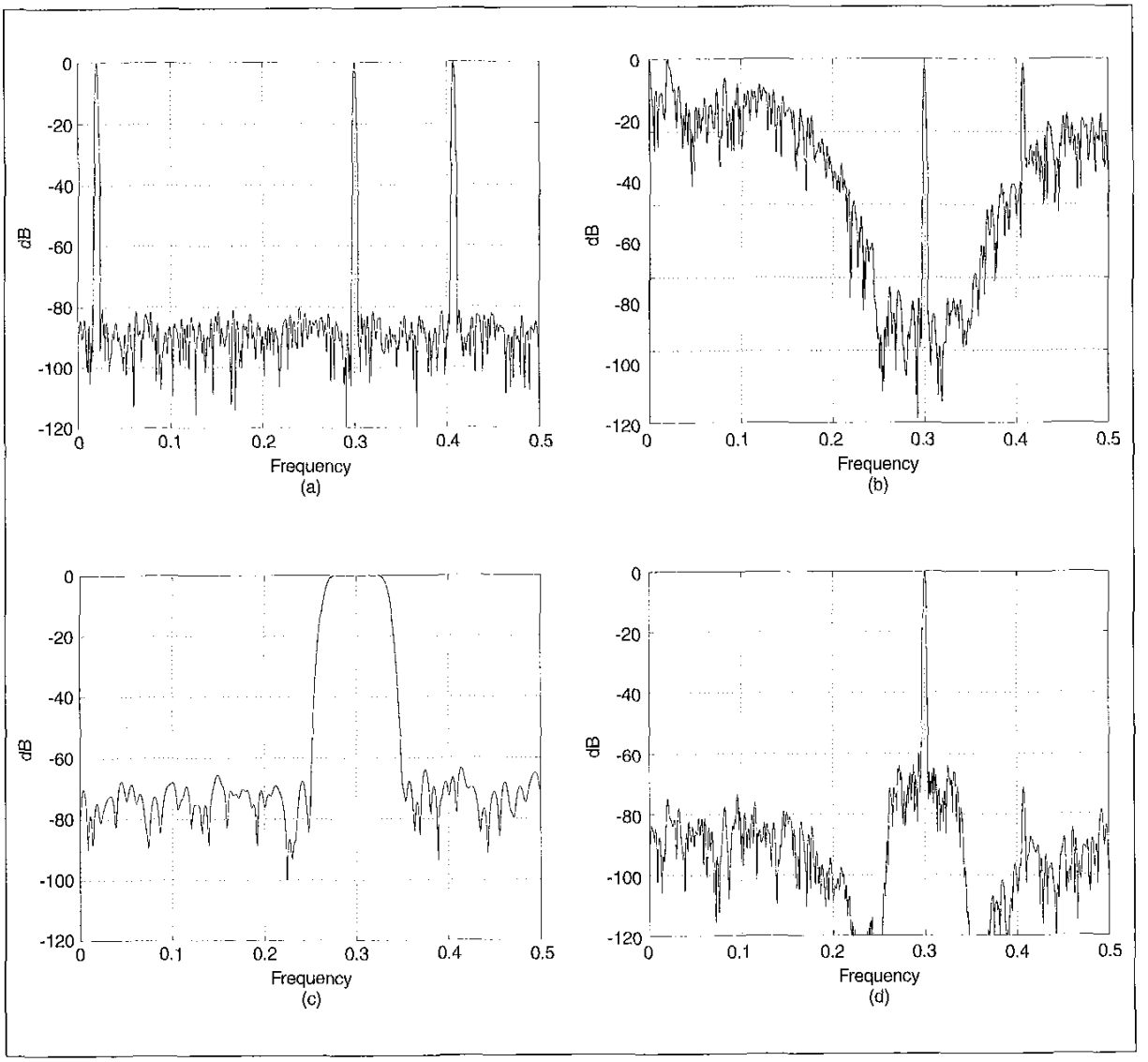
▲ *Fig. 10. Tunable sigma-delta modulator using a linear filter in the feedback path.*

design bandwidth, the term $Q(z)(1-P(z)z^{-1})=0$, therefore, $\hat{X}(z)=X(z)$. By designing $P(z)$ to be commensurate with the system passband specifications, the in-band spectrum of the requantizer output will ideally be the same as the corresponding spectral region of the input signal.

To illustrate the operation of the system, consider the task of recovering a signal that occupies 10% of the available bandwidth, and is centered at a normalized frequency of 0.3 Hz. The stopband requirement is to provide 60 dB of attenuation. Fig. 11a shows the input test signal. It comprises an in-band component and two out-of-band tones that are to be rejected. Fig. 11b is a frequency domain plot of the signal after it has been requantized to 4 bits of precision by a $\Sigma\Delta$ modulator employing an eighth-order predictor in the feedback path. Notice that the 60-db dynamic range requirement is supported in the bandwidth of interest, but that the out-of-band SNR has been compromised. This is, of course, acceptable, since the subsequent filtering operation will provide the necessary rejection. A 160-tap filter $H(z)$ satisfies the problem specifications. The frequency response of $H(z)$ using 12-bit filter coefficients is shown in Fig. 11c. Finally, $H(z)$ is applied to the reduced sample precision data stream $\hat{X}(z)$ to produce the spectrum shown in Fig. 11d. Observe that the desired tone has been recovered, the two out-of-band components have been rejected, and the in-band dynamic range meets the 60-db requirement.

Prediction Filter Design

The design of the error predictor filter is a signal estimation problem [10], [5]. The optimum predictor is designed from a statistical viewpoint. The optimization



▲ *Fig. 11. (a) Input signal - 12b samples. (b) Shaped input - 4b samples. (c) Filter response - 12b coefficients. (d) Filtered result.*

The most challenging aspect of implementing the data modulator is producing an efficient implementation for the prediction filter $P(z)$.

criterion is based on the minimization of the mean-squared error. As a consequence, only the second-order statistics (autocorrelation function) of a stationary process are required in the determination of the filter. The error predictor filter is designed to predict samples of a band-limited white noise process $N_{xx}(\omega)$ shown in Fig. 12.

$N_{xx}(\omega)$ is defined as

$$N_{xx}(\omega)=\begin{cases}1 & -\theta\le\omega\le\theta\\ 0 & \text{otherwise}\end{cases} \tag{8}$$

and related to the autocorrelation sequence $r_{xx}(m)$ by discrete-time Fourier transform (DTFT)

$$N_{xx}(\omega)=\sum_{n=-\infty}^{\infty} r_{xx}(k)e^{-j\omega n}. \tag{9}$$

The autocorrelation function $r_{xx}(n)$ is found by taking the inverse DTFT of (9)

$$r_{xx}(n)=\frac{1}{2\pi}\int_{-\pi}^{\pi} N_{xx}(\omega)e^{-j\omega n}d\omega. \tag{10}$$

$N_{xx}(\omega)$ is non-zero only in the interval $-\theta\le\omega\le\theta$ giving $r_{xx}(n)$ as

$$r_{xx}(n)=\frac{\theta}{\pi}\text{sinc}(\theta n). \tag{11}$$

So the autocorrelation function corresponding to a band-limited white noise power spectrum is a sinc function. Samples of this function are used to construct an autocorrelation matrix, which is used in the solution of the normal equations to find the required coefficients. Leaving out the scaling factor in (11), the required autocorrelation function $r_{xx}(n)$, truncated to p samples, is defined as

$$r_{xx}(n)=\frac{\sin(n\theta)}{n\theta}\quad n=0,\ldots,p-1. \tag{12}$$

The normal equations are defined as

$$r_{xx}(m)=\sum_{k=1}^{p} a(k)r_{xx}(m-k)\quad m=1,2,\ldots,p \tag{13}$$

equations [10]. This system of equations can be compactly written in matrix form by first defining several matrices.

