The Implementation Of Digital Filters For High Fidelity Audio

JON DATTORRO

ENSONIQ Corporation, Malvern, PA 19355, USA

In Part I, the problems are described which the practicing engineer encounters who unwittingly approaches the realization of IIR digital filters for the first time. It is assumed that suitable design programs are available to calculate the coefficients, and it is desired only to implement the filter. Elegant solutions are provided for some of the most intimidating problems typically encountered, which are: 1) input scaling requirements, 2) truncation noise propagation and recirculation, and 3)accurate low critical-frequency filtering. It is shown that the Direct Form I noncanonic topology is the best for use in the digital filtering of audio, and while 16/32-bit DSP chips such as the TMS32010 or the ADSP-2100 can be used in many high-fidelity applications, they will not meet the most demanding requirements. In Part II, we cover the DSP theory and the VLSI circuit implementation of a one-stage multirate 64:1 FIR decimator for use in one-bit Sigma-Delta A/D applications

PART I - IIR

THE IMPLEMENTATION OF RECURSIVE DIGITAL FILTERS

I-0 INTRODUCTION

An expanded version of PartI of this paper, covering recursive filters, was previously published in the Journal [39] this past November. Although there is some new material here delving further into Truncation Error Cancellation, we will cover only the most important concepts, but we will not discuss them in as much detail. Wherever possible the same figure, equation, and reference numbers have been used as were in the Journal.

Part II, which concerns itself with one-bit signal-width FIR implementation for Sigma-Delta A/D applications, is new material.

The design of digital filters and the implementation of digital filters can be carried out as two separate tasks. The design procedure involves the generation of the floating point coefficients, whereas the implementation involves the choice of topology, coefficient and signal wordlength, and handling of truncation (or rounding) effects. In this paper we will deal only with the implementation (Consult [1] for design aspects.).

The word 'digital' as applied to audio means discrete both in time and amplitude. Digital filters, then, are inherently nonlinear devices. A DSP (Digital Signal Processing) engineer must recognize and resolve aberrations from linear behaviour.

Coefficient quantization, of itself, does not induce nonlinear behaviour. The foremost aberration in digital filtering is due to truncation noise. Truncation noise arises whenever a numerical result or operand must be foreshortened to meet the limited precision of some element input, be it a multiplier input or a succeeding filter stage. Truncation introduces error, hence nonlinearities, by lowering mathematical precision. Since the source of these errors is usually known, we can write deterministic equations characterizing them in both the time and the frequency domains. Analysis in the frequency domain is essential to obtaining an intuitive grasp of the problem. The outcome of the analysis should lead to a means of minimizing the impact of truncation noise and perhaps some other associated nonlinear effects.

Overflow is a phenomenon which occurs when numerical calculations within a fixed point filter exceed the largest number physically representable; vice versa for underflow. If overflow is to be prevented at all cost, then the dynamic range of the digital filter will be overly constrained to be less than that of the input signal. This is because the input signal would need to be attenuated (scaled) prior to its entry to the filter circuit. This scaling is undesirable because the attainable S/N at the filter output becomes compromised. We will find a topology in Section I-1.1 which can deal with overflow using no scaling.

Filter designs having extreme critical frequency ("center" or "cutoff" frequency near 0 or π) are difficult to implement unless the coefficient wordlength is adequate (about 24 bits). Currently, 24 bit processors are available,

such as the MotorolaDSP56000 series, which alleviates this problem. If higher precision is needed, "residual coefficient coding" can be used

I-1 REVIEW OF TOPOLOGY

Figure 1 shows the Direct Form I second order digital filter. It has only one accumulator, hence only one source of truncation error. This error appears at the output of the accumulator and sounds like colored noise. The error occurs because the accumulator is capable of producing 32 bit results but the multiplier and succeeding stage can only accept 16 bit inputs. Most DSP processors now have N-by-Nbit multipliers which produce 2N-bit results which the accumulators accept directly. So, the truncation error is restricted to the feedback paths in the Direct Form I. If we could find a way to control the truncation noise recirculation, this topology might be workable.

Let us establish the term "unity gain filter", a filter whose transfer function deviates in magnitude from unity but is usually less than unity. Some examples of unity gain design are shown in Figure6. There is an apparent contradiction in the boost filter in Figure6, but we can argue that the boost raises low level signals to unity level or less at the filter output. If we were to design a unity gain filter then we would expect the output level not to exceed unity, much of the time, therefore output overflow would not be a problem. If we could tolerate intermediate overflows in the accumulator, which sometimes occur during calculations prior to final output, then we would have no need to scale the input signal. If both truncation and intermediate overflow were controllable using the Direct FormI, it would be a workable topology. We will later see that we can gain control over both these phenomena using Error Spectrum Shaping (ESS) and Jackson's rule, respectively.

Figure 2 shows the Direct Form II (canonic) topology. The same coefficient set produces the same filter transfer as for the Direct Form I. Using the Direct Form II, we would have two sources of truncation error to deal with; one at the output of each accumulator, permeating both the feedforward and feedback paths this time. Note that the state w(n) sees only the amplification of the input signal by the system poles (the recursive part, b_1 and b_2); w(n) can become quite large as a result. There is an overflow problem here due to the five multiplier inputs that w(n) feeds; these inputs cannot tolerate overflowed operands. So, we are forced to limit the magnitude of w(n) which can only be done by scaling the input signal, x(n).

Notice that when either the Direct Form I or II topologies are cascaded, they begin to look the same. High order filters are most often designed by cascading second order sections (or 'stages') because the Direct Form realization of a high order filter would result in a geometric growth in the problems we are dealing with. In audio, it is customary to see Graphic and Parametric Equalizers constructed as arrangements of second order sections where each section forms a complete filter. The lowpass bound on the transition region of 6dB per octave per conjugate pole pair does not exist in Moorer's second-order designs [1]. For these reasons, this paper deals exclusively with second order digital filters.

Figure 3 shows the transposes of the Direct Forms I and II. How many truncation error sources are there? How does each topology handle intermediate overflow in unity gain designs? Figure 4 shows more second order topologies. Does Direct FormII Transpose have the same overflow and truncation properties as Direct Form I? (Yes.)

I-1.1 OVERFLOW AND JACKSON'S RULE

Leland B. Jackson [3] [5] has demonstrated an interesting property of 2's complement arithmetic; namely, it is a modulo arithmetic. In terms of digital filters this means that an accumulator may be allowed to experience intermediate overflow, without necessarily having physical headroom bits, and still yield the correct result. This will work if it is known ahead of time that the final accumulation result is bounded by the physical wordlength of the accumulator; in DSP we call this unity gain design. Some examples of unity gain design are shown in Figure 6.

An example of Jackson's rule is given in Figure 5. A corollary to Jackson's rule allows overflowed operands as input to the accumulator. It would be nice if there were a similar rule for multipliers (this would be a good research topic).

In summary, using Jackson's rule and the Direct Form I we can eliminate the need for input scaling, using the unity gain design criterion, since we have infinite headroom in the accumulator.

I-1.2 FLOATING VERSUS FIXED POINT

At this point, interested readers may query that a floating point processor might obviate this overflow problem; this is not necessarily true. As overflow begins to occur, the floating point processor keeps track of the expanding MSBs of a large intermediate accumulation. It does so at the expense of the LSBs which get thrown out. In contrast, an infinite headroom fixed point accumulator keeps track of the LSBs throughout all intermediate overflows at the expense of the expanding MSBs which, usually, all become redundant at the final accumulation. The floating point accumulator would thus introduce truncation error into the LSBs whereas the fixed pointaccumulator would not.

One place where a floating point processor would become handy is at interstage boundaries where, for high order filters consisting of cascades of second order stages, output overflow is likely to occur at any stage. Input scaling is generally required for fixed point high order filters at each stage because unity gain design can only be employed for the whole filter. A floating point processor would eliminate the interstage scaling requirement, then, for high order filters. Even so, we always want to use Jackson's rule and fixed point arithmetic in the accumulator within each individual stage so that we can take advantage of the infinite headroom and LSB retention.

Floating point processors will neither solve the truncation noise problem. Truncation noise abatement summons the greatest mantissa precision. In terms of truncation noise

recirculation, the problem must be dealt with in exactly the same manner regardless of whether a 24-bit fixed point processor or a 24-bit with 8-bit-exponent floating point processor is used.

I-2 TRUNCATION NOISE PROPAGATION

Figure 8 shows the truncation noise situation in a Direct FormI filter. The only recirculation of error occurs in the feedback paths. The truncation error signal, e(n), emanating from the truncator box, Q, is a signed quantity that represents the difference between the full precision output, y(n), and the truncated output, $\hat{y}(n)$;i.e.,

$$y(n) = \hat{y}(n) + e(n) \tag{6}$$

In traditional implementations, e(n) is usually discarded. In the frequency domain, the truncated output appears like so:

$$\hat{Y}(z) = X(z)(\Sigma\, a_i z\text{-}i) / (1 - \Sigma\, b_i z^{-i}) - E(z) / (1 - \Sigma\, b_i z^{-i}) \tag{9}$$

This equation (9) says that the truncation error, E(z), is amplified by the system poles whether the filter boosts or cuts! For many practical cases of high-fidelity audio filtering, this is a severe problem. Equation (9) will serve as our reference since this is the outcome if we do nothing about the output truncation error.

I-2.1 TRUNCATION ERROR FEEDBACK

The network of Figure 11 shows one solution to the problem of truncation error recirculation. As can be seen from the figure, the truncation error, e(n), is being delayed (saved) and then fed back into the circuit. The multiplier coefficients, K1 and K2, operating on the truncation error are trivial (having only one nonzero binary digit) and, therefore, do not use the hardware multiplier. The effect that this error feedback has on the truncated output can be more easily seen in the frequency domain where,

$$\hat{Y}(z) = X(z)(\Sigma\, a_i z^{-i}) / (1 - \Sigma\, b_i z^{-i}) - E(z)(1 - \Sigma\, K_i z^{-i}) / (1 - \Sigma\, b_i z^{-i}) \tag{13a}$$

As can be seen from (13a), the coefficients, K_i, only operate on the truncation error and do not adversely affect the transfer of X(z) in any way. Through a discerning choice of the K_i, we can place zeroes in the truncation noise transfer (in the "error function" [39]) right on the unit circle to completely squelch audible truncation noise in the immediate vicinity of those zeroes. A tabulation of viable K_i versus their normalized frequency region of impact is given in Table 1. In the table, "once/twice" refers to the number of times an error feedback zero is encountered in the evaluation of the truncation noise transfer as the upper half of the unit circle is traversed in the z-plane. In the case of second order error feedback, the number is usually once; it can only be twice when the conjugate zero is in close proximity. A popular choice of zero location is at θ=0 (DC) because then the impact in the audible frequency region is doubled. The same is true for θ=π although not as popular a choice.

TABLE 1
ERROR FEEDBACK ZERO LOCATIONS FOR THE POSITIVE TOPOLOGY

K_1	K_2	REGION(θ)	
+2	-1	0	twice
-2	-1	π	twice
+1	-1	$\pi/3$	once
-1	-1	$2\pi/3$	once
+1	0	0	once
-1	0	π	once
0	+1	0 and π	once
0	-1	$\pi/2$	once

The region governed by the K_i should be chosen to lay closest to the region of amplification by the filter poles. This Truncation Error Feedback technique is quite powerful and economical and has been used in the digital playback circuitry of commercial compact disc players. THD+N measurements contrasting non-error feedback circuitry will show tens of decibels disparity. Further references to this technique can be found in [39] and the topic is collectively referred to as Error Spectrum Shaping (ESS).

I-2.2 TRUNCATION ERROR CANCELLATION

The audio engineer will observe an opportunity in equation (13a) to set the K_i equal to the b_i which has the effect of squelching the truncation noise to zero across the entire audio band. In this case, mostly all that remains of the truncation noise in (13a) is the truncation error at the output itself, E(z) (3dB in magnitude at the 16 bit level), which does not feed back into the circuit and so does not become amplified.

