An FPGA Implementation of an Oversampling, Second-order Noise Shaping DAC

Eric Jonas, Massachusetts Institute of Technology Class of 2003

***Abstract*— An oversampling, noise-shaping digital-to-analog converter architecture is designed and implemented in a Xilinx Spartan-II Field-Programmable Gate Array. The system uses 128-times and 256-times oversampling and has zero-, first-, and second-order noise shapers which may be selected by the end user in real-time. 1-bit DAC output is passed through a 4-pole analog Butterworth filter. Particular emphasis is paid to minimizing silicon space and achieving performance comparable to that of Phillips Semiconductor's UDA1320ATS device, as well as allowing the ned user to select oversampling rate and noise-shaping order.**

I. Introduction

THE modern development of oversampling, noise-shaping (also known as *delta-sigma*) ADCs and DACs is a testament to the tremendous progress made in digital technology. These devices increase the effective resolution of coarse converters by employing a combination of oversampling and negative feedback to reduce quantization noise in the baseband. This increase can be tremendous – one-bit converters which achieve 16-bits of in-band resolution are common.

The negative feedback "noise-shaping" architecture reduces the noise markedly in the passband, and (following a 1-bit DAC) allows for a much more trivial analog antialiasing filter. The trade-off allows more complicated (yet still relatively inexpensive) digital hardware to be used in place of more expensive analog circuitry.

Similarly, tremendous advances have been made over the past decade in the area of reconfigurable logic, specifically field-programmable gate arrays. FPGAs (such as the Xilinx Spartan-II used here) are arrays of generalized logic that can be reconfigured for arbitrary functions in-circuit. Xilinx FPGAs are arrays of "slices" – each slice consists of two four-input, single-output function generators, two registers, some buffer logic, and additional fast-carry logic. A complex logic block (CLB) is comprised of two slices. Thus a single CLB can implement a latched four-bit-wide adder. The Spartan-II XC2S50 used herein (a $35 part) contains 1176 CLBs, as well as other useful logic (such as 14 blocks of BlockSelect+ RAM, a 4096-bit dual-ported SRAM) [1].

What follows is an attempt to implement an oversampling, noise-shaping DAC similar in specification to the Philips device. Device parameters are discussed, particularly as they apply to the FPGA implementation. Relevant signal theory concepts are reviewed as they apply to sampled discrete-time systems. The hardware and its properties are described, and both simulated DAC output and actual measured response are reported.

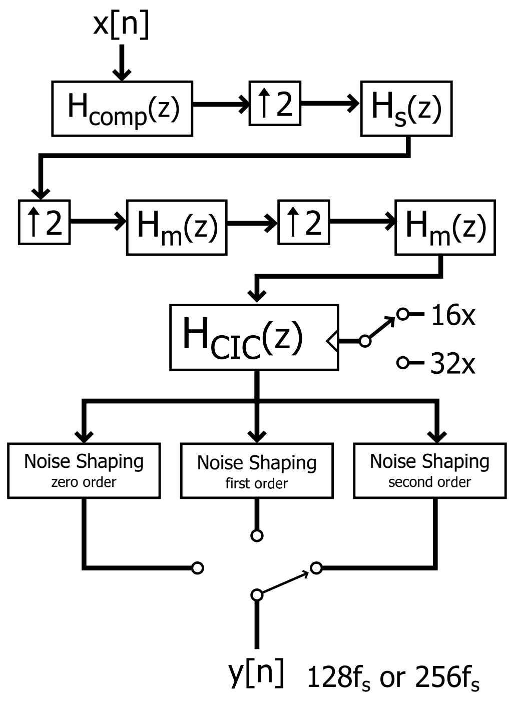


Fig. 1. Flow diagram of oversampling, noise-shaping DAC.

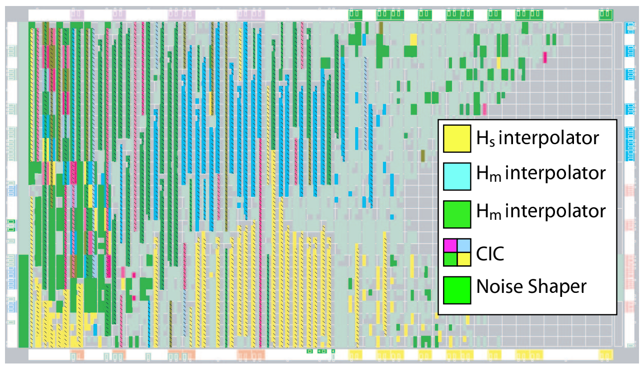


Fig. 2. Device utilization of the Xilinx Spartan-II FPGA used to implement the DAC.

II. System Overview

The overall system runs from a 20 MHz input clock which is clock-doubled to 40 MHz and doubled again to 80 MHz. An overview of the resulting system can be seen in figure 1. The FPGA development board was already assembled with a 20Mhz clock, so the initial sampling rate is $f_s = 62.5$ kHz, and the output is either $128f_s = 8$ MHz or $256f_s = 16$ MHz. Note that the net result is the input audio (resampled by the host computer) simply occupies a smaller portion of the sampled spectrum. This is treated as a minor implementation detail – all analysis is done for the bandwidth of the original Philips DAC, assuming a sampling rate $f_s = 44.1$ kHz.