To design a p-tap error predictor filter, first compute a sinc function consisting of $p+1$ samples and construct the autocorrelation matrix R_{xx} as

$$R_{xx}=\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(p-1)\\ r_{xx}(1) & r_{xx}(0) & \cdots & r_{xx}(p-2)\\ \vdots & \vdots & \vdots & \vdots\\ r_{xx}(p-1) & r_{xx}(p-2) & \cdots & r_{xx}(0)\end{bmatrix}. \tag{14}$$

Next define a filter coefficient row-vector A as

$$A=[a(0),a(1),\cdots,a(p-1)], \tag{15}$$

where $a_i\ \ i=0,\ldots,p-1$ are the predictor filter coefficients. Let the row-vector R'_{xx} be defined as

$$R'_{xx}=[r_{xx}(1),r_{xx}(2),\cdots,r_{xx}(p)]. \tag{16}$$

The matrix equivalent of (13) is

$$R_{xx}A^T=(R'_{xx})^T. \tag{17}$$

The filter coefficients are, therefore, given as

$$A^T=R_{xx}^{-1}(R'_{xx})^T. \tag{18}$$

For the case in-hand, the solution of (18) is an ill-conditioned problem. To arrive at a solution for A, a small constant ϵ is added to the elements along the diagonal of the autocorrelation matrix R_{xx} in order to raise its condition number. The actual autocorrelation matrix used to solve for the predictor filter coefficients is

$$R_{xx}=\begin{bmatrix} r_{xx}(0)+\epsilon & r_{xx}(1) & \cdots & r_{xx}(p-1)\\ r_{xx}(1) & r_{xx}(0)+\epsilon & \cdots & r_{xx}(M-2)\\ \vdots & \vdots & \vdots & \vdots\\ r_{xx}(p-1) & r_{xx}(p-2) & \cdots & r_{xx}(0)+\epsilon\end{bmatrix}. \tag{19}$$

Bandpass Predictor

The previous section described the design of a lowpass predictor. In this section, bandpass processes are considered.

A bandpass predictor filter is designed by modulating a lowpass prototype sinc function to the required center frequency θ_0 [4]. The bandpass predictor coefficients $h_{BP}(n)$ are obtained by solving the normal equations with a heterodyned sinc function

$$\text{sinc}_{BP}(n)=\text{sinc}_{LP}(n)\cos(\theta_0(n-k))\quad n=0,\ldots,2p, \tag{20}$$

where $k=\left\lfloor\frac{2p+1}{2}\right\rfloor$.

Highpass Predictor

The highpass predictor coefficients $h_{HP}(n)$ are obtained by solving the normal equations with a sinc function heterodyned to the half sample rate

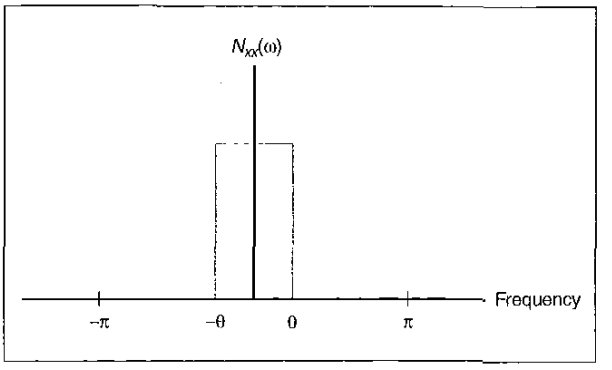
$$\text{sinc}_{HP}(n) = \text{sinc}_{LP}(n)(-1)^{n-k} \quad n = 0,\ldots,2p\,. \tag{21}$$

ΣΔ Modulator FPGA Implementation

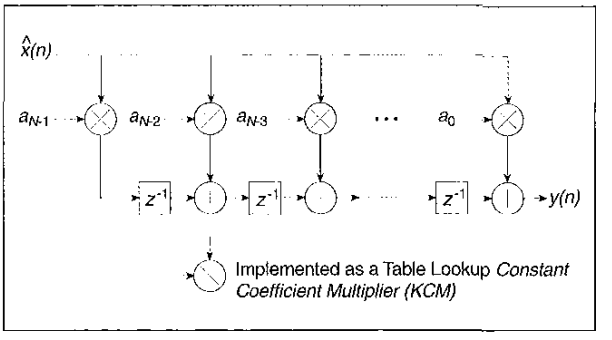
The most challenging aspect of implementing the data modulator is producing an efficient implementation for the prediction filter $P(z)$. The desire to support high-sample rates, and the requirement of zero latency for $P(z)$, will preclude bit-serial methods from this problem. In addition, for the sake of area efficiency, parallel multipliers that exploit one time-invariant input operand (the filter coefficients) will be used, rather than general variable-variable multipliers. The *constant coefficient multiplier* (KCM) is based on a multi-bit inspection version of Booth's algorithm [8]. Partitioning the input variable into 4-bit nibbles is a convenient selection for the Xilinx Virtex function generators (FG) [12]. Each FG has four inputs and can be used for combinatorial logic or as application RAM/ROM. Each logic slice [12] in the Virtex logic fabric comprises two FGs, and can accommodate a 16×2 memory slice. Using the rule of thumb that each bit of filter coefficient precision contributes 5 db to the sidelobe behavior, 12-bit precision is used for $P(z)$. 12-bit precision will also be employed for the input samples. There are three 4-bit nibbles in each input sample. Concurrently, each nibble addresses independent 16×16 lookup tables (LUTs). The bit growth incorporated here allows for worst-case filter coefficient scaling in $P(z)$. No pipeline stages are permitted in the multipliers as a result of $P(z)$'s location in the feedback path of the modulator. It is convenient to use the transposed FIR filter for constructing the predictor. This allows the adders and delay elements in the structure to occupy a single slice. 64 slices are required to build the accumulate-delay path. The FPGA logic requirements for $P(z)$, using a nine-tap predictor, is 424 logic slices [12]. A small amount of additional logic is required to complete the entire ΣΔ modulator. The final slice count is 450. The entire modulator comfortably operates with a 113 MHz clock. This clock frequency defines the system sample rate, so the architecture can support a throughput of 113 Msamples/s. The critical path through this part of the design is related to the exclusion of pipelining in the multipliers.

Reduced-Complexity FIR Mechanization

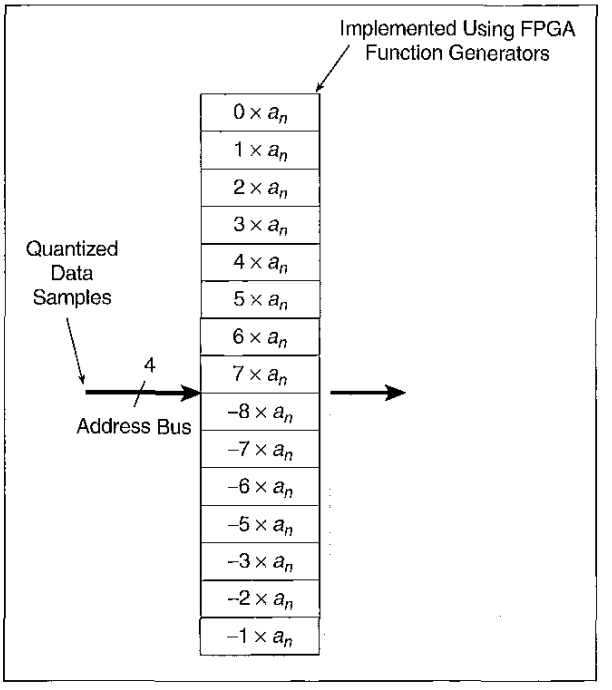
Now that the input signal is available as a reduced precision sample stream, filtering can be performed using area-optimized hardware. For the reasons discussed above, 4-bit data samples are a convenient match for Virtex devices. Fig. 13 shows the structure of the reduced complexity FIR filter. The coded samples $\hat{x}(n)$ are presented to the address inputs of N coefficient LUTs. In accordance with the modulated data stream precision, each LUT stores the 16 possible scaled coefficient values for one tap as shown in Fig. 14. An N-tap filter requires N such elements. The outputs of the minimized multipliers are combined with an add-delay datapath to