Figure 11-II shows the complete truncation error situation. This Truncation Error Cancellation circuit is an outgrowth of ESS and very much resembles a double precision implementation. The K_i are no longer trivial so the hardware multiplier must be used. Notice that a new second degree source of truncation error, $e^{(2)}(n)$, has been introduced as a result of the non-trivial error feedback multipliers. This new error source arises because the error feedback multiplications now produce 32 bit products. We must truncate the 16 LSBs when we combine the MSBs of the error accumulation with the least significant word of the signal accumulation. Since this truncation of the truncation-error-accumulation is ideally 32 bits below full scale, we can expect this second degree error to reside at approximately -186dB in level. We could justifiably ignore $e^{(2)}(n)$ for many applications, but we can never ignore the output truncation error, e(n), for serious digital filter work.

This second degree truncation error will only become a problem if the amplification of it by the system poles raises it above the signal noise floor (-90dB). Let's see what is happening in Figure 11-II in the frequency domain. If we define the truncation error accumulation, $\varepsilon(n)$, as we did

for the filter output signal,

$$\varepsilon(n) = \hat{\varepsilon}(n) + e^{(2)}(n) \quad (14a)$$

where,

$$|e^{(2)}(n)| << |e(n)| \quad (14b)$$

then the truncated output signal in Figure 11-II has the frequency domain representation,

$$\hat{Y}(z) = X(z)(\sum a_i z^{-i}) / (1 - \sum b_i z^{-i}) - E(z) - E^{(2)}(z) / (1 - \sum b_i z^{-i}) \quad (17a)$$

where E(z) is the output truncation noise at the 16 bit level, and $E^{(2)}(z)$ is the truncation-error-accumulation truncation noise at the 32 bit level. If we were to use a 24 bit processor, we could expect $|E^{(2)}(z)|$ to be at least 234 dB below unity (i.e., at the 40 bit level) before it experienced amplification by the system poles. In this case, we could rightly ignore it for most all applications. It is for this reason that a 24-bit processor in conjunction with Truncation Error Cancellation is recommended for high fidelity digital audio work.

I-3 NONLINEAR PHENOMENA

I-3.1 OVERFLOW OSCILLATIONS

If we do nothing about overflow when it occurs at the output of a digital filter (we are not talking about intermediate overflows), then the circuit will enter into a nonlinear state. If and when it appears to fully recover, the digital filter may still be trapped in a weakly nonlinear mode; unwanted oscillations may be autonomously present. The overflow oscillation problem is completely solved by saturating the filter output upon detection of overflow there, as in Figure 15.

The proper way to detect output overflow is by examination of the MSBs of the output accumulator at and above the first sign bit [39,Appendix2]. If the MSBs are different, overflow has occurred (the output is not within the first modulo) and the proper sign is probably that of the accumulator MSB. The MSBs directly indicate the current modulo. If there were physical headroom bits, the detection of overflow would be their intended purpose.

We never want the accumulator to autonomously saturate because then we would not be taking advantage of Jackson's rule and infinite headroom. We discourage the use of traditional overflow detection employing the carry bit in the status register because it does not indicate when we have returned to the first modulo.

I-3.2 LIMIT CYCLES

Limit cycles are omnipresent autonomous oscillations caused by finite precision arithmetic in the signal feedback paths. This phenomenon is a concern with any new implementation. The possibility of the presence of limit cycles for a particular architecture can be predicted as a function of coefficient value, although it is not the coefficients themselves that activate the nonlinearity. Increasing the width of the feedback signal paths, physically or effectively (via Error Spectrum Shaping techniques), diminishes the amplitude of these oscillations. The results in [31] show that Truncation Error Cancellation eliminates limit cycles in second order digital filters; hence we solve both the truncation noise recirculation problem and the limit cycleproblem at once.

I-3.3 FILTER TRANSFER FUNCTION ANOMALIES

Another benefit of Truncation Error Cancellation or Feedback is that the filter transfer function becomes much less dependent on absolute signal level. Recall from linear circuit theory that a time-invariant filter transfer is theoretically independent of the input signal level. Using no error feedback however, transfer anomalies can be observed at low (constant) input-signal levels in digital filters; indeed, a time-varying response can be observed under the proper conditions for a fixed frequency input sinusoid. This nonlinear phenomenon is a direct consequence of the digitization of the audio signal in the filter.

I-3.4 FORCED OVERFLOW OSCILLATIONS

Once the digital filter output saturates (clips), under program control, its behaviour again becomes nonlinear. Figure15 shows only the feedback portion of a second order digital filter having a saturator at its output; the truncators and the feedforward paths are not essential to this discussion. Figure16 shows a typical time domain response of this second order digital filter overdriven by a sinusoid. The input is present for this type of response which explains the term 'forced'.

Unfortunately, the use of ESS does not solve this problem; neither does it exacerbate the oscillations. To return to normal behaviour, the input signal level needs to be reduced to well below that which elicited the response, in some cases. Figure 17 shows the stability triangle having the coefficient region shaded for which overflow oscillations are a potential problem. Although there appear nulls in the problem region, at this time it seems that the only viable solution is to reduce the input signal level. Further research is required in this area.

ACKNOWLEDGEMENT

We would like to extend special thanks to Abbie Cohen and Ingeborg Stochmal of the AES for the fastidious editing, and typeset of the equations in the original 1988 November Journal publication.

ERRATA

We would like to note further discussion of the original paper, published in the 1988 November Journal, which appears in a subsequent Journal Letters, dealing with the topic of second degree truncation error using Truncation Error Cancellation. There were also some errors which appeared in the original paper.

1) Table 1, pg.863.

Fourth entry: should read '$\pi/3$ Once'

(not 'Twice').

Fifth entry: should read '2 π/3 Once'
(not 'Twice').
A new (eighth) entry: $K_1 = 0$, $K_2 = -1$,
Region $\theta = \pi/2$ Once.

2) Section 2.5.1, prgh.2, line 15, pg.864.
Little Σ should be squared.

3) pg.870.
First equation at top left of page: should read
ci + binary code$(ec_i)/(2^{qe}2^{qe}) \equiv c_{Fi}$
An asterisk should appear in the caption for Fig.14
preceding 'See TMS ...'

4) Section 4.1.2, prgh.1, pg.872.
Delete lines 21 through 24; i.e. delete,
'worsens when... ...problem'

5) pg.874.
The approximation to h(n) equation should take the absolute value on the left hand side.

6) Section 1.2.2, last paragraph, line 11, pg.858;
should read: 'gain and less. The responsibility...'

PART II - FIR

THE IMPLEMENTATION OF A ONE-STAGE MULTI-RATE 64:1 FIR

DECIMATOR FOR USE IN ONE-BIT SIGMA-DELTA A/D APPLICATIONS

JON DATTORRO, ALBERT CHARPENTIER, & DAVID ANDREAS

ENSONIQ Corporation

II-0 INTRODUCTION

Sigma-Delta modulation is emerging as a preferred alternative to successive approximation techniques for analog to digital conversion [42]-[53]. The Sigma-Delta system can, conceptually, be divided into two distinct parts: the analog front end (the one-bit modulator), and the digital decimation filter. The decimation filter outputs the desired digital signal. This paper concerns itself with the implementation of the decimation filter only, when presented with a one-bit stream from the front end flash converter (a lone comparator). The advantages of the binary state signal are capitalized on in this design.

The attraction to Sigma-Delta A/D converters in terms of hardware, is the relaxed constraints on the input anti-alias filter and the lack of the need for a sample/hold circuit. The foremost theoretical reason for the preference of Sigma-Delta is the fact that [51] as the signal level goes down, the harmonic distortion increases at a much slower rate. This is primarily due to the superb linearity of the analog front end. In the one-bit case, the linearity easily exceeds that of the best successive approximation designs.

The Sigma-Delta process, in simple terms, spreads the quantization noise of a very low resolution flash A/D converter over a broad region covering several MegaHertz, and then shapes that noise via feedback of filtered quantized signal. The output of the low resolution A/D converter is presented to the decimation filter whose task it is to take the low resolution high speed samples and convert them to high resolution low speed samples.

II-1 THEORY

II-1.0 DECIMATION

Our one-bit A/D converter is running at 3.072 MHz. The desired sample rate is 48kHz.We then have a decimation ratio of 64. If we use an FIR filter to perform the decimation, then the current decimator output is not dependent on previous outputs because of the nonrecursive structure. There is, therefore, no filter output truncation noise recirculation to worry about. In the time domain we are allowed to literally throw away 63 out of every 64 output samples calculated. In fact, it is not even necessary to calculate those 63 intermediate samples. If we use only one FIR filter to perform the decimation from the 3.072MHz rate to the 48kHz rate, then we say that we are decimating in one stage.

Other commercial implementations [44] [47] [50] [51] of the decimator comprise several small moving average type FIR stages in cascade, operating at a much higher rate. This is a good approach but the attraction to the one-stage approach is the small area of silicon upon which a large ROM can be constructed (ROM is cheap), and the high alias rejection at the first foldover frequency (-110dB at 28kHz for a 48kHz sample rate). We have found that 2048 22-bit coefficients are required at 3 MHz to reach the theoretical performance level of a 16 bit A/D converter, which agrees with Adams'[47] assessment of about 4000 coefficients at 6MHz.

Figure 18 shows the process of decimation in the frequency domain. In Figure 18(a) a fictitious baseband audio spectrum is shown out to 3.072MHz with its first replication. The prime on the frequency argument denotes the high sample rate. Figure18(b) shows the FIR transfer. The original spectrum is multiplied by the FIR transfer at the high sample rate (not shown) corresponding to the convolution in the time domain. Note that while the stopband attenuation of the FIR is high, it is not absolute zero. We can infer that the quality of the decimation is somehow related to the absolute spectral level of that out-of-band (the 24kHz -> 3MHz region) material and the degree to which it becomes attenuated. This is true, since when we throw away 63 out of 64 samples, the spectrum in Figure 18(c) resultswhich shows aliasing as a result of the decimation. The aliases are shifted replications of the filtered 3.072MHz spectrum. We need to know the total accumulation of unwanted alias distortion. First note some incidentals concerning the aliases: 1) there are only 63 aliases into the audio baseband [40], 2) the baseband spectrum remains symmetrical after decimation.

II-1.1 DECIMATOR ALIAS NOISE

To determine the amount of alias distortion, we need to know whether the out of band signal is correlated or uncorrelated to the base band audio signal. If the out of band signal were correlated with the audio signal, the amount of aliasing noise in power spectral level could be as bad as $10\log(63^2 S_x)$ [22,chap.3-4], or 36dB over S_x, where S_x is the power spectral density of the FIR filtered out-of-band signal prior to decimation. Refer to Figure19. It becomes the job of the Sigma-Delta analog front end to make sure that the out of band signal is uncorrelated. In this case the amount of aliasing noise is approximately $10\log(63S_x)$, or 18dB over S_x. This alias noise, $63S_x$, is combined as the sum of the squares with the in-band pre-decimation noise power spectral density, S_q, to get the total post-decimation noise power spectral density in-band,

$$S_y = S_q + 63S_x. \qquad \text{(II-1)}$$

To reach the noise performance of a 16 bit converter, we need at least 90dB S/N in the 24kHz baseband. The noise power, N, refers to the integral of the total noise power spectral density, S_y. In the time domain, this means that the RMS level of the noise is such that only the LSB would ever be toggled in response to the noise alone. In the frequency domain it means that the power spectral density, S_y, should have a level of about -140dB [22,chap.6-2] with respect to a unity level sinusoid. Refer to Figure 20. The total noise power can be estimated in the frequency domain over a 24kHz bandwidth as follows:

$$10\log(N) = 10\log\left(\int_0^{24\,kHz} {}_0S_y\, df\right) = -96dB \qquad \text{(II-2)}$$

when $S_y = 10^{-140/10}$ [V^2/Hz]

We can now determine the required FIR attenuation. Referring to Figure 19, using (II-1) and realizing that

$S_x = 10^{(M+H_F)/10}$ [V^2/Hz]
then,

$$H_F \cong 10\log(S_y - S_q) - 18 - M[dB] \qquad \text{(II-3)}$$

;for $S_q < S_y$

where M is the level of the out-of-band pre-decimation modulator noise power spectral density in dB (about -34dB, [44], Figure 21), and H_F is the (negative) FIR stopband attenuation level in dB. When the noise contribution due to aliasing is only 18dB, and $10\log(S_q)$ is about -144dB, then we find from (II-3) that the required FIR stopband level, H_F, is about -126dB.