The overall system is FIR once pole-zero cancellation is considered – in reality, the system is a cascaded FIR-IIR structure.

Data is taken in over USB via a Cypress CY7C64613 USB micro-controller from a Linux host. The byte-wide words are pushed into an internal FIFO of the FPGA implemented in BlockSelect+ RAM. Data is passed at f_s through a simple compensation filter to counteract attenuation of higher frequencies later on in the system. The input

This project began as a filter design project for MIT subject 6.341, *Discrete Time Signal Processing*, Fall 2002

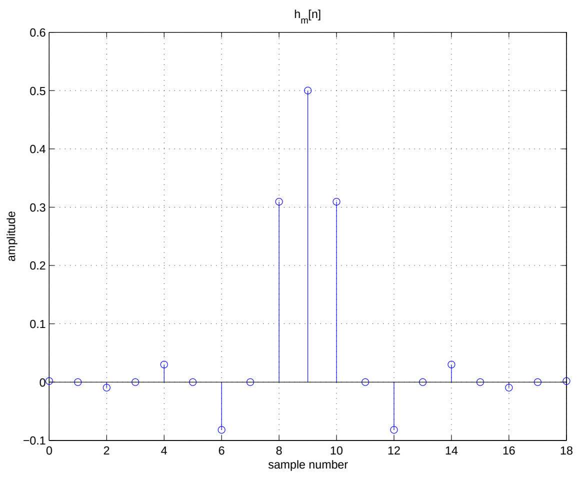


Fig. 3. Half-band FIR impulse response (for h_m, $n = 19$). Response is symmetric, all odd coefficients are zero, except for the center which is fixed at $\frac{1}{2}$.

signal is then passed through a sharp FIR filter to remove images arising from the expander, and then two identical cascaded FIR systems with more relaxed cutoffs. The final filter-interpolator pair responsible for the bulk of the oversampling is a cascaded integrator-comb filter as described by Hogenauer [2] . The CIC-interpolator is a cascaded FIR-IIR system which nulls imaged components while allowing the passband through with only minor attenuation.

An initial survey of the literature suggested a CIC-interpolator to accomplish all oversampling. However (see below) the CIC-interpolator works on the assumption that the input signal occupies a very limited band – the input here, by contrast, occupies $-\pi$ to π. Thus initial stages of upsampling were necessary to reduce the relative bandwidth of the input signal.

III. Interpolator

The high oversampling factor necessitates a fantastically sharp low-pass filter if the oversampling is done at once. However, a series of three 2x stages with progressively less-demanding FIR anti-imaging low-pass filters, followed by a cascaded integrator-comb structure at 16x or 32x, enables the desired response.

A. Half-band polyphase FIR interpolator design

All FIR filters used are *half-band* filters [3]. A half-band FIR filter meets the following criteria:

1) The passband and stopband are symmetric around $\frac{\pi}{2}$, i.e. $\omega_p + \omega_s = \pi$.
2) Passband and stopband have equal specified ripples, i.e. $\delta_p = \delta_s$.

For FIR systems with odd length, this results in every odd coefficient being zero except for the center coefficient, which is always $\frac{1}{2}$ (Fig. 3).

The savings are particularly evident given a polyphase implementation with upsampling, here using an oversampling factor of two. Note that the impulse response of an LTI system can be decomposed into even-samples and odd-samples (Fig. 5):

$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^{-n} \quad (1)$$

$$= \sum_{n=-\infty}^{\infty} h[2n]z^{-2n} + z^{-1} \sum_{n=-\infty}^{\infty} h[2n+1]z^{-2n} \quad (2)$$

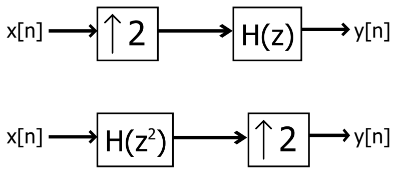


Fig. 4. The noble identities:the above two systems are equivalent.

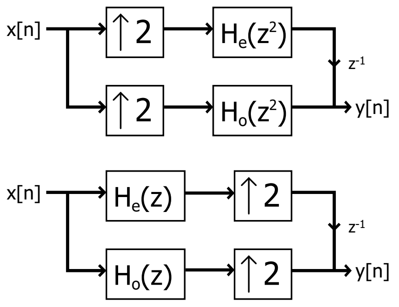


Fig. 5. Polyphase decomposition using noble identities

So if we allow

$$H_e(z) = \sum_{n=-\infty}^{\infty} h[2n]z^{-n} \quad (3)$$

$$H_o(z) = \sum_{n=-\infty}^{\infty} h[2n+1]z^{-n} \quad (4)$$

Then we see that $H(z) = H_e(z^2) + z^{-1}H_o(z^2)$. Now, the noble identities [5] (Fig. 4) allow us to combine the above decomposition with upsampling, yielding the identity shown in figure 5. The resulting system has the same upsample-filtering properties as the original, but with half the number of multiplies. The interpolated outputs have zeros for every other sample, so the delay element can be replaced by a commutator (Fig. 6) switching between $y_e[n]$ and $y_o[n]$ at $2f_s$, i.e. at twice the sampling rate [4].