▲ *Fig. 12. The ΣΔ modulator error predictor filter is designed to predict samples of a narrow-band white noise process.*

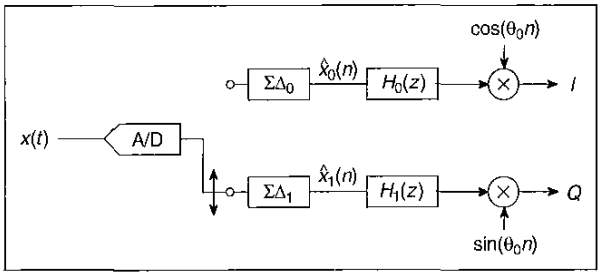


▲ *Fig. 13. Area optimized FPGA FIR structure.*



▲ *Fig. 14. Constant coefficient multiplier.*

produce the final result. The logic requirement for the filter is $\Gamma(H(z)) = N\Gamma(MUL) + (N-1)\Gamma(ADD\_z^{-1})$, where $\Gamma(MUL)$ and $\Gamma(ADD\_z^{-1})$ are the FPGA area cost functions for a KCM multiplier and an add-delay datapath component, respectively.

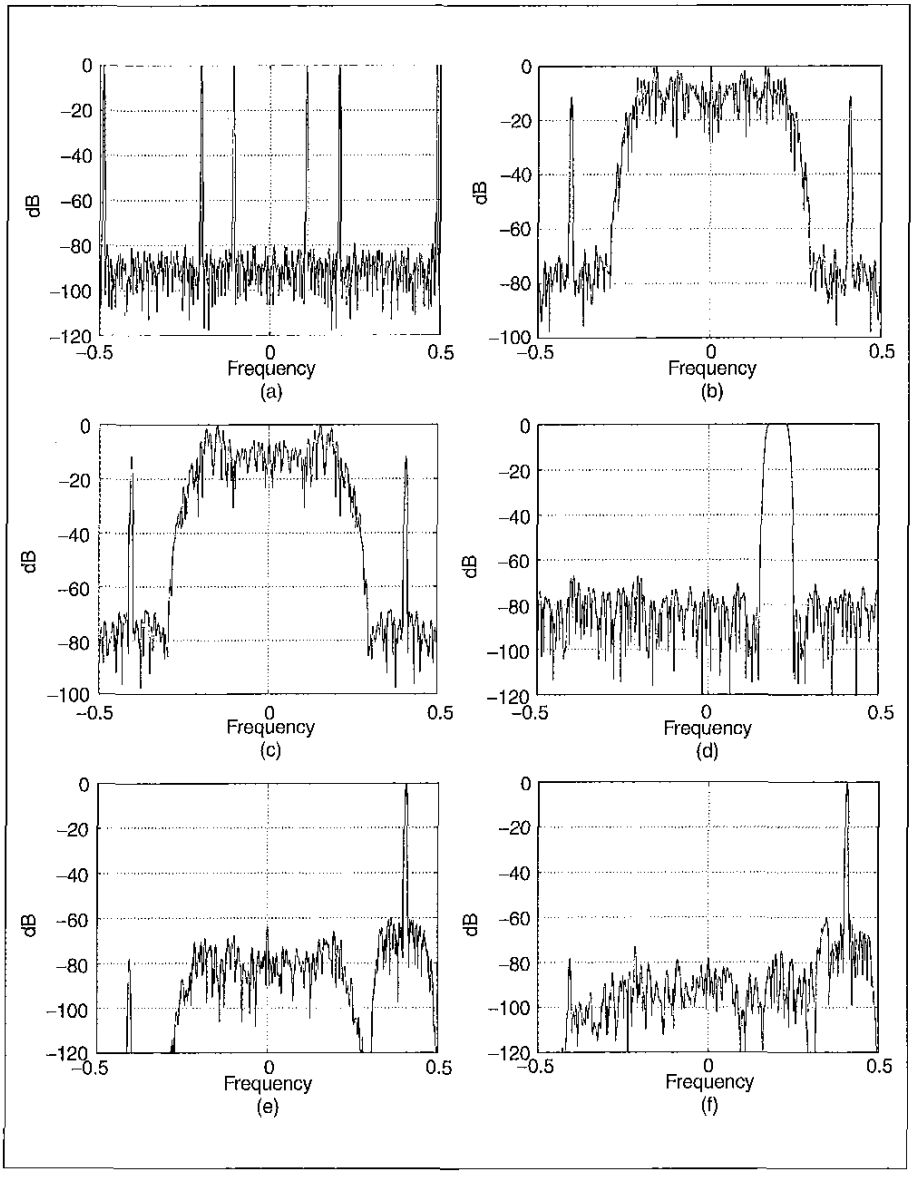


▲ *Fig. 15. Dual half-rate ΣΔ modulators in m:1 complex decimator configuration.*

Using full-precision input samples without any ΣΔM encoding, each KCM would occupy 40 slices. The total cost of a direct implementation of $H(z)$ is 7672 slices. The reduced-precision KCMs used to process the encoded data each consume only eight slices. Including the ΣΔ modulator, the slice count is 3002 for the ΣΔ approach. Therefore, the data requantization approach consumes only 39% of the logic resources of a direct implementation.

ΣΔ *Decimators*

The procedure for requantizing the source data can also be used effectively in an m:1 decimation filter. An interesting problem is presented when high input sample rates ($\geq 150 MHz$) must be supported in FPGA technology. High-performance multipliers are typically realized by incorporating pipelining in the design. This naturally introduces some latency into the system. The location of the predictor filter $P(z)$ requires a zero-latency design. (It is possible that the predictor could be modified to predict samples further ahead in the time series, but this potential modification will not be dealt with here.) Instead of requantizing, filtering, and decimating, which would require a ΣΔ modulator running at the input sample rate, this sequence of operations must re-ordered to permit several slower modulators to be used in parallel. The process is performed by first decimating the signal, requantizing, and then filtering. Now, the ΣΔ modulators operate at the reduced output sample rate. This is depicted in Fig. 15. To support arbitrary center frequencies, and any arbitrary, but integer, down-sampling factor m, the bandpass decimation filter must employ complex weights. The filter weights are just the bandpass-modulated coefficients of a lowpass prototype filter designed to support the bandwidth of the target signal. Samples are collected from the ADC and alternated between the two modulators. Both modula-



▲ *Fig. 16. (a) Input signal. (b) Shaped data $\hat{x}_0(n)$. (c) Shaped data $\hat{x}_1(n)$. (d) Complex filter. (e) Recovered result. (f) Filtered signal - single modulator.*

tors are identical and use the same predictor filter coefficients. The requantized samples are processed by an $m:1$ complex polyphase filter to produce the decimated signal. Several design options are presented once the signal has been filtered and the sample rate lowered. Fig. 15 illustrates one possibility. Now that the data rate has been reduced, the low rate signal is easily shifted to baseband with a simple, and area efficient, complex heterodyne. One multiplier and a single digital frequency synthesizer could be time-shared to extract one or multiple channels.