Figure 21 shows a simulated modulator output signal and noise floor in response to an input sinusoid. The simulation was performed in floating point arithmetic. The out-of-band noise power spectral density, M, is lower than our conservative estimate of -34dB, above. The bandwidth of this plot is 1.536MHz and so the signal, at exactly 1.500kHz, is scrunched up against the left hand side. Figure22 shows the audio band only, of the same modulator output. The in-band noise power spectral density, S_q, is a little higher than we would like but this is compensated in equation (II-3) by the lower out-of-band noise, M. Figure 23(a) shows the audio band of the simulated decimator output, post-decimation, in response to the modulator output of Figure 21. This part of the simulation was performed using all integer arithmetic. Figure 23(a) represents a 21 bit decimator output. Figure23(b) represents a 16 bit decimator output. The character (or correlation) of the noise floor after truncation to 16 bits depends totally on the modulator design which can be considered to be a pre-dithered noise shaping system. Harmonic distortion is more likely here because the 1-bit sinusoid frequency is a sub-multiple of 48kHz.

Figures 21, 22, and 23 are estimates of power spectral density [4,chap.11]. The size of the FFT required for adequate spectral resolution was 65536 points. Although our plots of (power) spectral density use decibels on the ordinate, they should not be confused with "noise power" which is the integral of spectral density.

II-1.2 DECIMATOR FILTER SHAPE

To get 126dB of attenuation requires at least 21 bits (assuming 6dB per bit) of resolution in the FIR coefficients. We can understand this intuitively by realizing that the FIR filter coefficients are quantized samples of the impulse response of the desired filter. If the quantization RMS noise floor of the coefficients exceeds the desired stopband level, then it is not likely that the filter will meet specifications. For example, in order that a one-bit signal be attenuated downward 21 bits, the mathematics at the 21 bit level must be accurate. Obviously, the greater the precision in the calculations, the more this will be true. The noise floor at 21 bits, then, is the lower bound, while some number of bits in excess of 21 becomes the upper bound on the number-of-bit criterion for accurate high attenuation filtering.

Another way to look at this is in analogy to IIR filters. Recall that the coefficient resolution of an IIR filter primarily determines deviation from the shape of the desired filter; the same is true for FIR. This can be seen easily by taking the Fourier transform of the digital impulse response.

In reality, the 21 bit impulse response does not utilize the whole quantization space and we can lose as much as about 4.4 dB from the theoretical limit. For this reason we will use 22 bit coefficients to guarantee 126dB attenuation. The 2048 quantized coefficients which comprise the FIR decimator are shown in Figure 24. The coefficients were calculated on a VAX8700 at ENSONIQ using a standard Parks/McClellan algorithm in quadruple precision, and about 1 hour of CPU time. The FIR transfer function is shown in Figure25; it was calculated using an FFT on the 22 bit coefficients. Note that the normalized passband width is only 0.0064 which yields a -3dB point at 22kHz, and the attenuation is 110dB at 28kHz. The transition region is therefore about 392dB

per octave. A blowup of the passband is shown in Figure26; the ripple is negligible.

II-1.3 FIR COEFFICIENT GAIN AND OFFSET

The floating point coefficients out of the Parks/McClellan algorithm are all much less than 1.0 in magnitude; interestingly enough, the algorithm produces a unity gain design which means that the gain is unity in the passband. This means that we can introduce a gain into the passband if we desire to compensate some system loss and/or to eliminate leading binary zeroes from the coefficients to increase their precision. If the minimum floating point value produced by the Parks/McClellan algorithm is called **min** (-0.00312), and the maximum value is called **max** (0.0148), then the maximum gain, g, that we could ever introduce while still maintaining 22 bit precision is

$$g \leq 1/(\text{max-min}) \; (\cong 55.7) \qquad \text{(II-4a)}$$

In our implementation we work with unsigned coefficients to simplify the hardware. In this case we need to add an offset,

$$d \geq -\text{min} \qquad \text{(II-4b)}$$

to the floating point coefficients prior to introducing the gain factor so as to make all the floating point coefficients positive and to maximize the utilized quantization space.

The normalized impulse response is then,

$$h_{norm}(n) = \{\, h(n) + d \,\}\, g \qquad \text{(II-4c)}$$

where h(n) are the floating point coefficients produced by the design procedure, d is the floating point offset, and g is the floating point gain factor. The floating point coefficients, $h_{norm}(n)$, are all non-negative as a result of the offset. They will later be encoded and then stored in ROM using 22 bits of precision but having no sign bit.

The desired (standard) floating point convolution is,

$$y_{des}(n) = \sum h(k)x(n-k) \qquad \text{(II-5)}$$

The calculation we will actually perform on chip is,

$$y(n) = \sum\{h_{norm}(k)x_{norm}(n-k) + C_0[1-x_{norm}(n-k)]\} - g/2 \qquad \text{(II-6)}$$

where, $x_{norm}(n-k) = 1$ or $0 = x(n) + 0.5$
$x(n) = -0.5$ or 0.5

Since the quantized signal, $x_{norm}(n)$, has only two states, we can force no symmetry about zero. It has, therefore, a DC offset of 0.5 which should be subtracted out of the accumulation. This is the purpose of g/2 which is subtracted after the completion of the accumulation. The second term in equation (II-6) involving C_0 will neutralize the term, dg, in the normalized coefficients in (II-4c). C_0 is the floating point representation of a positive constant whose value is chosen such that N (=2048) times C_0 exceeds the available accumulator dynamic range;i.e., in floating point,

$$NC_0 = (NI / 2^{22}) > g \qquad \text{(II-7)}$$

;for I a trivial binary integer (a binary integer having only one nonzero digit). The purpose of C_0 will be to trivially overflow the accumulator into another modulo; but always into the same place within the same modulo.

Expanding equation (II-6) we find,

$$y(n) = \sum\{[gh(n) + dg][x(n-k) + 0.5] + C_0[1 - (x(n-k) + 0.5)]\} - g/2 \qquad \text{(II-8a)}$$

$$= 0.5g \sum\{h(n)\} - g/2 + \sum\{gh(n)x(n-k) + 0.5dg + 0.5C_0 + [dg - C_0]x(n-k)\} \qquad \text{(II-8b)}$$

The first two terms vanish because the Parks/McClellan filter design is unity gain (the coefficient quantization produces a tiny DC offset). We can only rid the last term if

$$C_0 = dg \qquad \text{(II-8c)}$$

At this point we need to adjust d and g so that their product equals C_0 exactly, for C_0 constrained as in (II-7).If we do this, then (II-8b) becomes,

$$y(n) = g \sum\{\, h(n)x(n-k) \,\} + NC_0 \qquad \text{(II-9)}$$

Note that the multiplication of two trivial binary integers results in another trivial binary integer. Since NC_0 has been chosen to overflow the accumulator such that if the $x_{norm}(n-k)$ were all zero then the accumulator would be zero, then, in effect, NC_0 goes away and we are left with the desired convolution times a gain factor.

II-1.4 FIR ACCUMULATOR WIDTH

The accumulator size is not arbitrary; it must be chosen such that we know which bits out of the accumulator will be used as the output bits. Since the normalized filter uses unsigned Q22 coefficients having widths of 22 bits, and the Q0 signal is one-bit, then at least a 22-bit accumulator is required, following the rule: Q22XQ0= Q22, 22-bitS_x1-bit= 23-1 redundant (in this case, superfluous) sign bit = 22 bits.

If the gain, g, in (II-4c) were equal to 1, then a 22 bit accumulator would be sufficient because the filter design is unity gain. But, since the maximum allowable value of g, which will not demand greater than 22-bit coefficient precision, is 55.7 for our particular filter design (II-4a), we choose g to be 2^5 (=32). This particular value of filter gain, g, serves to eliminate 5 leading binary zeroes from the coefficients, h(n), and establishes the number of extra bits required in the accumulator to be exactly 5, which brings us

up to 27 bits. If there were a system loss to compensate, we could use a g of greater value (but less than 55.7), having no need to adjust the accumulator width.

Finally, we want one more (guard) bit in the accumulator which we will use for overflow detection. (The proper way to detect overflow is discussed in [39,Appendix 2] and in section I-3.1 here.) This brings us to the requirement of a 28 bit accumulator. The accumulator output comprises the MSBs excepting the guard bit.

II-1.4.1 BOOKKEEPING

Even though all the normalized coefficients were non-negative, equation (II-9) indicates that the result, y(n), is a signed quantity because both h(n) and x(n) are signed. After we account for the gain, g, the effective binary point in the accumulator lays between the guard bit and the 27th bit (calling the LSB the first bit). Since x(n) is bounded in magnitude by 0.5, and since the filter design was unity gain, then y(n) must obey the same bound. This is the purpose of the -g/2 term in equations (II-6) and (II-8). Specifically,

$$-0.5 \le y(n)/g < 0.5$$

We can now move the binary point one place right, effectively multiplying by two, recovering the sign bit. The 28 bit accumulator ends up in Q26 format.

II-1.5 SPECIFIC VALUES OF FIR COEFFICIENT GAIN AND OFFSET

The present section is tedious and can be skipped without any loss of comprehension. It is included for the engineer who wishes to pursue this design philosophy.

Equation (II-4a) expresses the maximum value of the filter gain, g, which occurs for a coefficient offset, d = -min. In general, to insure normalized coefficients bounded by 1,

$$g \le 1/(\max + d) \qquad \text{(II-10)}$$

Using this equation, (II-8c), and tightening the bound in (II-7) to account for the extra accumulator (guard) bit, we can derive the following bounds on C_0:

$$2^W/2^{22}N \le C_0 \le 1 - g(\max) \quad (\cong 0.526) \qquad \text{(II-11a)}$$

where W is the accumulator width, 22 is the desired coefficient resolution, and N is the number of coefficients.

For this particular implementation having a gain chosen to be 32, and our filter impulse response which, non-normalized, has a max of about 0.0148 and a min of about -0.00312, the only possible values of C_0 are (in floating point):

$$C_0 = dg = 2^i/2^{22}N \text{ ; for } i = W, W+1, W+2, W+3, W+4$$
$$= 0.03125, 0.0625, 0.125, 0.25, 0.5$$

Actually, C_0 can be the sum of any subset of these values and still cause the accumulator to overflow trivially, as long as (II-11a) is satisfied. But there is another constraint on our choice of C_0 which raises the lower bound on the subset sum. Whereas (II-10) insures normalized coefficients, the inequality (II-4b) insures that the coefficients are all positive. From (II-4b) we can derive,

$$(0.0998 \cong)$$
$$(-\min)g \le C_0 \qquad \text{(II-11b)}$$

We will setC_0 as close as possible to -min/(max-min) (~0.174) to maximize the conceivable range of g in (II-11);

$$C_0 = 0.0625 + 0.125 = 2^{-4} + 2^{-3} = 0.1875$$

In general, choosing g any higher than 2^5 would require another accumulator bit. If there were a system loss to compensate however, g can be chosen to exceed 2^5, and then C_0 can be set according to (II-11) without the need for an extra accumulator bit. g is not constrained to be an integer. As long as (II-11) is complied, the coefficient offset, d, will effectively adjust itself, however g is adjusted, to satisfy (II-8c).