B. Pipelined Half-band filter hardware

The structure for filter implementation is general (Fig. 7) to allow reuse for the three cascaded twice-oversampling half-band filters. The two polyphase components of the half-band system ($h_e[n]$ and $h_o[n]$ for the even and odd coefficients, respectively) are particularly efficient, as the odd coefficient vector is all zeros except for the center coefficient of $\frac{1}{2}$ which can be implemented as a right-shift.

The resulting system takes in samples at Fs_l and outputs them at $Fs_h = 2Fs_l$. Each filter system has a circular buffer for storage of the samples (implemented as one 4096-bit segment of BlockSelect+ RAM), and a similar RAM segment for a coefficient vector. This coefficient RAM only needs to store $h_e[n]$, and as $h_e[n]$ is inherently symmetric, only needs to store the unique $M/2$ coefficients.

To process an input sample, the system stores a new sample in the circular buffer. Then dual index pointers $xoff_l$ and $xoff_h$ read

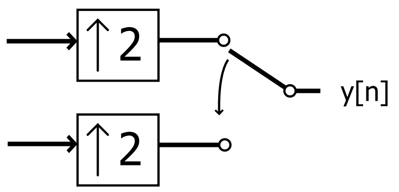
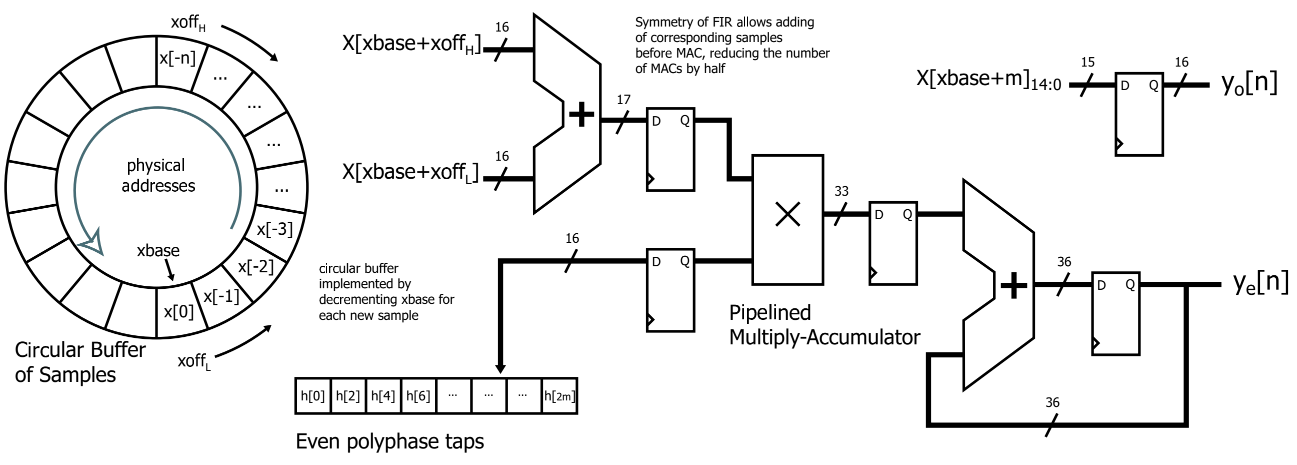


Fig. 6. Delay replacement with a commutator



xoff_H
x[-n]
physical addresses
xbase
x[-3]
x[-2]
x[-1]
x[0]
Circular Buffer of Samples
xoff_L
X[xbase+xoff_H]
X[xbase+xoff_L]
16
17
Symmetry of FIR allows adding of corresponding samples before MAC, reducing the number of MACs by half
circular buffer implemented by decrementing xbase for each new sample
h[0] h[2] h[4] h[6] h[2m]
Even polyphase taps
Pipelined Multiply-Accumulator
33
36
X[xbase+m]14:0
15
y_o[n]
y_e[n]

Fig. 7. Generalized hardware system to implement a 2x-expander and associated half-band FIR lowpass filter.