It is interesting to investigate some of the changes that are required to support the ΣΔ decimator. What may not be immediately obvious is that the center frequency of the prediction filter must be designed to predict samples in the required spectral region in accordance with the output sample rate. For example, consider $m=2$, and the required channel center frequency located at 0.1 Hz, normalized with respect to the input sample rate. The prediction filter must be designed with a center frequency located at 0.2 Hz. In addition, the *quality* of the prediction must be improved. With respect to the output sample rate, the predictors are required to operate over a wider fractional bandwidth. This implies more filter coefficients in $P(z)$. The increase in complexity of this component must be balanced against the savings that result in the reduced complexity filter stage to confirm that a net savings in logic requirements is produced. To more clearly demonstrate the approach, consider a 2:1 decimator, a channel center frequency at 0.2 Hz, and a 60-dB dynamic range requirement.

Fig. 16(a) shows the double-sided spectrum of the input test signal. The input signal is commutated between $\Sigma\Delta_0$ and $\Sigma\Delta_1$ to produce the two low precision sequences $\hat{x}_0(n)$ and $\hat{x}_1(n)$. The respective spectrums of these two signals are shown in Figs. 16b and 16c. The complex decimation filter response is defined in Fig. 16d. After filtering, a complex sample stream supported at the low output sample rate is produced. This spectrum is shown in Fig. 16e. Observe that the out-of-band components in the test signal have been rejected by the specified amount, and that the in-band data meets the 60-dB dynamic range requirement. For comparison, the signal spectrum resulting from applying the processing stages in the order requantize, filter, and decimate, is shown in Fig. 16f. The interesting point to note is that while the dual ΣΔ modulator approach satisfies the system performance requirements, its out-of-band performance is not quite as good as the response depicted in Fig. 16f. The stopband performance of the dual modulator architecture has degraded by approximately 6 dB. This can be explained by noting that the shaping noise produced by each modulator is essentially statistically independent. Since there is no coupling between these two components prior to filtering, complete phase cancelation of the modulator noise cannot occur in the polyphase filter.

To provide a frame of reference for the ΣΔ decimator, consider an implementation that does not preprocess the input data, but just applies it directly to a polyphase decimation filter. A complex filter processing real-valued data consumes double the FPGA resources of a filter with real weights. For $N=160$, the area cost $\Gamma(\text{FIR})=15344$ slices. This figure is based on a cost of 40 CLBs for each KCM and eight logic slices for an add-delay component.

What may not be immediately obvious is that the center frequency of the prediction filter must be designed to predict samples in the required spectral region in accordance with the output sample rate.

Now consider the logic accounting for the dual modulator approach. The area cost $\Gamma(\hat{\text{FIR}})$ for this filter is

$$\Gamma(\hat{\text{FIR}})=2\Gamma(\Sigma\Delta)+\Gamma(\hat{\text{MUL}})+\Gamma(n-1)(\text{ACC\_}z^{-1}), \quad (22)$$

where $\Gamma(\Sigma\Delta)$ represents the logic requirements for one ΣΔ modulator, and $\Gamma(\hat{\text{MUL}})$ is the logic needed for a reduced-precision multiplier. Using the filter specifications defined earlier, and 18-tap error prediction filters, $\Gamma(\hat{\text{FIR}})=6596$. Comparing the area requirements of the two options produces the ratio

$$\lambda=\frac{\Gamma(\hat{\text{FIR}})}{\Gamma(\text{FIR})}=6596/15344\approx 43\%. \quad (23)$$

Therefore, for this example, the requantization approach has produced a realization that is significantly more area-efficient than a standard tapped-delay line implementation.

Center Frequency Tuning

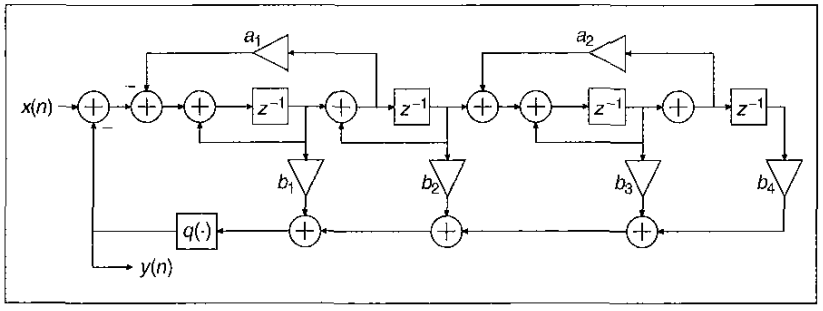
For both the single-rate and multi-rate ΣΔ-based architectures, the center frequency is defined by the coefficients in the predictor filter and the coefficients in the primary filter. The constant coefficient multipliers can be constructed using the FPGA function generators configured as RAM elements. When the system center frequency is to be changed, the system control hardware would update all of the tables to reflect the new channel requirements. If only several channel locations are anticipated, separate configuration bit streams [12] could be stored, and the FPGA(s) reconfigured as needed.

Bandpass ΣΔMs Using Allpass Networks

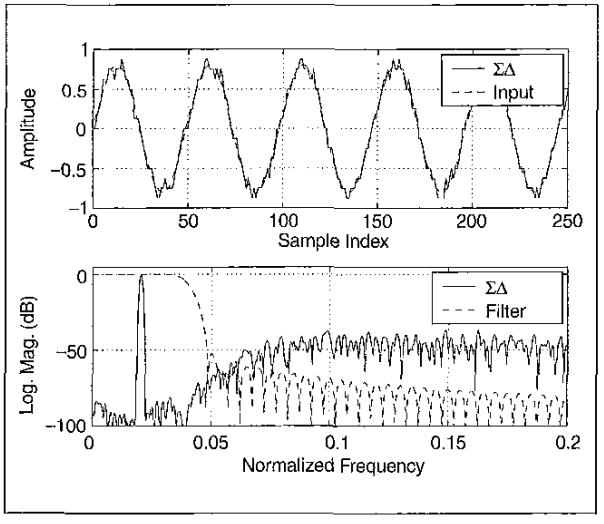
In an earlier section, we discussed how to design a predicting filter for the feedback loop of a standard ΣΔ modulator. The predicting filter increases the order of the modulator so that the modified structure has additional

degrees of freedom relative to a single-delay noise feedback loop. These extra degrees of freedom have been used in two ways, first to broaden the bandwidth of the loop's noise transfer function, and second to tune its center frequency. The tuning process entailed an offline solution of the Normal equations, which while not difficult, does present a small delay and the need for a background processor.