In summary the Parks/McClellan floating point coefficients would be encoded in (unsigned) binary for storage in ROM using the following equation:

$$\text{binary code}(h_{norm}(n)) = 2^{22}\{\, gh(n) + C_0 \,\}$$

As a final note, to simplify the hardware, the term g/2 in equation (II-6) could be integrated into C_0 and the coefficients, $h_{norm}(k)$, which would eliminate the final subtracter circuit. This affects the preceding math which can be re-figured using d'=d-1/2N in place of d, and C_0'= C_0-g/2N=d'g in place of C_0. C_0 would still have the lower bound in (II-11a).

II-2 HARDWARE

II-2.0 BRUTE FORCE IMPLEMENTATION

The speed requirements of a brute force implementation would be excessive for low power CMOS. Since the input signal is one bit in width, we are performing only conditional additions; no multiplications. At a 48kHz output rate, the accumulation time would be 20.83uS / 2048coefficients~ 10nS. This is about an order of magnitude too fast. The decimation rate is 64. This means that every 64-3.072MHz clocks, we must output a new sample. If we use the 3MHz system clock as the accumulator clock, then we could only calculate one 64-tap FIR at the 48kHz rate.

To solve this problem, we will use 32 parallel processes all operating at the system rate of 3MHz [41][54]. Each parallel process, operating using a time skew of 64 clocks with respect to its neighbor, will only be required to output a sample every 2048 system clocks. Not more than one parallel process will yield an output at any given time. In this manner, a sample will be output every 64-3MHz clocks from a successive parallel process. Although the output rate will be 48kHz, each one of the

parallel processes will be operating at an subsampled output rate of only 1.500kHz.

Formally, the structure we have described is called a "multirate filter" [40] because the final output rate is not the same as any one parallel process' output rate. The structure is shown in Figure 27. We will omit the C_0 offset compensation for clarity. The parallel processes are themselves FIR filters working at a 1.5kHz rate. It is interesting that the output of each of these slow FIRs can be combined to form the desired output, $y_{des}(n)$, with no aliasing due to 1.5kHz replications. The time skew adds a phase factor to the transfer of each of the multirate FIRs so that their sum produces the desired result. It is also interesting to note that the impulse response of the decimator at the 48kHz rate is time-variant; there are 64 different impulse responses.

II-2.0.1 CONCEPT - CONVOLUTION

The classical process of FIR filtering is formally described as a convolution. The coefficients of the FIR filter are simply the sampled impulse response of that filter. Graphically, convolution means that the (symmetrical) FIR impulse response is time reversed, and then the input signal is shifted one system clock at a time underneath it. At every clock, the sum of the products of each sample and the value of the impulse response directly above is computed. Each sum constitutes an output sample. The process of decimation allows us to throw away 63 out of every 64 samples calculated. If we examine all of the multirate FIRs at any one instant in time, we find that they are each working with a coefficient set which is displaced 64 coefficients from either neighbor.

II-2.1 COMMUTATION

Each of the 32 FIRs in Figure 27 has its own ROM, complete with a redundant copy of the coefficients. Since the signal is only one bit in width, it gates the coefficient from the ROM into its associated accumulator. The accumulators are all attached to a commutator. The commutator 'spins' so that it selects another accumulator output at a 48kHz rate. When it reaches the last accumulator, it goes back to the first. The counter operates at the 3MHz rate and its 11 bit output is used as the address to all the ROMs. The accumulators all work at the 3MHz rate.

ROM #1 is organized having coefficient #1 at location 0. Starting from absolute time 0 with the first 1-bit signal sample, accumulator #1 (of FIR #1) will begin its computations starting with coefficient #1 which resides at location 0 in ROM. 2048 signal samples later, it will be read by the commutator and then zeroed only to begin its computations again starting with the 2nd set of 2048 signal samples.

Starting from absolute time 0, after 64 1-bit signal samples have arrived, we will expect accumulator #2 to begin its computations using the coefficient #1; but the address from the counter is pointing to location 64. Therefore, location 64 in ROM #2 must have coefficient #1 there. After 2048 more signal samples arrive, FIR #2's accumulator will be read and then zeroed, and the whole process will repeat.

In this fashion all 32 multirate FIRs operate in time skew. The coefficient ordering is shown in Figure27. The stored coefficients become rotated in steps of 64 from ROM to ROM.

II-2.1.1 CONCEPT - MULTIRATE PROCESS

This section is theoretical and can be skipped without loss of continuity.

The commutator in Figure 27 is the processing element that makes the decimator multirate. A simple time domain proof that the commutator can be used without unwanted aliasing is as follows: Assume that $y_{des}(n)$ is the desired 48kHz rate (decimated by 64) signal which is known to be good. Then

$$y_{des}(n) = \sum_{p=0}^{31} y_p(q) \quad \begin{array}{l} ; n \lesssim 48\text{kHz} \\ ; q \lesssim 1.5\text{ kHz} \end{array} \qquad \text{(II-12)}$$

where yp(q) is the 1.5kHz rate signal ($y_{des}(n)$ further subsampled by 32) from each parallel process;i.e.,

$$y_p(q) = y_{des}(32q - p) \qquad \text{(II-13)}$$

Then there will be no unwanted aliasing only if,

$$n = 32q - p \qquad \text{(II-14)}$$

For an alternate proof in the frequency domain, see [40,Sec.4.2.1]. Essentially, the proof there can be boiled down to first time-shifting the desired sequence,

$$z^{-p} Y_{des}(z) \qquad \text{(II-15)}$$

and then subsampling (decimating further) by 32,

$$Y_p(z) = (1/32) \sum_{k=0}^{31} e^{-j(w-2pik/32)p}\, Y_{des}(e^{j(w-2pik/32)}) \qquad \text{(II-16)}$$

Then prove that:

$$Y_{des}(z) = \sum_{p=0}^{31} Y_p(z) \qquad \text{(II-17)}$$

II-2.2 MULTIPLEXING

Figure 28 is more efficient in terms of ROM since there is no longer a duplication of it. The ROM is now segmented having 32 outputs from 32 segments. Each segment holds 64 unique coefficients. The 6 LSBs from the 3MHz counter address all the ROM segments at once. Each segment runs through each of its 64 coefficients at the 3MHz rate and sends them to its respective output. The coefficient segment assignment is shown. The multiplexers now route any one of their 32 22-bit inputs to their respective output. The routing is controlled by the 5MSBs out of the counter at the 48kHz rate. The operation below the multiplexers is the same as before. The purpose of the multiplexers is to perform the coefficient rotation. Looking at multiplexer #1, we see that the segments are ordered into it

(left to right) starting with segment #1. Multiplexer #2 orders the segments such that segment #32 is in the first (leftmost) input while segment #1 is the second input. This agrees with the reasoning we used to order the coefficients in Figure 27.

II-2.3 VLSI

The actual physical organization is shown in Figure 29. There, all the multiplexers have been replaced by one barrel shifter. The multiplexer input wiring in Figure 28 was quite hairy. The barrel shifter in Figure 29 takes any one of its 32 inputs and routes it to any one of its 32 outputs. The wiring external to the barrel shifter is greatly simplified. Internally, the barrel shifter embodies a type of matrix organization. The ROM is still segmented as before, controlled by the 6 LSBs out of the counter at the 3MHz rate. The accumulators are now aligned vertically one above the other to form a more compact silicon structure.

The actual VLSI realization uses a bit-wise organization such that bit 0, for example, for all the coefficients in the ROM, all the barrel shifter inputs, and for all the accumulators, are aligned in one column. This allows easy decomposition for coefficients of varying widths.

The advantages of this design include an easy stereo implementation; all that must be done is to double the number of accumulators. The ROM and barrel shifter are shared for any number of channels. The present implementation has 64 accumulators for stereo operation. Every 4 accumulators comprise one multiplexed accumulator. We estimate the dimensions at about 140 X 180 mils.

ACKNOWLEDGEMENT

We would like to thank Steve Kozachyn of ENSONIQ for the excellent job done on the figures for both this and the original 1988 November Journal publication.

REFERENCES

[1] J. Andrew Moorer, "The Manifold Joys of Conformal Mapping: Applications to Digital Filtering in the Studio", Journal of the Audio Engineering Society, vol.31, no.11, pg.826, November 1983.

[4] Alan V. Oppenheim and Ronald W. Schafer, Digital Signal Processing, Prentice Hall, publisher, Englewood Cliffs, New Jersey, USA, 1975

[5] Leland B. Jackson, Digital Filters and Signal Processing, First Edition, Kluwer Academic Publishers, Boston, Massachusetts, USA, 1986

[22] George R. Cooper, Clare D. McGillem, Probabilistic Methods of Signal and System Analysis, First Edition, Holt Rinehart Winston, publisher, New York City, USA, 1971

[31] Tien-Lin Chang, "Suppression of Limit Cycles in Digital Filters Designed with One Magnitude-Truncation Quantizer", IEEE Transactions on Circuits and Systems, vol.CAS-28, no.2, pg.107, February 1981

[38] John Vanderkooy and Stanley P. Lipshitz, "Resolution Below the Least Significant Bit in Digital Systems with Dither", Journal of the Audio Engineering Society, vol.32, no.3, pg.106, March 1984; Correction, ibid. (Letters), vol.32, p.889, November 1984

[2]-[37] found in [39]

[39] Jon Dattorro, "The Implementation of Recursive Digital Filters for High-Fidelity Audio", Journal of the Audio Engineering Society, vol.36, no.11, p.851, November 1988

[40] R. E. Crochiere and L. R. Rabiner, Multirate Digital Signal Processing,(Prentice-Hall, Englewood Cliffs, NJ, 1983)

[41] Max W. Hauser, Robert W. Brodersen, "Monolithic Decimation Filtering for Custom Delta-Sigma A/D Converters", IEEE International Conference on ASSP, 1988, Volume III (D), pg.2005

[42] Ning He, Andres Buzo, and Federico Kuhlmann, "Multi-Loop Sigma-Delta Quantization: Spectral Analysis", IEEE International Conference on ASSP, 1988, Volume III (D), pg.1870

[43] Sasan H. Ardalan, "Analysis of Delta-Sigma Modulators with Bandlimited Gaussian Inputs", IEEE International Conference on ASSP, 1988, Volume III (D), pg.1866

[44] D.R. Welland, B.P. Del Signore, E.J. Swanson, CRYSTAL Semiconductor Corp., Austin, Texas, et Al., "A Stereo 16-Bit Delta-Sigma A/D Converter for Digital Audio", The 85th Convention of the AES, Nov. 1988, Reprint #2724 (H-12), also Journal Audio Eng. Soc., Vol.37, No.6, 1989 June, pg.476

[45] Timothy F. Darling, Malcolm J. Hawksford, "Oversampled Analogue-To-Digital Conversion for Digital Audio Systems", The 85th Convention of the AES, Nov. 1988, Reprint #2740 (H-11)

[46] E. Stikvoort, "Higher Order One Bit Coder for Audio Applications", The 84th Convention of the AES, March 1988, Reprint #2583 (D-3)

[47] Robert W. Adams, "Design and Implementation of an Audio 18-Bit Analog-to-Digital Converter Using Oversampling Techniques", Journal AES, vol.34, no.3, March 1986, pg.153

[48] Bernhard E. Boser, Bruce A. Wooley, "The Design of Sigma-Delta Modulation Analog-to-Digital Converters", IEEE Journal of Solid-State Circuits, vol.23, no.6, December 1988, pg.1298

[49] Kuniharu Uchimura, Toshio Hayashi, Tadakatsu Kimura, Atsushi Iwata, "Oversampling A-to-D and D-to-A Converters with Multistage Noise Shaping Modulators", IEEE Transactions on ASSP, vol.36, no.12, December 1988, pg.1899

[50] MOTOROLA, "Principles of Sigma-Delta Modulation for Analog-to-Digital Converters", 1989, Tactical Marketing Manager, Motorola DSP Operations, 6501 William Cannon Dr. West, Mail Drop OE314, Austin, TX, 78735

[51] DBX/CTI Research, "Application Notes, F410 Front-End/D20C10 Decimator, High-Resolution A/D Converter IC Set", Carillon Technology Inc. Research Group, 71 Chapel St., Box 100C, Newton, MA 02195

[52] P.J.A. Naus, E.C. Dijkmans, "Low Signal-Level Distortion in Sigma-Delta Modulators", The 84th Convention of the AES, March 1988, Reprint #2584 (D-4)

[53] M. Richards, "Improvements in Oversampling Ana-

logue to Digital Converters", The 84th Convention of the AES, March 1988, Reprint #2588 (D-8)

[54] Max W. Hauser, Paul J. Hurst, Robert W. Brodersen, "MOS ADC-Filter Combination That Does Not Require Precision Analog Components", Digest of Technical Papers, 1985 IEEE International Solid-State Circuits Conference, WAM 7.5, pg.80

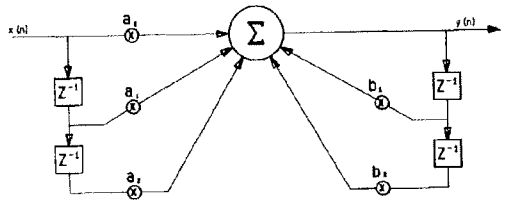


Fig. 1. For audio use. Direct Form I.