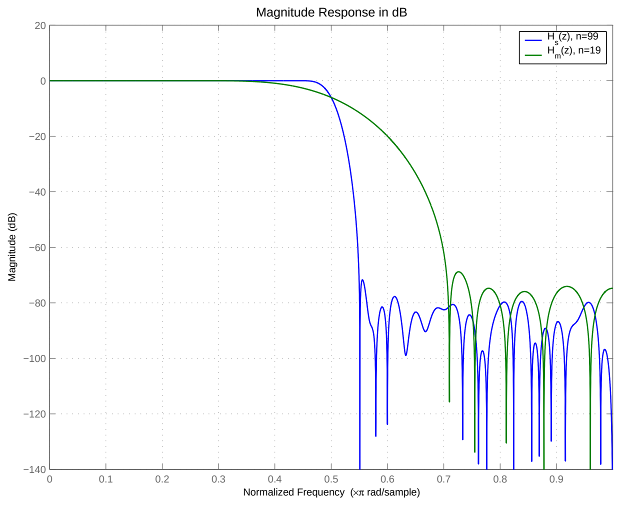


Fig. 8. Frequency response of the two FIR half-band lowpass filters for the interpolators, h_s is blue, and h_m is green. Both filters exhibit the half-band frequency response, that is, passband set at $\frac{pi}{2} - \omega_c$, stopband at $\frac{pi}{2} + \omega_c$

through the sample buffer, adding symmetric pairs of samples and then multiplying them by their corresponding coefficient. Simultaneous reads from the buffer are made possible by the dual-ported BlockSelect+ RAM on the Xilinx FPGA. The extra-wide multiply-accumulator can return an accurate result even if intermediate sums (for the entire series of MACs) would overflow.

Note that the $y_{even}[n]$ samples are the output from the MAC, whereas the $y_{odd}[n]$ samples are simply the single relevant sample, right-shifted one bit. The overall Fs_h output alternates between these.

The filter coefficients themselves were computed using `firls` least-squares implementation in MATLAB. Each interpolation stage compresses the relevant signal bandwidth into a smaller portion of the spectrum; thus each successive stage can have a slightly less sharp anti-imaging filter. Their frequency-response, following 16-bit coefficient quantization, can be seen in figure 8. Thus the first filter H_s has a length of 99, whereas the two later H_m filters have lengths of 19. Note that the second and third oversampling stages both use H_m as their LPF, due to implementation convenience – at the overall 40 MHz rate, there are clock cycles to spare.

The right-shift for $h_o[n]$ is an exact (ignoring rounding error) division by two, whereas the repeated MACs frequently would not sum to $\frac{1}{2}$ due to coefficient rounding effects. The result was ringing in the step response that would not die out. Thus filter length was determined to be that which, using rounded coefficients and `firls`, brought $\sum h_e[n]$ as close to $\frac{1}{2}$ as possible.

C. Cascaded Integrator-Comb Interpolator

Hogenauer [2] described a novel type of filter for interpolation and decimation of signals subjected to high sampling-rate changes (Fig. 9). The resulting cascaded integrator-comb is optimized for removing images from up/downsampled spectra, using a minimum of hardware.

The interpolator implementation consists of a cascade of N comb filters of the form

$$H_C(z) = 1 - z^M \tag{5}$$

followed by a series of N post-expansion integrators of the form

$$H_I(z) = \frac{1}{1 - z^{-1}} \tag{6}$$

Assuming an expansion by a factor of L, a cascade of N combs and N integrators has a frequency response

$$\begin{aligned} H_{CIC}(z) &= H_C^N(z^L) H_I^N(z) & (7) \\ &= \frac{(1 - z^{-LM})^N}{(1 - z^{-1})^N} = \left[\sum_{k=0}^{LM-1} z^{-k} \right]^N & (8) \end{aligned}$$

noting the $H_C(z^L)$ arises via the noble identities. Thus the overall system is FIR.

The resulting frequency response looks like figure 10 for our selected parameters, using 128x oversampling ($L = 16$, $N = 4$, $M = 1$). Note the nulls centered at multiples of $\frac{pi}{L}$. Unfortunately, the CIC-interpolator also rapidly begins attenuating frequencies outside a narrow lowpass region – this is the reason we must first oversample by a factor of 8 in our system. Even so, there is minor attenuation of the higher portions of our original passband, necessitating the previously-discussed compensation filter. Behavior is very similar for 256x oversampling.

Hogenauer's innovation can be implemented in a minimum of silicon – his original design used modular 4-bit combs and 4-bit integrators. Using the hardware of the Xilinx FPGA, a 4-bit integrator takes one CLB and a 4-bit comb takes two. The integrator stage cannot tolerate rounding without the error variance increasing boundlessly, resulting in instability. To compensate, each integrator