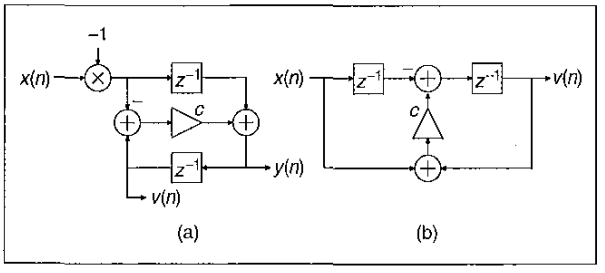
We can define a ΣΔ loop with a completely different architecture that offers the same flexibility, namely wider bandwidth and a tunable center frequency that does not require this background task. In this alternate architecture, a fixed set of feedback weights from a set of digital integrators defines a baseband prototype filter with a desirable NTF. The filter is tuned to arbitrary frequencies by attaching to each delay element z^{-1}, a simple subprocessing element that performs a baseband-to-bandpass transformation of the prototype filter. This processing element tunes the center frequency of its host prototype with a single real and selectable scalar. The structure of a fourth-order prototype ΣΔ loop is shown in Fig. 17. The time series and spectrum obtained by using the loop with a 4-bit quantizer is shown in Fig. 18. In this structure, the digital integrator poles are located on the unit circle at dc. The local feedback ($a1$ and $a2$) separates the poles by sliding them along the unit circle, and the global feedback ($b1$,$b2$,$b3$, and $b4$) places these poles in the feedback path of the quantizer so they become noise transfer function zeros. These zeros are positioned to form an equal-ripple stop band for the NTF. The coefficients are selected to match the NTF pole-zero locations to an elliptic highpass filter. The single-sided bandwidth of this fourth-order loop is approximately 4% of the input sample rate.



▲ *Fig. 17. Fourth-order sigma-delta loop.*



▲ *Fig. 18. Input and output time series of base-band prototype fourth order, 4-bit sigma-delta loop (top) and output spectrum (bottom).*



▲ *Fig. 19. Block diagram of all-pass transfer function $G(z)$ form Equation (1).*

The lowpass to bandpass transformation for a sampled data filter is achieved by substituting an allpass transfer function $G(z)$ for the allpass transfer function z^{-1}. This transformation is shown in (24)

$$z^{-1} \rightarrow -z^{-1}\left(\frac{1-cz}{z-c}\right). \tag{24}$$

A block diagram of a digital filter with the transfer function for $G(z)$ is shown in Fig. 19. Examining the left block diagram, we find the transfer function from $x(n)$ to $y(n)$ is the allpass network $-(1-cz)/(z-c)$, while the transfer function from $x(n)$ to $v(n)$ is $-(1/z)(1-cz)/(z-c)$. When we absorb the external negative sign change in the internal adders of the filter, we obtain the simple right-hand side version of the desired transfer function $G(z)$.

After the block diagram substitution has been made, we obtain Fig. 20, the tunable version of the lowpass prototype. The basic structure of the prototype remains the same when we replace the delay with the tunable allpass network. The order of the filter is doubled by the substitution, since each delay is replaced by a second-order sub-filter. Tuning is trivially accomplished by changing the c multiplier of the all-pass network. The tuned version of the system reverts back to the prototype response if we set c to 1.

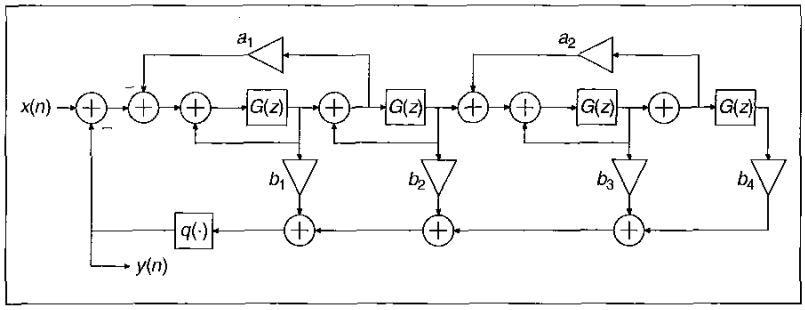
Fig. 21 presents the time series and spectrum obtained by using the tunable loop with a 4-bit quantizer shown in Fig. 20. The single-sided bandwidth of the prototype filter is distributed to the positive and nega-

tive spectral bands of the tuned filter. Thus, the two-sided bandwidth of each spectral band is approximately 4% of the input sample rate.

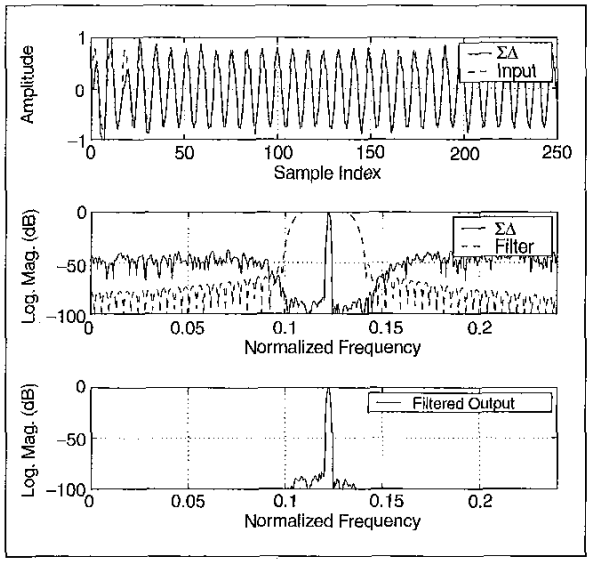
We now estimate the computational workload required to operate the prototype and tunable filter. The prototype filter has six coefficients to form the four poles and the four zeros of the transfer function. The two a_k $k=0,1$ coefficients determine the four zero locations. These are small coefficients and can be set to simple binary scalers. The values computed for this filter for a_1 and a_2 were 0.0594 and 0.0110, respectively. These can be approximated by 1/16 and 1/128, which lead to no significant shift of the spectral zeros in the NTF. These simple multiplications are virtually free in the FPGA hardware since they are implemented with suitable wiring. The four coefficients b_k $k=0,\ldots,3$ are 1.000, 0.6311, 0.1916, and 0.0283, respectively, and were replaced with coefficients containing one or two binary symbols to obtain values 1.000, 1/2+1/8 (.625), 1/8+1/16 (0.1875), and 1/32 (0.03125). When the ΣΔ loop ran with these coefficients, there was no discernable change in bandwidth or attenuation level of the loop. The loop operates equally as well in the tuning mode and the non-tuning mode with the approximate coefficients listed above. Thus, the only real multiplies in the tunable ΣΔ loop are the c coefficients of the allpass networks. These networks are unconditionally stable and always exhibit allpass behavior even in the presence of finite arithmetic and finite coefficients. This is because the same coefficient forms the numerator and the denominator. Errors in approximating the coefficients for c simply result in a frequency shift of the filter's tuned center. The c coefficient is determined from the cosine of the center frequency (in radians/sample). The curve for this relationship is shown in Fig. 22. Also shown is an error due to approximating c by $c+\delta c$. The question is, what is the change in center frequency θ, from θ to θ + δθ due to the approximation of c? We can see that the slope at the operating point on the cosine curve is $-\sin(\theta)$, so that $\delta c / \delta\theta \approx -\sin(\theta)$. As a result, $\delta c \approx -\delta\theta\sin(\theta)$ is the required precision to maintain a specified error. We note that tuning sensitivity is most severe for small frequencies where $\sin(\theta)$ is near zero. The tolerance term, $\delta\theta\sin(\theta)$, is quadratic for small frequencies, but the lowest frequency that can be tuned by the loop is half the NTF passband bandwidth. For the fourth-order system described here, this bandwidth is 4% of sample rate, so the half-bandwidth angle is 2% or 0.126 radians. To assure that the frequency to which the loop is tuned has an error smaller than 1% of center frequency, $\delta c < \delta\theta\sin(\theta) \Rightarrow \delta c < (0.126/100)(0.126) = 0.0002$, which corresponds to a 14-bit coefficient. An error of less than 10% center frequency can be achieved with 10-bit coefficients.

The *tuning* multipliers could be implemented as full multipliers in the FPGA hardware or as dynamically reconfigured KCMs, or a *KDCM*, as shown in Fig. 23. The later approach conserves FPGA resources at the expense of introducing a start-up penalty each time the center frequency is changed. The start-up period is the initialization time of the KCM LUT.