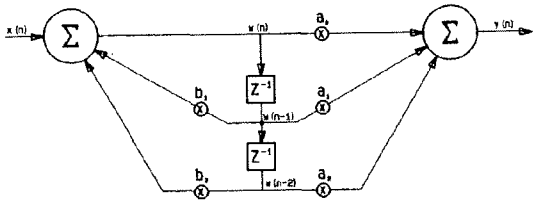


Fig. 2. Not for audio use. Direct Form II-Canonic.

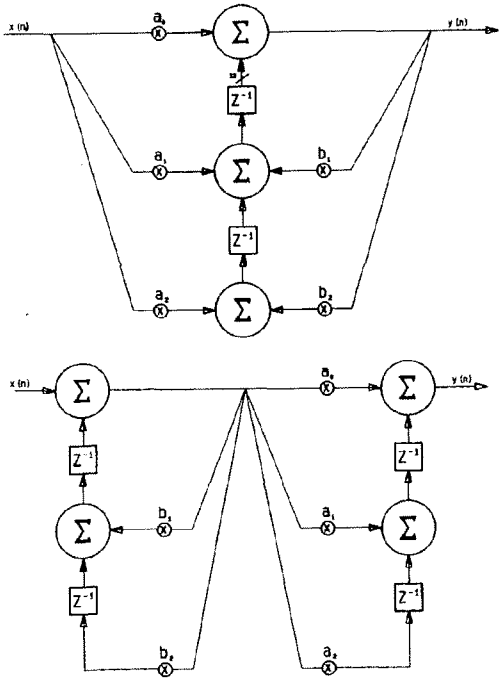


Fig. 3. a) OK for audio use. Direct Form II Transpose. b) Not for audio use. Direct Form I Transpose.

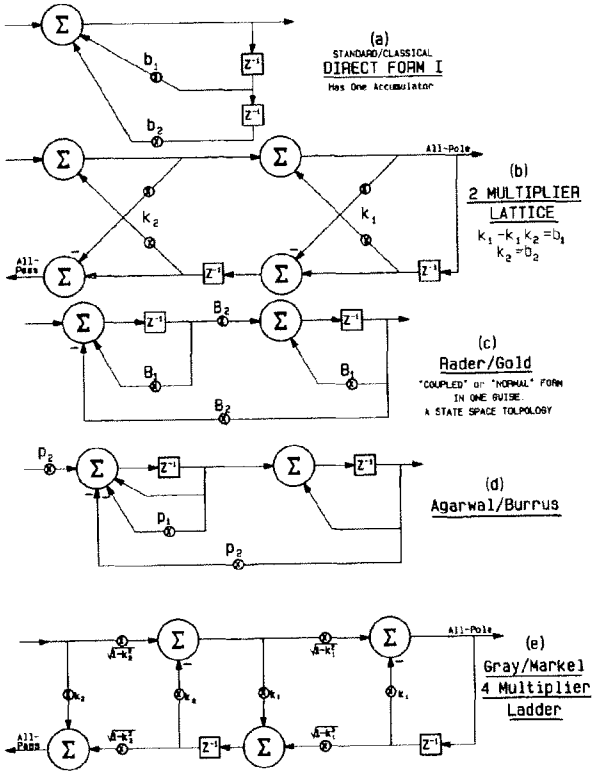


Fig. 4. Some second order topologies.

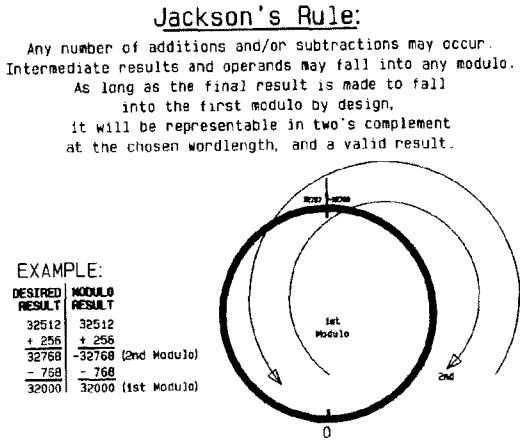


Fig. 5. Two's complement is a modulo arithmetic. The first modulo (ring) for 16 bits is shown. The arrows help visualize the traversal from the first into the second modulo, and then back again.

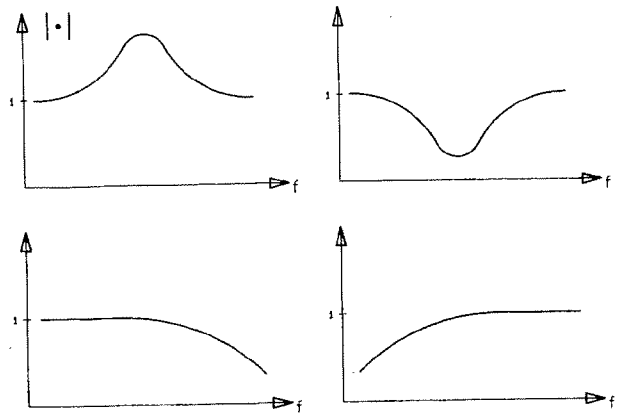


Fig. 6. Unity gain design.

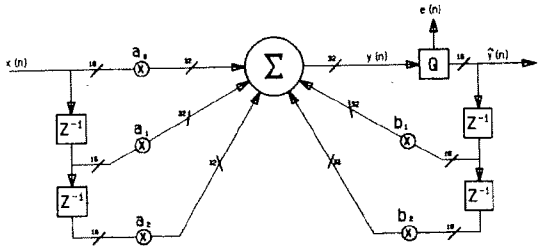


Fig. 8. Direct Form I having truncator.

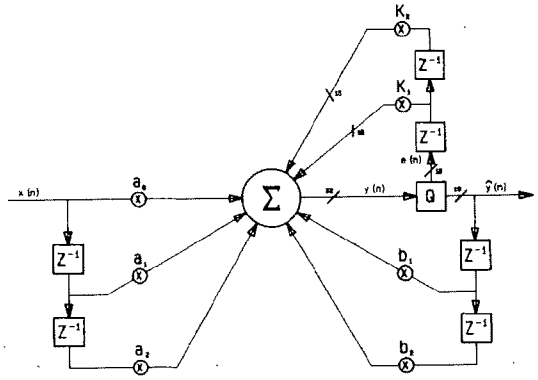


Fig. 11. Second order error feedback.

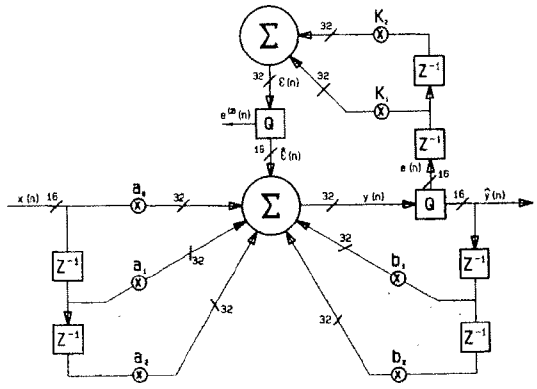


Fig. 11-II. Second order truncation error cancellation showing all truncation errors.

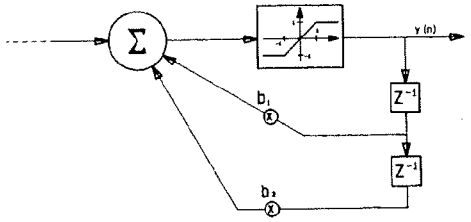


Fig. 15. Showing the saturator explicitly, forced overflow analysis.

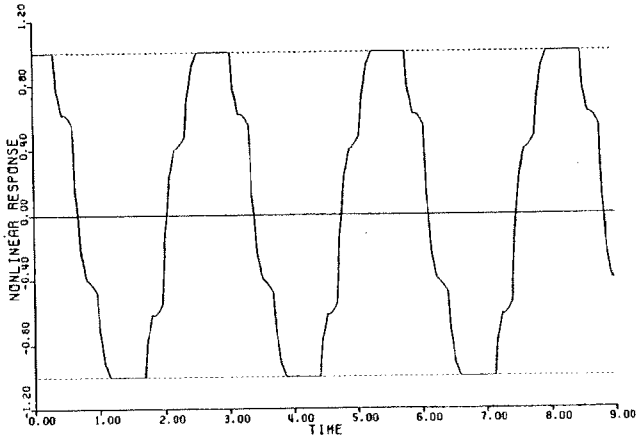


Fig. 16. Typical forced overflow response for a sinusoidal input.

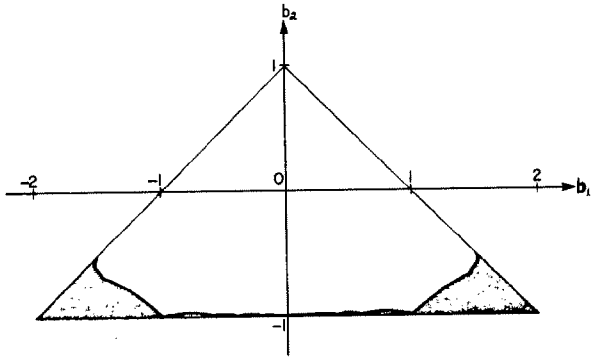


Fig. 17. Stability triangle for positive topology.

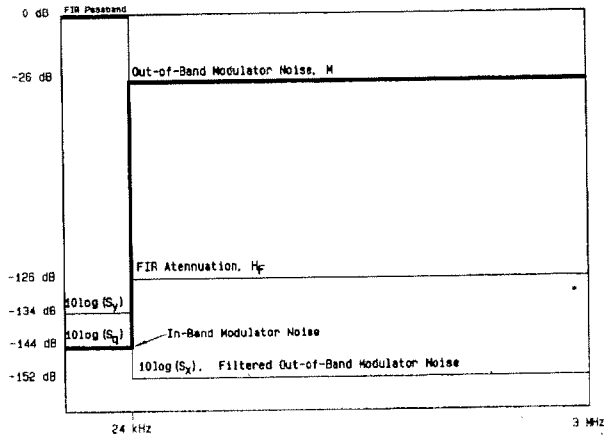


Fig. 19. Relationship of M, S_Q, and H_F to achieve S_y, the in-band spectral density after decimation.