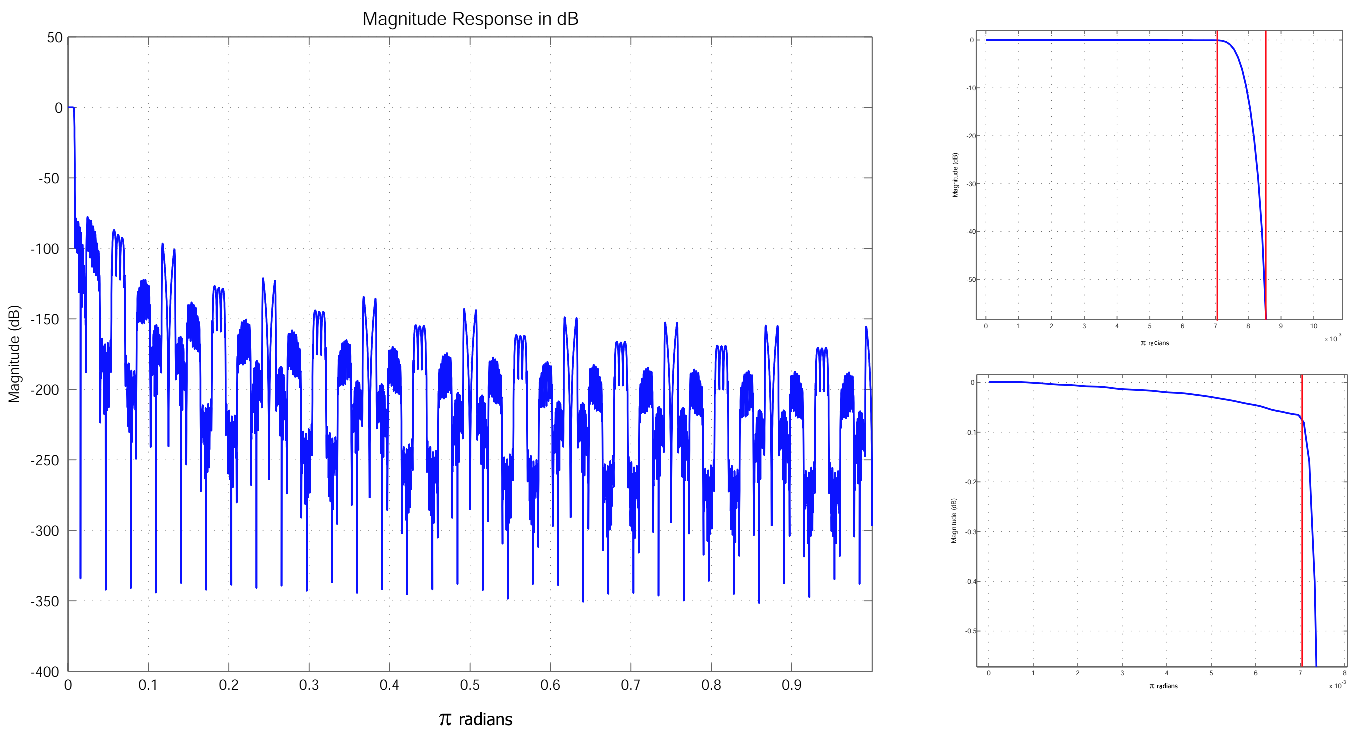


Fig. 11. Complete frequency response for FIR 128-times interpolation filter. Note that original specifications called for $\omega_p = 0.45*2\pi$, $\omega_s = 0.55*2\pi$. With 128x oversampling, these become (normalized by π) $\omega_p = 0.00703$, $\omega_s = 0.008594$, shown in red. Left: total frequency response over entire $[0, \pi]$ range. Upper right: frequency response over transition band. Lower right: Frequency response in passband to measure passband ripple. Appropriate pass/stopband frequencies shown in red.

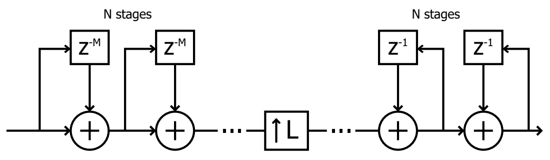


Fig. 9. Cascaded integrator-comb architecture

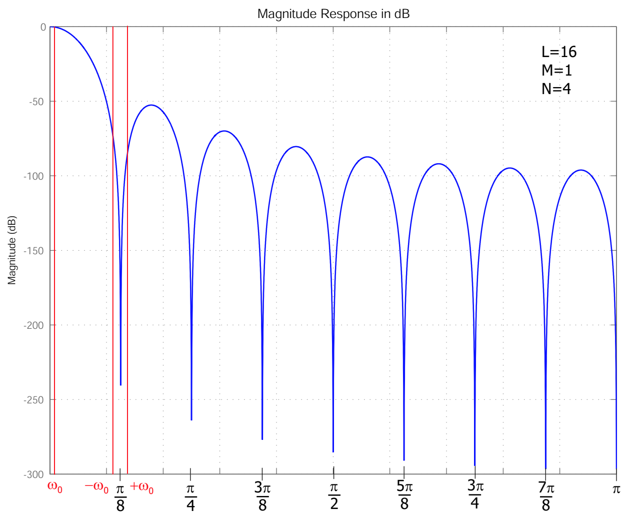


Fig. 10. CIC interpolator frequency response. $N = 4$, $L = 16$, $M = 1$. Note the nulls where the images of the original signal would be. The figure above assumes a signal bandwidth of ω_0, indicated in red. Note that the CIC passband centered at $\omega = 0$ is only flat for a small region around $\omega = 0$

stage must be sufficiently larger than the previous to avoid overflow and eliminate rounding. The algorithm presented [2] is beyond the scope of this paper, but resulted in a cascade of integrator sections 24, 30, 36, and 40 bits wide. The result, however, is that the only truncation/rounding noise occurs at the output to the last integrator, due to the lack of multiplies. The two different upsampling ratios inside the CIC (16x and 32x) necessitate selecting different bits of the CIC output as input to the noise shaper.