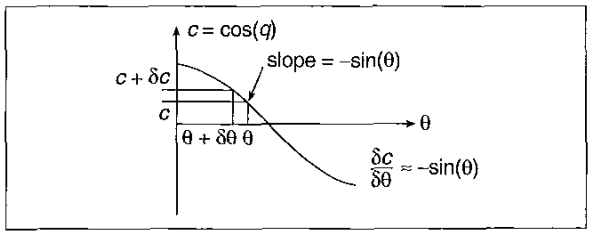
When a new center frequency is desired, the tuning constant is presented to the k input of the KDCM and the load signal *LD* is asserted. This starts the initialization en-



▲ *Fig. 20. Tunable sigma-delta loop. Prototype is fourth order. Tunable version is eighth order.*

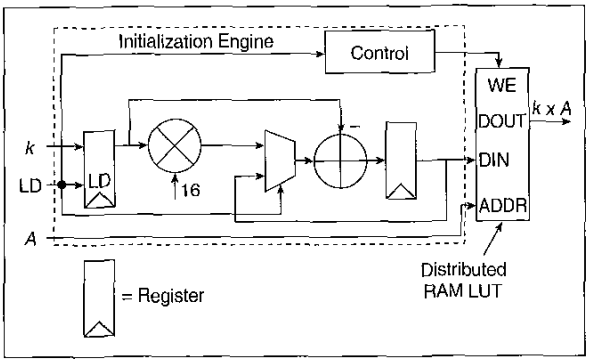


▲ *Fig. 21. Input and output time series of tuned base-band prototype fourth-order, 4-bit sigma-delta loop (top), filter and modulator output spectrum (middle), and filtered output spectrum (bottom).*

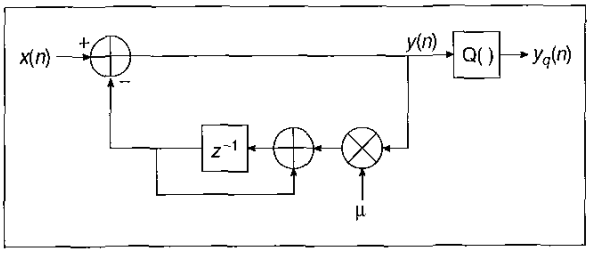


▲ *Fig. 22. Relationship between errors in constant c in all-pass networks to errors in tuning frequency.*

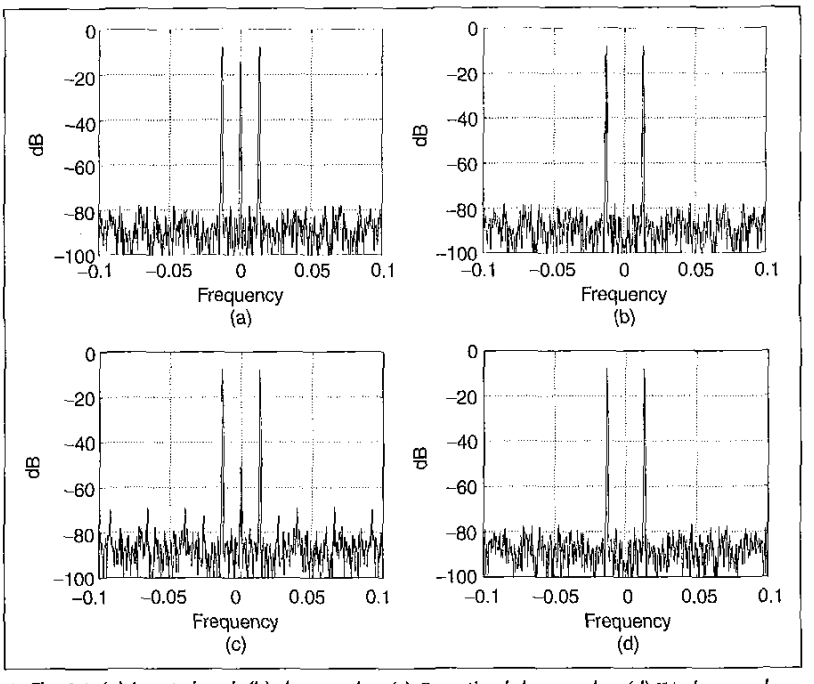
gine, which requires 16 clock cycles to initialize 16 locations in the multiplier LUT. The initialization engine relies on the automatic shift mode [12] of the Virtex LUTs. In this mode of operation, a LUT's register contents are passed from one cell to the next on each clock tick. This avoids the requirement for a separate address generator and multiplexer in the initialization hardware. Observe from Fig. 23 that the initialization engine only introduces a small amount of additional hardware over that of a static KCM.



▲ *Fig. 23. Loadable constant coefficient multiplier (KDCM).*



▲ *Fig. 24. Simple dc canceler.*



▲ *Fig. 25. (a) Input signal, (b) dc canceler, (c) Quantized dc canceler, (d) ΣΔ dc canceler.*

There is approximately a factor of four difference in the area of a KDCM and full multiplier.

ΣΔ DC Canceler

Unwanted DC components can be introduced into a DSP datapath at several places. It may be presented to the system via an untrimmed offset in the analog-to-digital conversion preprocessing circuit, or may be attributed to bias in the ADC itself. Even if the sampled input signal has a zero mean, DC content can be introduced though arithmetic truncation processes in the fixed-point datapath. For example, in a multi-stage multi-rate filter, the intermediate filter output samples may be quantized between stages to compensate for the filter processing gain, thereby keeping the word-length requirements manageable. The introduced DC bias can impact the dynamic range performance of a system and potentially increase the error rate in a digital receiver application.

In a fixed-point datapath, the bias can cause unnecessary saturation events that would not occur if the DC was not present in the system.

In a digital communication receiver employing M-ary QAM modulation, the DC bias can interfere with the symbol decision process, causing incorrect decoding and therefore, increasing the bit error rate.

In some cases, the introduced bias can be ignored and is of no concern. However, for other applications it is desirable to remove the DC component.

One solution to removing the unwanted DC level is to employ a DC canceler.

A simple canceler is shown in Fig. 24. It is easy to show that the transfer function of the network is

$$H(z)=\frac{z-1}{z-(1-\mu)}. \tag{25}$$

The cancelation is due to the transfer function zero at 0 Hz. The pole at $1-\mu$ controls the system bandwidth and hence, the system transient response. The location of the zero at $z=1$ completely removes the DC component in the signal, but there are some problems with a practical implementation of this circuit.

Fig. 25a is a spectral domain representation of a biased signal presented to the DC canceler. Fig. 25b is the processed signal spectrum at $y_q(n)$ in Fig. 24. We observe that the DC content in the input signal has been completely removed. However, in the process of running the canceling loop, the network processing gain has caused a dynamic range expansion. So, although the sample

stream $y_q(n)$ is a zero mean process, it requires a larger number of bits to represent each sample than is desirable. The only option with the circuit is to requantize $y_q(n)$ to produce $y(n)$ using the quantizer $Q(\cdot)$. The effect of this operation is shown in Fig. 25c, which demonstrates, not surprisingly, that after an 8-bit quantizer, the signal now has a DC component, and we are almost back to where we started. How can the canceler be reorganized to avoid this implementation pitfall? One option is to embed the requantizer in the feedback loop in the form of a ΣΔ modulator as shown in Fig. 26. The modulator can be a very simple first-order loop such as the error-feedback ΣΔ modulator shown in Fig. 9. Fig. 25d demonstrates the operation of the circuit for 8-bit output data. Observe from the figure that the dc has been removed from the signal while it employs the same 8-bit output sample precision that was used in Fig. 24. The simple ΣΔM employed in the canceler is easily implemented in an FPGA.