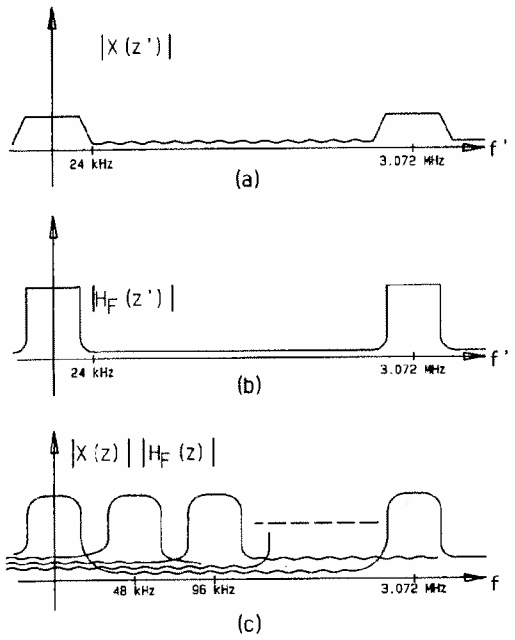


Fig. 18. Decimation in the frequency domain.

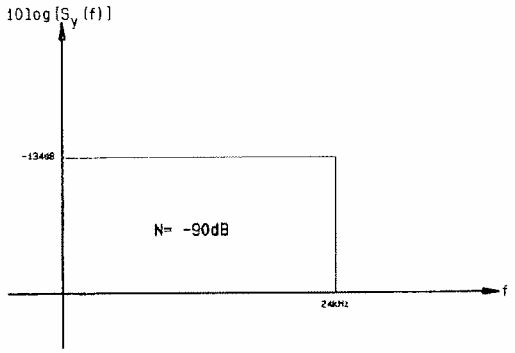


Fig. 20. Required power spectral density of noise power, N, to achieve 90 dB S/N.

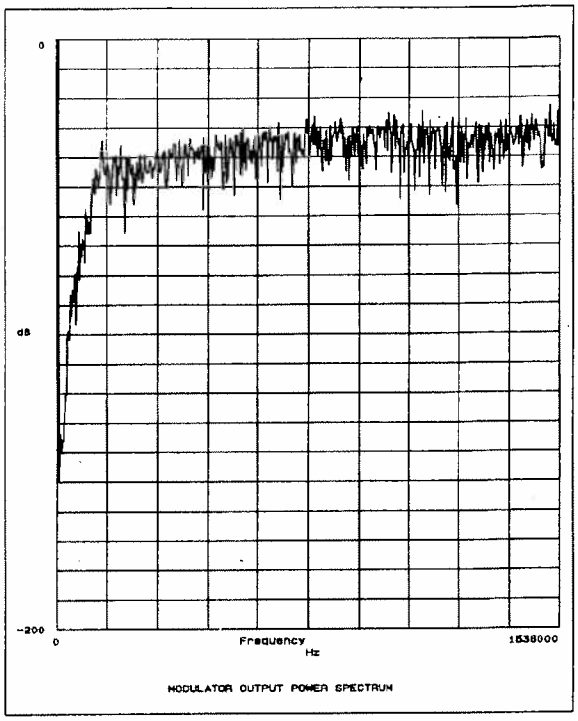


Fig. 21.

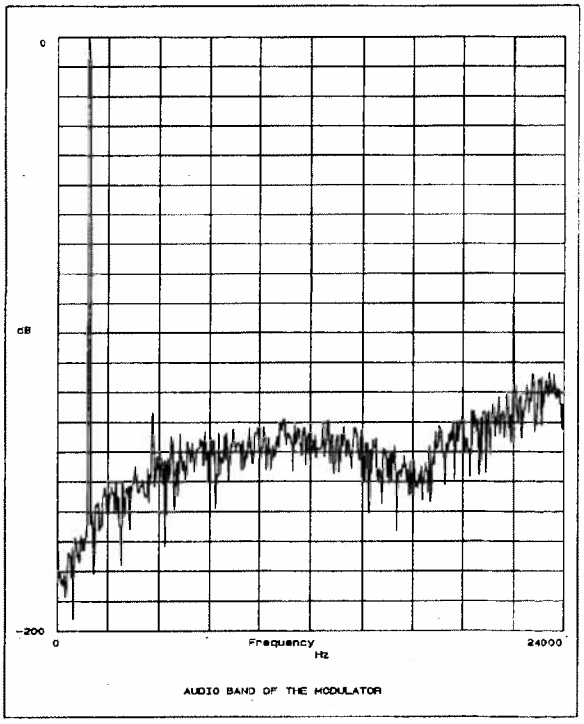


Fig. 22.

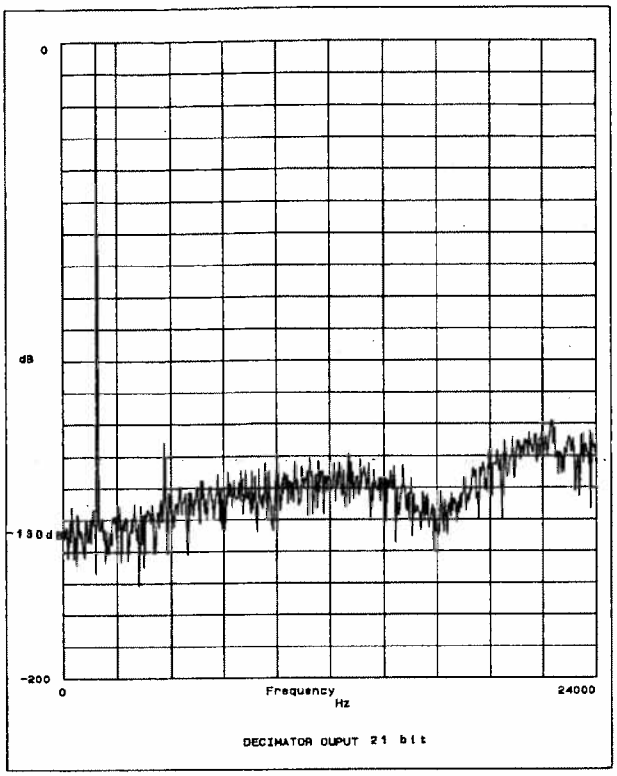


Fig. 23A.

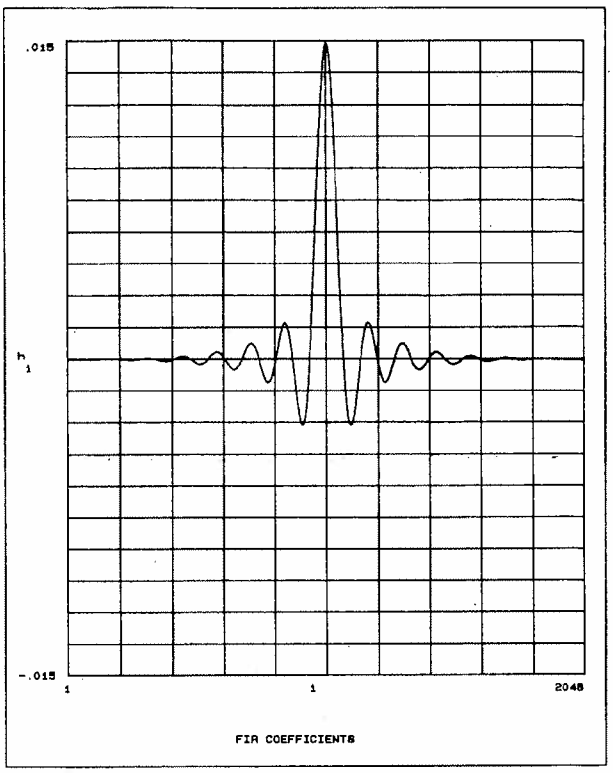


Fig. 24.

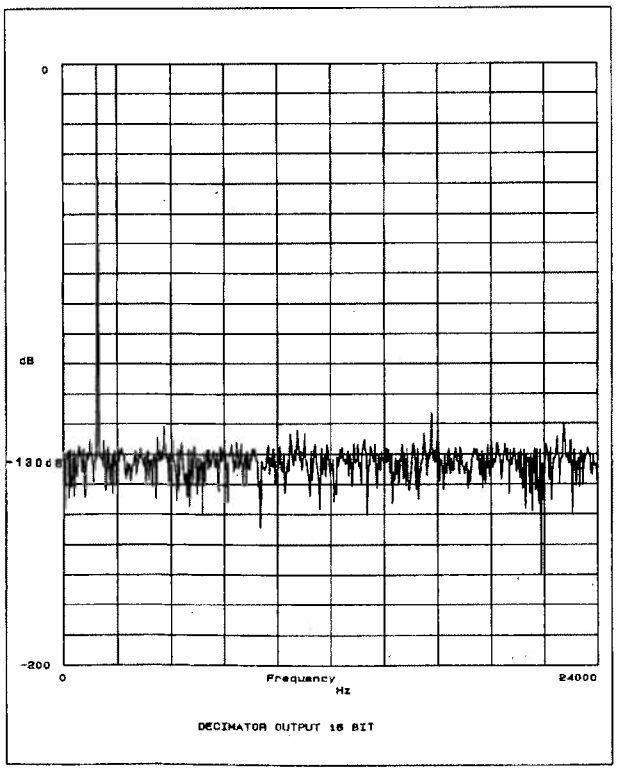


Fig. 23B.

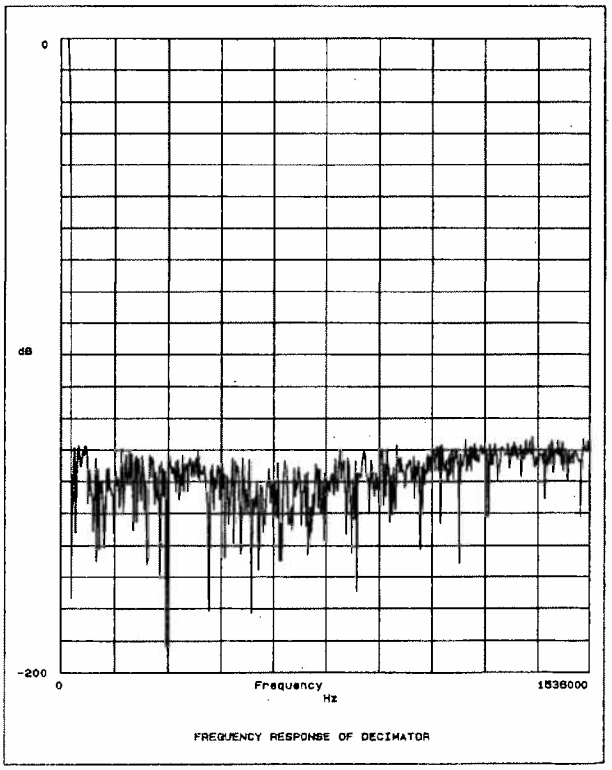


Fig. 25.

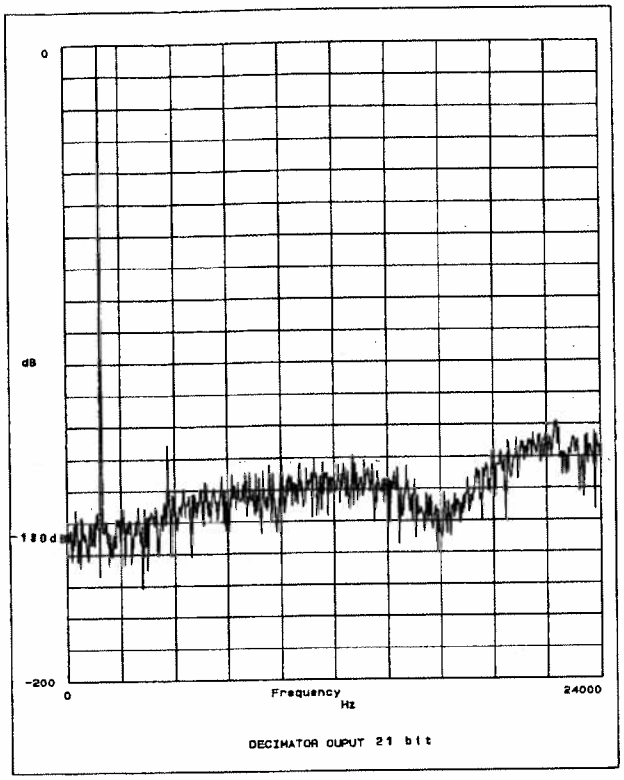


Fig. 23A.