D. *Complete Interpolator Response*

Note that original specifications called for $\omega_p = 0.45 * 2\pi$, $\omega_s = 0.55 * 2\pi$. With 128x oversampling, these become (normalized by π) $\omega_p = 0.00703$, $\omega_s = 0.008594$, and with 256x oversampling, $\omega_p = 0.003515$, $\omega_s = 0.0042968$ (Fig. 11).

The total response is shown using quantized coefficients – the only other system artifacts will arise from quantization noise and potential overflow effects.

IV. Noise Shaping

Any system for producing analog output from digital input will create artifacts in the signal from the inherently quantized output. Noise-shaping is a technique of using feedback to significantly lessen the effects of these artifacts in the passband. We adopt the conventional linear quantization noise model (Fig. 12a), replacing the non-linear quantizer with additive white noise distributed between $-\frac{\Delta}{2}$ and $\frac{\Delta}{2}$, where Δ is the quantization step size. The resulting noise has a constant power-spectral density of $\Phi_{ee}(e^{j\omega}) = \sigma_e^2 = \frac{\Delta^2}{12}$

It can be shown that oversampling results in an increase in the signal to quantization noise ratio (SQNR), measured as the ratio of signal variance to noise variance. This is equivalent to 3 dB for each oversampling factor of two [5] – effectively an extra bit in resolution for every fourfold increase in oversampling. The above oversampling

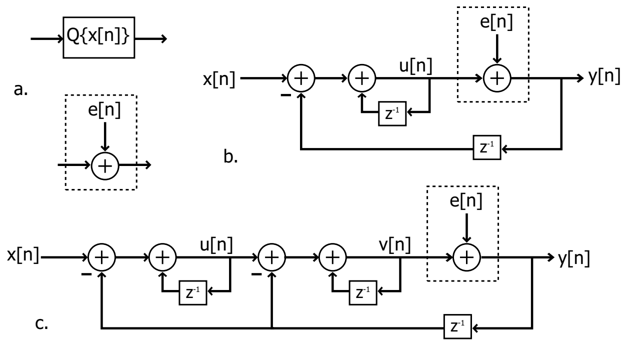


Fig. 12. Noise shaping: a. linear quantization noise model b. first-order noise shaper with quantization noise model c. second-order noise shaper with quantization noise model

system would thus have roughly 3.5 bits of resolution. This is referred to henceforth as "zero-order" noise shaping.

Quantization noise power remains constant regardless of oversampling rate. Oversampling reduces the DT spectrum bandwidth (which is limited to a region of 2π) occupied by a given signal, so it becomes easy to filter out the higher-frequency noise with inexpensive analog reconstruction filters. Oversampling effectively "spreads out" the noise over a larger spectral area relative to the signal of interest – thus the final filter will remove more noise, reducing noise power and increasing SNR.

A. *First-order Noise Shaping*

The first-order noise shaping system is shown in figure 12b with the quantizer replaced by the linear noise model. The signal $u[n]$ is the output of the integrator stage. Assume $Y[n] = Y_d[n]+Y_e[n]$, that is, the sum of the output error and the desired output due to $x[n]$. Then

$$Y_d(z) = U(z) = X(z) + z^{-1}U(z) - z^{-1}U(z) = X(z) \quad (9)$$

Thus the input passes unaffected through to the output. $Y_d[n]$ can be shown as follows:

$$\begin{aligned} Y(z) &= E(z) + U(z) & (10)\\ U(z) &= z^{-1}U(z) - z^{-1}(U(z) + E(z)) & (11)\\ &= -z^{-1}E(z) & (12)\\ Y(z) &= (1 - z^{1})E(z) & (13) \end{aligned}$$

The noise transfer function (NTF) of the system is thus $1 - z^{-1}$. As $e[n]$ is white, the output noise spectrum $\Phi_{y_e y_e}(e^{j\omega})$ is $\sigma_e^2|H_{NTF}(e^{j\omega})|^2$ or

$$\Phi_{y_e y_e}(e^{j\omega}) = \sigma_e^2[2sin(\omega/2)]^2 \quad (14)$$

B. *Second-order Noise Shaping*

The second-order noise shaper shown in figure 12c passes the input unadulterated, but shapes the noise still further:

$$\begin{aligned} Y(z) &= E(z) + V(z) & (15)\\ V(z) &= -z^{-1}E(z) + U(z) & (16)\\ U(z) &= (z^{-2} - z^{-1})E(z) & (17)\\ Y(z) &= (1 - z^{-1})^2 & (18) \end{aligned}$$