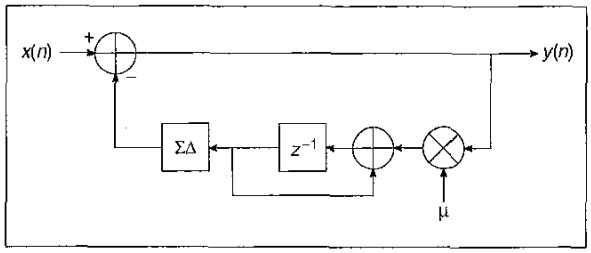
Simplify Digital Receiver Control Loops Using ΣΔ Modulators

In earlier sections, we recognized that when a sampled data input signal has a bandwidth that is a small fraction of its sample rate, the sample components from this restricted bandwidth are highly correlated. We took advantage of that correlation to use a digital ΣΔ modulator to requantize the signal to a reduced number of bits. The ΣΔ modulator encodes the input signal with a reduced number of bits while preserving full-input precision over the signal bandwidth by placing the increased noise due to requantization in out-of-band spectral positions that are already scheduled to be rejected by subsequent DSP processing. The purpose of this requantization is to allow the subsequent DSP processing to be performed with reduced arithmetic resource requirements since the desired data is now represented by a smaller number of bits.

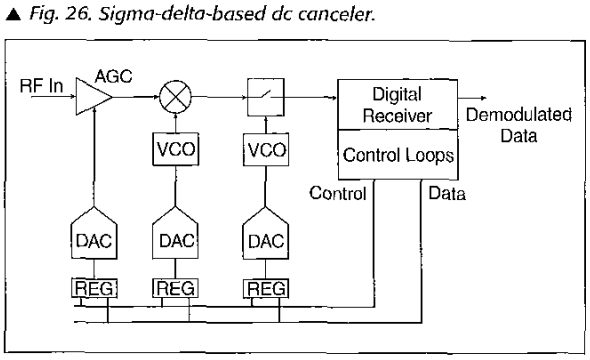
A similar remodulation of data samples can by be employed for signals generated within a DSP process when the bandwidth of the signals are small compared to the sample rate of the process. A common example of this circumstance is the generation of control signals used in feedback paths of a digital receiver. These control signals include a gain control signal for a voltage-controlled amplifier in an automatic gain control (AGC) loop and voltage-controlled oscillator (VCO) control signals in carrier recovery and timing recovery loops [4], [6], [2]. A block diagram of a receiver with these specific controls signals is shown in Fig. 27. The control signals are generated from processes operating at a sample rate appropriate to the input signal bandwidth. The bandwidth of control loops in a receiver are usually a very small fraction of the signal bandwidth, which means that the control signals are very heavily oversampled. As a typical example, in a cable TV modem, the input bandwidth is 6 MHz, the processing sample rate is 20 MHz, and the loop bandwidth may be 50 kHz. For this example, the ratio of sample rate to bandwidth is 400-to-1.

As seen in Fig. 27, the process of delivering these oversampled control signals to their respective control points entails the transfer of 16 bit words to external control registers, requiring appropriate busses, addressing, and enable lines as well as the operation of 16-bit digital-to-analog converters (DACs).

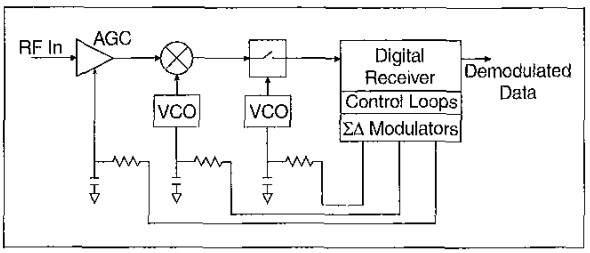
We can use a ΣΔ modulator to requantize the 16-bit oversampled control signals in the digital receiver prior to passing them out of the processing chip. The ΣΔ can preserve the required dynamic range over the signal's restricted bandwidth with a 1-bit output. As suggested in Fig. 28, the transfer of a single bit to control the analog components is a significantly less difficult task than the original. We no longer require registers to accept the transfer, the busses to deliver the bits, or the DAC to convert the digital data to the analog levels the data represents. All we need is a simple filter (and likely an analog amplifier to satisfy drive-level and offset requirements). Experience shows that a 1-bit, one-loop,



▲ *Fig. 26. Sigma-delta-based dc canceler.*



▲ *Fig. 27. Block diagram of digital receiver showing feedback paths containing digital signals converted to analog control signals for analog components.*



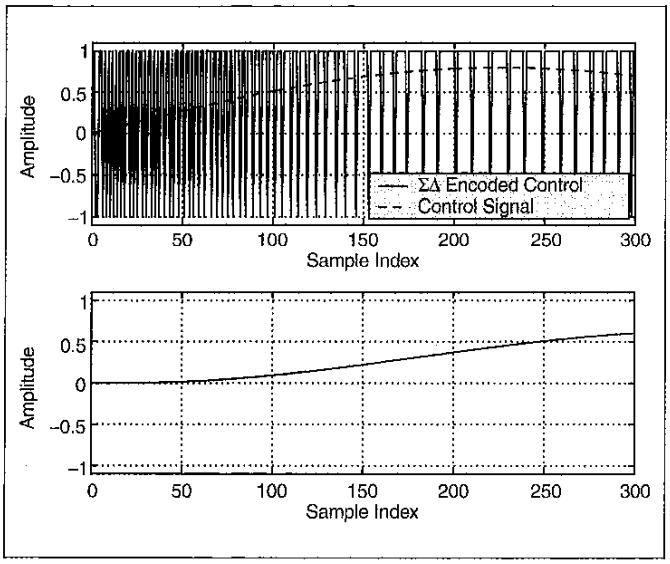
▲ *Fig. 28. Block diagram of digital receiver showing feedback paths delivering and processing one-bit analog control signals generated by sigma-delta modulators.*

The signal processing literature is full of creative solutions to real-world problems. Often these solutions are excluded to a designer because they do not map well to software-programmable DSP architectures.

ΣΔ modulator could achieve an 80-dB dynamic range, and requires a single RC filter to reconstruct the analog signal. A two-loop ΣΔ modulator is required to achieve 16-bit precision for which a double RC filter is required to reconstruct the analog output signal. Fig. 29 shows the time response of the 1-bit, two-loop ΣΔ converter to a slowly varying control signal and the reconstructed signal obtained from the dual-RC filter. Fig. 30 shows the spectrum obtained from a 1-bit two-loop modulator and the spectrum obtained from an unbuffered RC-RC filter.

This example has shown how, with minimal additional hardware, an FPGA can generate analog control signals to control low-bandwidth analog functions in a system.

An observation worthy of note is that the audio engineering community has recognized the advantage offered by this option of requantizing a 16-bit oversampled data stream to 1-bit data stream. There, the output signal is intentionally upsampled by a factor of 64 and then requantized to 1-bit in a process called a MASH converter. Nearly all CD players use the MASH converter to deliver analog audio signals.



▲ *Fig. 29. Control loop signal and ΣΔ encoded control signal (top), filtered ΣΔ sequence (bottom).*

What Have We Gained?

What has been achieved by expressing our signal processing problems in terms of ΣΔM techniques?

The article has demonstrated some ΣΔM techniques for the compact implementation of certain types of filter and control applications using FPGAs. This optimization can be used in several ways to bring economic benefits to a commercial design. By exploiting ΣΔM filter processes, a given processing load may be realizable in a lower-density, and hence, less expensive, FPGA than is possible without access to these techniques. An alternative would be to perform more processing using the same hardware; for example, processing multiple channels in a communication system.