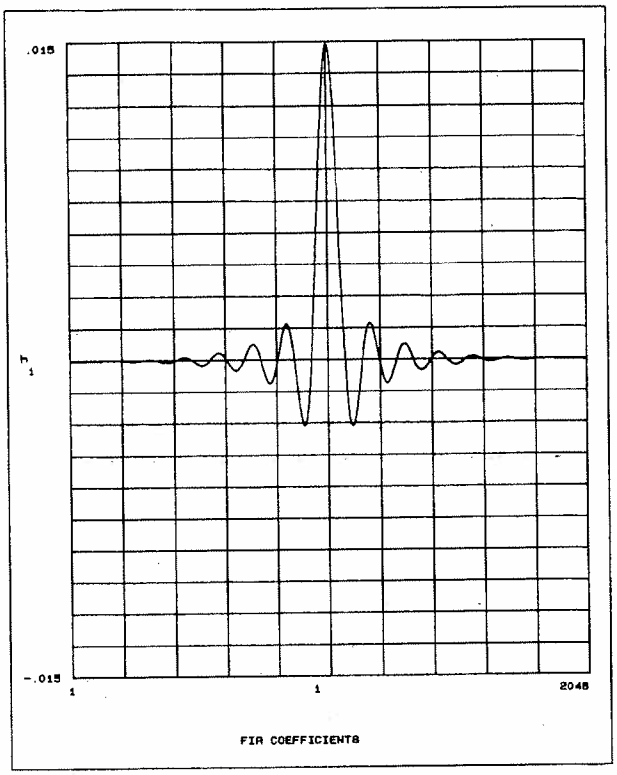


Fig. 24.

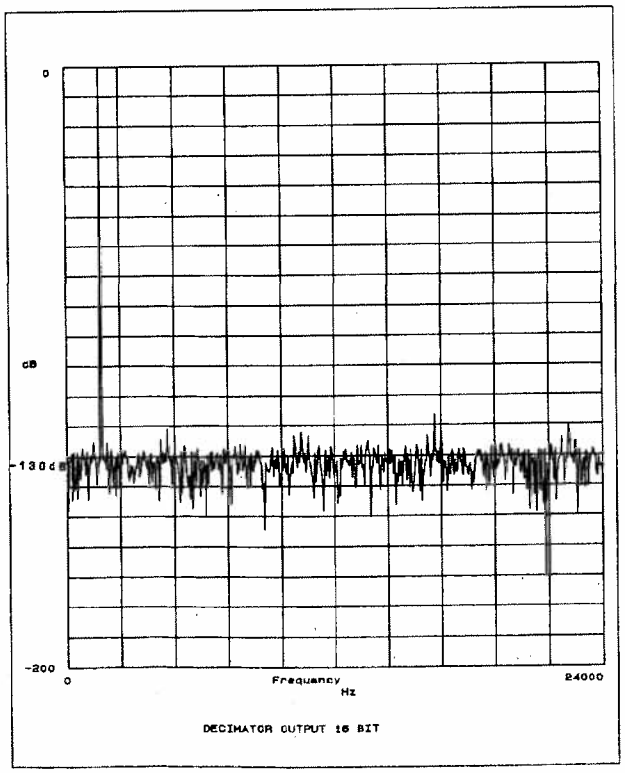


Fig. 23B.

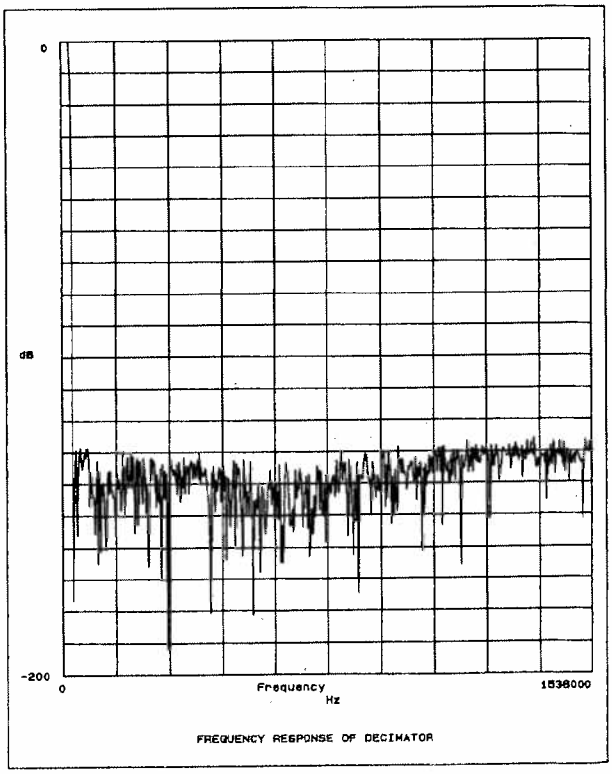


Fig. 25.

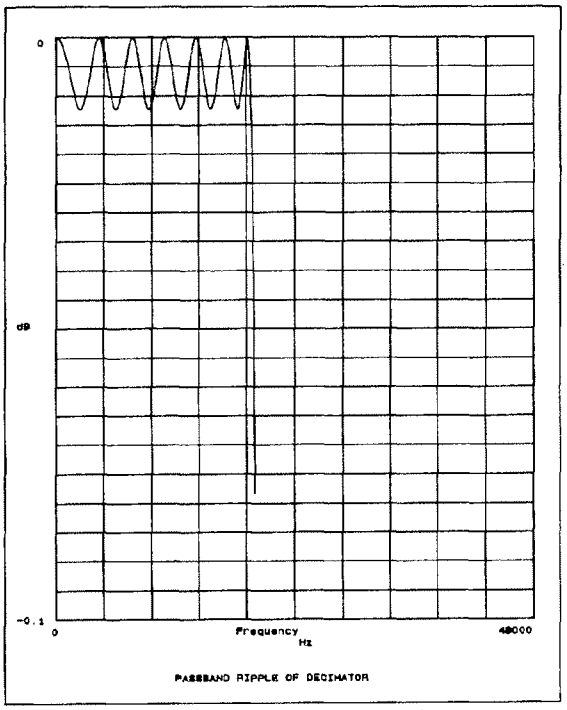


Fig. 26.

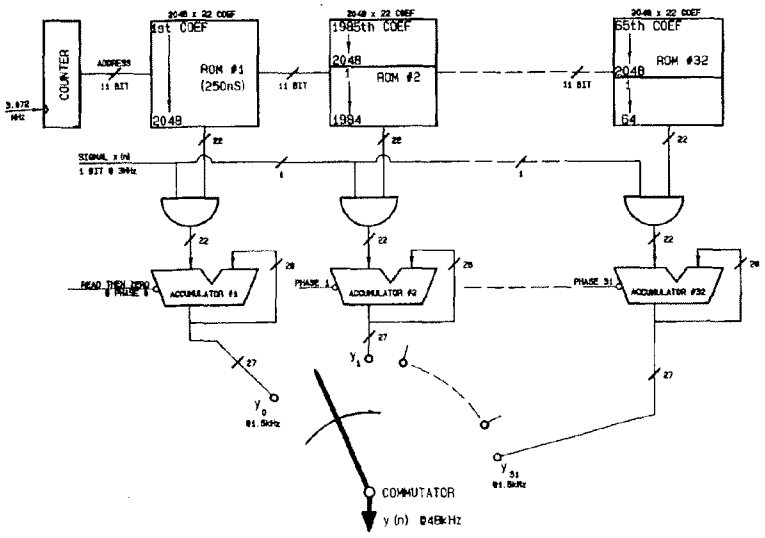


Fig. 27.Multirate FIR decimator.

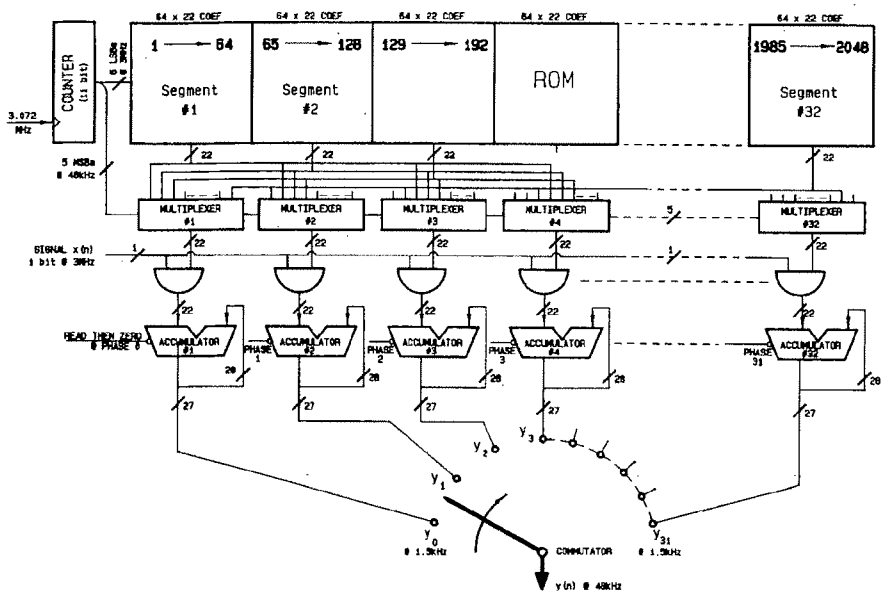


Fig. 28.ROM collapse.

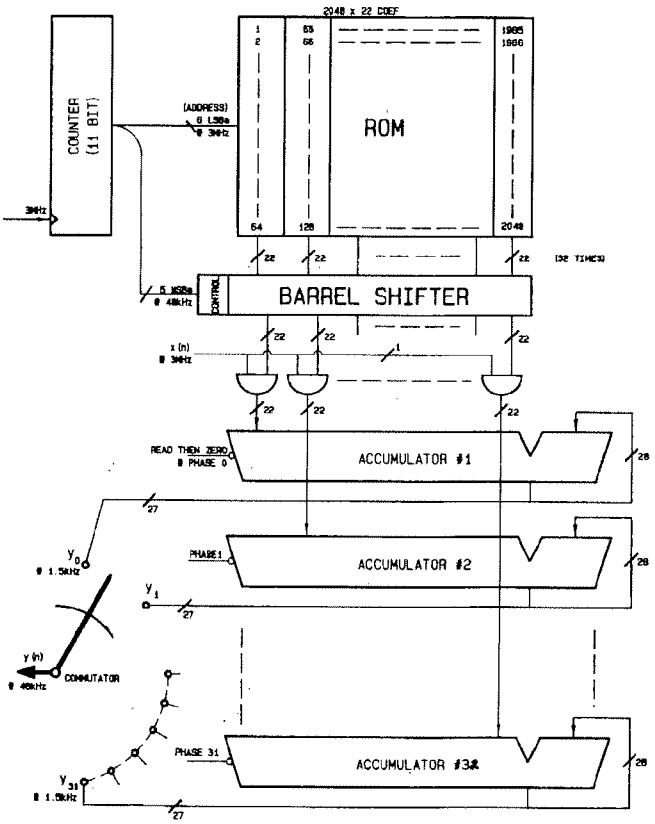


Fig. 29.Physical organization — MUX collapse.

The Implementation of Recursive Digital Filters for High-Fidelity Audio*

JON DATTORRO

ENSONIQ Corporation, Malvern, PA 19355, USA

The problems are described which the practicing engineer encounters who unwittingly approaches the realization of IIR digital filters for the first time. It is assumed that suitable design programs are available to calculate the coefficients, and it is desired only to *implement* the filter. Elegant solutions are provided for some of the most intimidating problems typically encountered, which are 1) input scaling requirements, 2) truncation noise propagation and recirculation, and 3) accurate low critical frequency filtering. It is shown that the direct form I noncanonic topology is the best for use in digital filtering, and while 16/32-bit DSP chips such as the TMS32010 or the ADSP-2100 can be used in many applications of high-fidelity digital audio, they will *not* meet the most demanding requirements.