This yields a noise-transfer function of the form

$$\Phi_{y_e y_e}(e^{j\omega}) = \sigma_e^2[2sin(\omega/2)]^4 \quad (19)$$

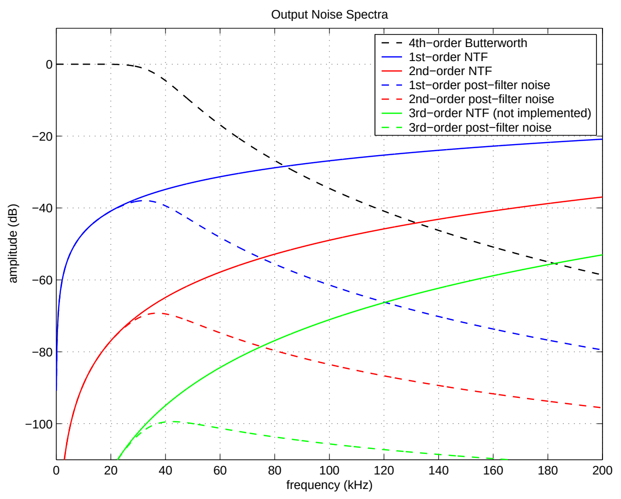


Fig. 13. Analytic plots of noise transfer functions and resulting output noise following a four-pole Butterworth filter

TABLE I
XILINX SPARTAN-II DEVICE UTILIZATION

Component	Number	Percent
Number of SLICEs	1785 out of 2352	75%
Number of BLOCKRAMs	8 out of 14	57%
External GCLKIOBs	1 out of 4	25%
External IOBs	21 out of 140	15%
Number of DLLs	2 out of 4	50%

C. *Noise-shaping results*

Noise shaping increases the total noise power but moves it away from $\omega = 0$. Since oversampling has the effect of scaling the input bandwidth so that it takes up less of the total $[0, \pi]$ spectrum, it is complementary to noise shaping – to achieve a given SQNR, lower order noise-shapers require greater oversampling and vice versa.

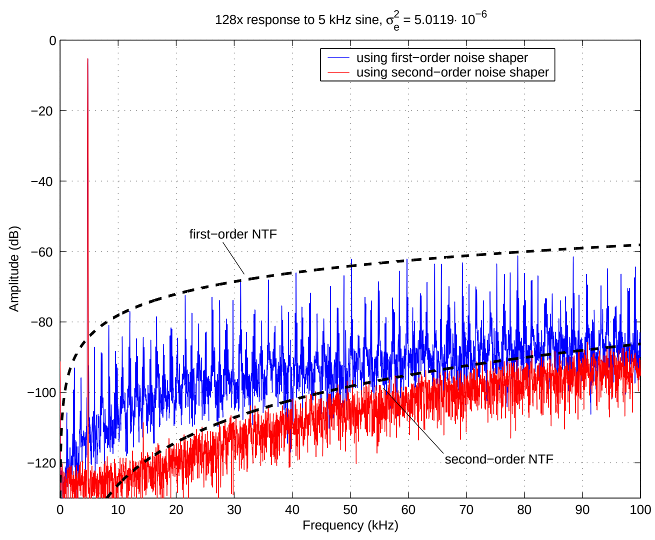
Noise-shaping only produces benefits if the output quantization noise can be removed by an analog filter. In figure 13 we plot the analytic noise transfer functions (in dB) for first, second, and third-order NTFs, and then the analytic output noise spectra assuming the post-DAC analog filter was a 4-pole Butterworth with $\omega_{-3dB} = 22kHz$. Although a third-order noise-shaper was not implemented, with the given analog filter it would be necessary to achieve true 16-bit resolution (96 dB SQNR).

V. IMPLEMENTATION RESULTS

The FPGA implementation was developed using VHDL, a hardware-description language for digital systems. The final digital behavior of the system can be simulated, including effects of propagation delay, setup-and-hold timing violations, and device temperature. The following numerical results for this implementation were created via simulation of the final FPGA design. This obviously only looks at digital behavior, neglecting the post-DAC analog filter.

A. *Simulated Device Performance*

VHDL simulation yields the plots shown in figure 14. The first plot contrasts the performance of first- and second-order noise shapers at an oversampling ratio of 128. Input was a half-scale 5 kHz sinusoid, and is clearly visible at 5 kHz. The analytic projections for the noise shaping closely match measured results. SNR ofthe overall system is dependent on performance of the post-DAC analog filter; however, we can measure performance here by assuming an ideal low-pass filter