In addition to FPGA area trade-offs, the ΣΔM methods can result in reduced power consumption in a design. Power P may be expressed as

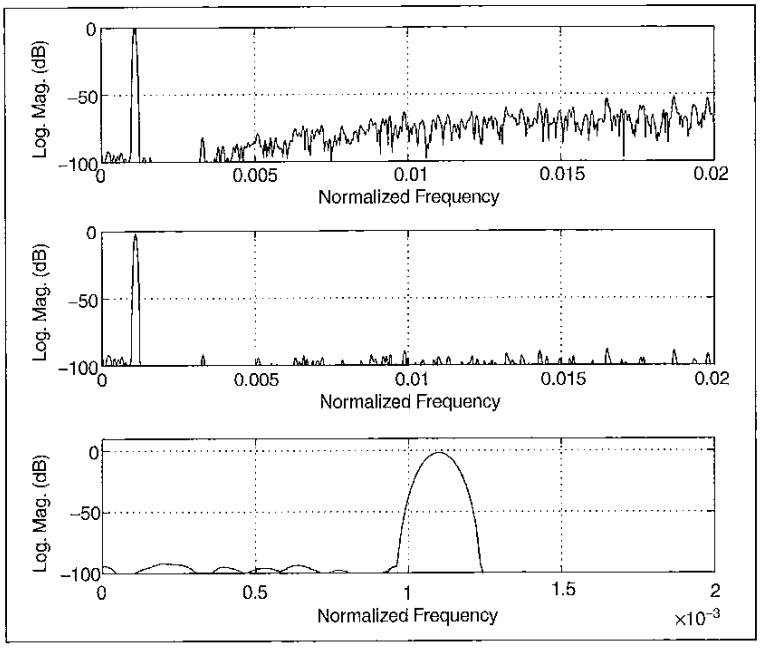
$$P = cv^2 f_{\text{clk}}, \tag{26}$$

where c is capacitance, v is voltage, and f_{clk} is the system clock frequency. By reducing the silicon area requirements of a filter, we can simultaneously reduce the power consumption of the design. For the examples considered earlier, logic resource savings of greater than 50% were demonstrated. The savings is proportional to increased efficiency in the system power budget, and this is very important for mobile applications.

The ΣΔM AGC, timing and carrier recovery control loop designs are also important examples in a industrial context. The examples illustrated how the component count in a mixed analog/digital system can be reduced. In fact, not only is the component count reduced, but printed circuit board area is minimized. This results in more reliable and physically smaller implementations. The reduced component count also results in reduced power consumption. In addition, since the control loops no longer require wide output buses from the FPGA to multi-bit DACs that generate analog control voltages, power consumption is decreased because fewer FPGA I/O pads are being driven.

Conclusion

FPGAs open a range of opportunities in the solution space that can result in high-performance and economic solutions to a DSP problem. Often, this is best achieved via an appropriate paradigm shift in the algorithmic domain to find an optimal approach that best exploits the cellular LUT/flip-flop FPGA architecture. This article has illustrated how ΣΔM techniques can be combined with FPGA technology to address a range of signal processing problems. These included single and multi-rate filters; dc cancelers; and the efficient and compact generation of analog control



▲ *Fig. 30. ΣΔ encoded control signal spectrum (top), filtered ΣΔ sequence spectrum (middle), exploded view at baseband of filtered ΣΔ control signal (bottom).*

signals for AGC, carrier recovery, and timing recovery functions in a communication receiver. The source data requantization approach is suitable for both single-rate and multi-rate filter processes. The proposed method arms the DSP/FPGA engineer with another tool that is useful for certain filtering requirements. For the examples considered here, logic savings in excess of 50% were demonstrated. As the frequency band of interest occupies a smaller fractional bandwidth, the order of the required filter increases. This growth tends to make the data requantization approach more attractive, as the cost of modulators consumes a decreasing proportion of the entire design.

While the article has exclusively focused on ΣΔM methods in the context of FPGA hardware, we feel that there is broader lesson delivered in the study. The signal processing literature is full of creative solutions to real-world problems. Often these solutions are excluded to a designer because they do not map well to software-programmable DSP architectures. The algorithm will have an ASIC solution, but this may not be an option for reasons of schedule, economies of scale, and flexibility. FPGAs do, however, give immediate access to the diverse range of potential solutions. And they do so while simultaneously providing flexibility and high-performance—frequently the performance equals or exceeds that of an ASIC.

In this era of systems-on-a-chip, increased fiscal pressure, tighter engineering deadlines, time-to-market constraints, and increasing performance demands, new system level and hardware architectures must be employed. FPGAs provide a mechanism for working with and performing trade-offs between all of these important variables. As we move into the next millennium, reconfigurable FPGA technology will increasingly provide solutions to signal processing problems.

Chris Dick is a Senior Systems Engineer and the DSP Group Manager at Xilinx Inc., San Jose, CA. His research focus is in the areas of fast algorithms for signal processing, digital communications, software radio, hardware architectures for real-time signal processing, parallel computing, and the use of FPGAs for custom computing machines and signal processing. He has a Ph.D. from La Trobe University, Melbourne, Australia, where he was a professor for 12 years. He is a Member of the IEEE and has authored or co-authored over 50 publications.

Fred Harris holds the CUBIC Signal Processing Chair of the Communication Systems and Signal Processing Institute at San Diego State University, where he has taught since 1967 in areas related to digital signal processing and communication systems. He has extensive practical experience in communication systems involving high-performance modems, underwater acoustics, advanced radar and high performance laboratory instrumentation. He holds a number of patents on digital receiver and DSP technology, and lectures throughout the world on DSP applications. He consults for organizations requiring high performance DSP systems. He is a Senior Member of the IEEE and his education includes a BSEE from the Polytechnic Institute of Brooklyn, an MSEE from San Diego State University, and Ph.D. work at the University of California at San Diego.

References

[1] Atmel, *AT40K/05/10/20/40 Data Sheet*, 1999.

[2] J.A.C. Bingham, *The Theory and Practice of Modem Design*, John Wiley & Sons, New York, 1998.

[3] J.C. Candy and G.C. Temes, "Oversampling Methods for A/D and D/A Conversion" in *Oversampling Delta-SIGMA Data Converters - Theory Design and Simulation*, IEEE Press, New York, 1992.

[4] M.E. Frerking, *Digital Signal Processing in Communication Systems*, Van Nostrand Reinhold, New York, 1994.

[5] S. Haykin, "Modern Filters", Macmillan Publishing Co., New York, 1990.

[6] H. Meyr, M. Moeneclaey and S. A. Fechtal, *Digital Communication Receivers*, John Wiley & Sons Inc., New York, 1998.

[7] S.R. Norsworthy, R. Schreier, and G. C. Temes, "Delta-Sigma Data Converters," IEEE Press, Piscataway, NJ, 1997.

[8] D.A. Patterson and J.L. Hennessy, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers Inc., California, 1990.

[9] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 22, pp. 456-462, Dec. 1974.

[10] J.G. Proakis, and D.G. Manolakis, *Digital Signal Processing Principles, Algorithms and Applications*, 2nd edition, Maxwell Macmillan International, New York, 1992.

[11] S.A. White, "Applications of distributed arithmetic to digital signal processing," *IEEE ASSP Magazine*, vol. 6, no. 3, pp. 4-19, July 1989.

[12] Xilinx Inc., *The Programmable Logic Data Book*, 1999.