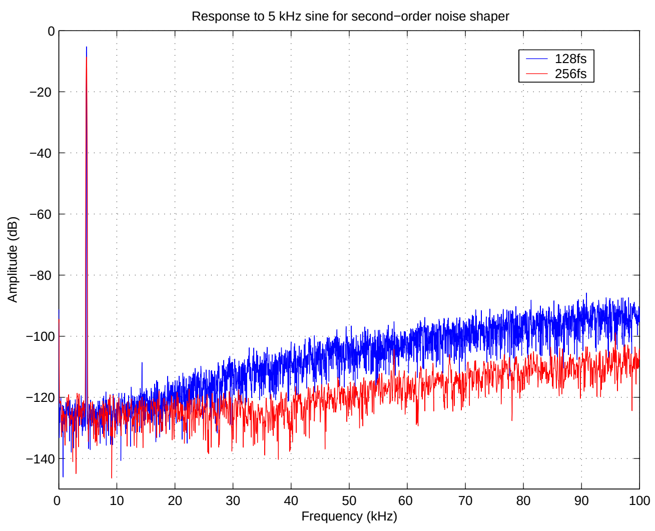


Fig. 14. Left: Output from VHDL behavioral simulation of DAC with 128-times oversampling. Single peak is original 5 kHz half-scale sine. Analytic results for noise-transfer functions are shown in black. Right: comparison of second-order noise shaping output for both 128-times and 256-times oversampling. Flat region close the noise floor of earlier processing systems.

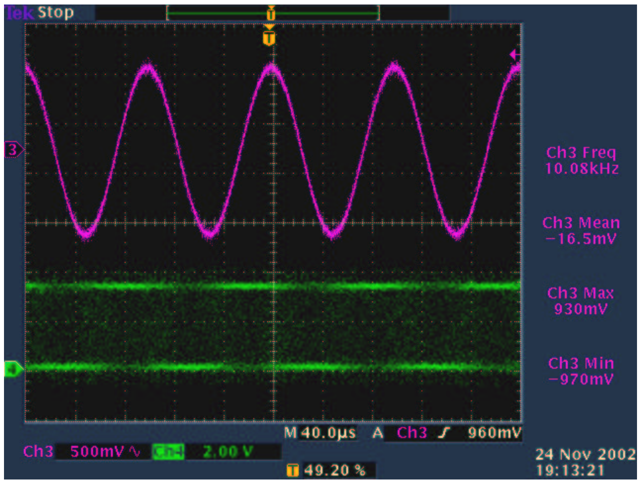


Fig. 15. Oscilloscope plot of second-order noise shaper at 128x oversampling. Top trace is output following low-pass filter; input waveform was half-scale 5 kHz sinusoid. Bottom trace is 1-bit DAC output. Note intensity changes corresponding to sine peaks and troughs – intensity variations slightly precede sine output due to group delay of filter.

with $\omega_c = 32$kHz. This gives the first-order shaper (at 128-times oversampling) an SNR of 64 dB, and the second-order an SNR of 94 dB.

The second plot shows the affect of oversampling ratio on the second-order noise shaper. The $128 f_s$ system has the above-indicated SNR (94 dB) and the $256 f_s$ system an SNR of 110 dB. In both cases, the SNR is limited by the quantization noise floor from previous stages.

B. Actual Analog Output

The post-DAC analog filter used was a fourth-order cascade of two second-order Butterworth filters using a Texas Instruments TLV2782. This device was selected due to its rail-to-rail capability on both the input and the output, and its 8 MHz bandwidth. Each second-order section uses a Sallen-key implementation with $F_{-3dB} = 22kHz$ [6].

Figure 15 shows an oscilloscope screen capture for a 50% fullscale 5 kHz sinewave input. The bottom trace is the 1-bit output from the FPGA; even though the oscillations are far too rapid to be seen on this timescale, note that their overall intensity correlates with the peaks and troughs of the post-filter sinusoid.

VI. Conclusion

This implementation of a one-bit digital-analog converter in commodity FPGA hardware has been a wonderful learning experience, especially because it actually *works*. You can hear the tremendous difference in sound quality when the noise-shaper is engaged.

The two potential sources of non-idealities in the system result in barely-audible differences when using different combinations of noise-shapers and oversampling ratios. First is the non-linearity of the output DAC – that is, the output pin of the FPGA. Parasitic capacitive and inductive effects cause very noticeable ringing at the input to the filter, substantially lessening the actual SNR. Additionally, nonlinearities in the Butterworth filter can cause some higher-frequency noise to alias down into the passband, further degrading the SNR.

The VHDL simulations confirm the system works as it should, with expected performance for the different parameters. Real delta-sigma converters typically use switched-capacitor implementations for the one-bit DAC which are capable of delivering exact quantities of charge and thus have much more linear response, and careful effort is made to minimize both clock feed-through and jitter, which both can lessen the overall SNR.

References

[1] *Spartan-II 2.5V FPGA Family Data Sheet*, 2nd ed., Xilinx, Inc., November 1991.
[2] E. B. Hogenauer, “An economical class of digital filters for decimation and interpolation,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-29, no. 2, pp. 155–162, April 1981.
[3] L. D. Milic and M. D. Lutovac, *Multirate Systems: Design and Applications*. Idea Group Publishing, 2002, ch. Efficient Multirate Filtering.
[4] R. E. Crochiere and lawrence R. Rabiner, *Multirate Digital Signal Processing*. Prentice-Hall, 1983.
[5] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Prentice-Hall, Inc, 1999.
[6] P. Horowitz and W. Hill, *The Art of Electronics*, 2nd ed. Cambridge Univeristy Press, 1989.