0 INTRODUCTION

0.1 Ground Rules

Let us assume at the outset that the theoretical design of digital filters in the z domain is a routine task. There are many books and papers written on this topic. Moorer [1] provides unity-gain, minimum-phase designs and is a superb reference for digital audio work. The low-pass bound on the transition region of 6 dB per octave per conjugate pole pair does not exist in Moorer's second-order parametric filter designs. We have tested and verified the C programs that he provides. As such, this paper will not deal with the derivation of floating-point filter coefficients. Rather, it is assumed that they are known and the only remaining task is to design the software (or hardware) that will implement the filters. It should be emphasized that the design of digital filters and the *implementation* of digital filters can be carried out as two distinct tasks. Most design procedures produce the coefficients for the direct form filter topologies. The coefficients for other topologies can usually be derived from these via suitable transformations. More specifically, a typical design program would produce one set of coefficients which are directly applicable to either of the two standard second-order sections (the direct forms, Figs. 1 and 2, or their transposes, Fig. 3) which are described in the frequency domain by the transfer

$$H(z) = \frac{a_{F0} + a_{F1}z^{-1} + a_{F2}z^{-2}}{1 - b_{F1}z^{-1} - b_{F2}z^{-2}} . \quad (1)$$

For brevity in the text, we will often write equations like Eq. (1) as

$$H(z) = \frac{\Sigma a_{Fi}z^{-i}}{1 - \Sigma b_{Fi}z^{-i}} \quad (2)$$

where the subscripts F indicate that these coefficients are the floating-point values provided by some design procedure. Our implementations will generally operate with fixed-point coefficient values, and so this becomes one of the issues that bear discussion.

We will always prefer minimum-phase designs (all zeros within the unit circle) because 1) the total phase excursion is always less than 360° for a second-order section; long phase ("propagation" or "transport") delays are unacceptable; 2) all coefficients are bounded as in Pascal's triangle if the filter design is unity gain. We prefer unity-gain designs because it will be less apparent when they are kicked in.

We will purposely show our filter circuit topologies as having positive-signed coefficients at the multipliers (see Fig. 1) because this leads to positive accumulations in the resulting software. We call this a *positive* to-

* Manuscript received 1988 July 18; revised September 7, 1988.

pology. This topology is sometimes more convenient for the programmer; hence the negative denominator coefficients in Eq. (1). The implementer must be aware of the convention concerning the signs of the coefficients produced by some design procedure. If the filter blows up at the start, this is the first place to look.

This work was conceived within the dimensions of the digital signal processing (DSP) architecture of the entire TMS320 series, but the concepts and problem solutions are universal and can be readily applied on other DSP processors. The primary feature of the TMS320 architecture (besides being nearly a full featured microprocessor) is a fast-signed 16- by 16-bit multiplier having a 32-bit product which can be accumulated to 32 bits. We call this a 16/32-bit architecture. So our work will be applicable to any architecture having N-bit by N-bit multiplications with nominally $2N$-bit products and $2N$-bit (the same wordlength) accumulations. This is typical of the current wave of DSP chips. The TMS32010 and 20 are peculiar in that there is no provision for an unsigned multiply. This at first seems to impair double-precision coefficient performance, but a very simple method to get around this limitation will be discussed.

TMS32010 code is provided in Appendix 1, which comes from a commercial parametric filtering application. It is provided mainly as proof of principle to the concepts presented herein.

0.2 Audio

In order for a digital filter to be usable for high-quality audio, it must not degrade system performance such that the system specifications are violated. This means that the effects of the filter must be transparent to the user except for the concomitant changes in frequency and phase response. Therefore such things as the signal-to-noise ratio (SNR), the dynamic range, or the total harmonic distortion plus noise (THD + N) of the entire system should not be degraded. The digital filter itself, then, must have specifications that surpass those of the host system.

We would like there to be some advantage to the use of a digital filter in comparison to the corresponding analog filter, so we mention one advantage which is not often cited: in analog filter implementations, transient intermodulation distortion (TIMD) may be a problem. In corresponding digital filter implementations there is no physical analogy to the cause of TIMD (slew-rate limiting) [2], thus there can be no TIMD induced.

In audio work it is customary to find designs centered in the very low-frequency region. To a recording engineer there is a vast difference between a filter whose critical frequency is placed at 50 Hz and another placed at 60 Hz. Single-precision (16-bit) coefficient realizations are not satisfactory for the required control over low critical frequency. It is generally accepted now that 24-bit coefficients are adequate for high-quality digital audio work.

0.3 Truncation Noise

The first widely used digital filters conceived in hardware in the 1960s were intended for telecommunications applications such as the telephone touch-tone decoder. Leland B. Jackson, who is often called the "father of digital filters" for his pioneering work in the analyses of associated nonlinear phenomena, was employed by Bell Telephone Laboratories when he and James F. Kaiser presented a hardware scheme [3] for the implementation of such a decoder. The outstanding topological idiosyncrasy of their circuit implementations, from our present vantage point, was that all the signal paths, including the multiplier paths, were constrained to have the same wordlength. Therefore their first analyses of roundoff noise warranted by the truncation process were based on a constraint that need no longer exist. DSP chips suitable for audio routinely supply double-wordlength products and accumulations. We must still be concerned, however, with the truncation at the output *and* its effect upon the digital circuit when fed back.

In concept, if a digital filter accepts as input an integer, then an integer plus a fraction (a fixed-point number having a nonzero fractional part) is, in general, produced at the output. If the output were rounded off to the nearest integer and we consider only the effects of errors that occur *at* the output, then the output would be in error by at most ½ LSB (3 dB) in magnitude. This rounding (or truncation) process at the filter output is unlike the process that describes the quantization errors

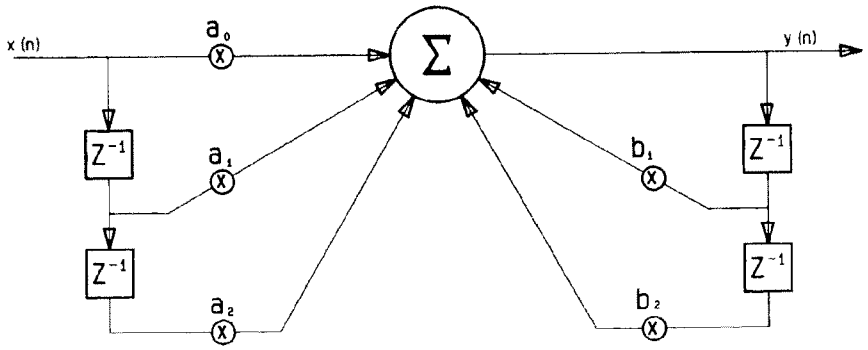


Fig. 1. Direct form I; for audio use.

inherent in analog-to-digital (A/D) conversion. Still, it is most often modeled in the literature [4, p. 413] as a random process that produces an additive white noise source at the output node. In any case, the input signals to cascaded second-order digital filters have, in addition to the signal itself, a noise source from the preceding stage which gets filtered along with the signal. We will be concerned with just how these noise sources combine through a cascade.

0.4 Filter Circuit Topology

0.4.1 Second Order

Since all of the required hardware to implement digital filters in real time has now been integrated onto a single programmable chip, there is a tendency to think of a digital filter as an amorphous software implementation as opposed to a hard circuital implementation. (Indeed, the first digital filters were realized as models of continuous systems on general-purpose computers in high-level languages for the purpose of analyzing statistical data in nonreal time.) From the viewpoint of a programmer, a digital filter appears at first to be the solution to a difference *equation*, and circuit topology seems not too important. This cannot be further from the truth because the choice of circuit topology is as intrinsic to a given software realization as it is to the corresponding discrete hardware realization. One needs only to convince oneself that soft instructions solving a difference equation are, in fact, shuffling data about predetermined paths inside some architecture made up of arithmetic elements, that is, hardware. The circuit topology chosen determines the number and the order of the computations, and vice versa. Why choose a topology which might require more computation or more storage? The answer has to do with numerical inaccuracy. Certain topologies react better than others to numerical errors. So for these reasons we often choose *not* to use the most straight forward solution to the filter difference equation in our software. This is where the study of circuit topology comes in.

There are other considerations besides numerical inaccuracy which impact our choice of topology. One of the great disappointments in the past with conventional 16/32-bit digital filter implementation was the finding that filter inputs needed to be scaled (attenuated). Scaling was required in order that overflow be avoided at internal nodes in a given filter topology, even though the output node itself may never have been threatened with overflow. The unfortunate side effect of scaling is that the SNR of the input audio is irrevocably worsened by an amount proportional to the scale factor. The same problem arises in analog designs. But good analog filter implementation can accommodate this reduction of input level because it can start off with as much as 120 dB SNR. For example, an analog signal with 90-dB SNR scaled by, say, 20 dB is ideally the same as an unscaled analog signal having 110-dB SNR. The analog filter can easily accommodate this SNR, and so analog scaling is generally not a problem. But the situation is different in digital. Using a 16-bit signal converter, we start out with a noise floor due to quantization in the vicinity of −90 dB, theoretically. With our 16/32-bit digital filter itself only having a 90-dB SNR, we really cannot ask it to represent a scaled signal now having a 110-dB SNR. We would end up with an output signal having an SNR of at most 70 dB.

A 24/48-bit architecture would ameliorate the scaling problem for 16-bit signals, in analogy to the 120-dB analog system mentioned. This is because using the feedback techniques presented herein, a 24/48-bit architecture digital filter can be made to operate with a 138-dB SNR. Input scaling a 16-bit signal by less than 48 dB would amount to a justification problem within the 24-bit wordlength, that is, no data loss. Even so, we are going to show how to sidestep the scaling problem altogether, regardless of the wordlength of the processor we are using, by clever choice of topology. For these reasons we venture to suggest that *any* 16/32-bit digital implementation employing input scaling is inadequate for high-fidelity digital audio.

0.4.2 High Order

Constructing higher order filters from second-order sections is done by cascading however many second-order filters are produced by a design procedure. A parallel realization would require what is called a partial fraction expansion, which transforms the given product of biquadratic transfers into sums of biquadratics. These procedures involve pole–zero pairing selection and the

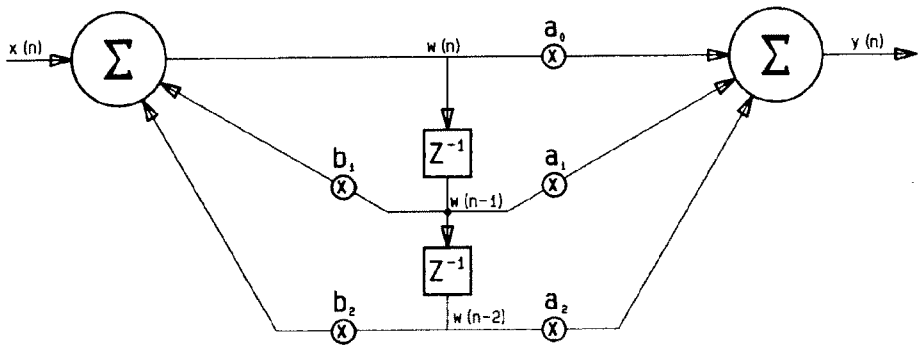


Fig. 2. Direct form II, canonic; not for audio use.

scaling of the input signal into individual sections for optimal SNR performance and the prevention of output overflow from individual sections. Since it is not possible, in general, to design high-order filters such that each second-order section is unity gain, input scaling is mandatory regardless of the section topology. If the pole–zero pairing is chosen for as many close-to-unity-gain sections as possible, and if the section topology is chosen to be direct form I for the reasons to be discussed shortly, then the scaling requirement can be minimized and limited to consideration of only the output node of each stage. (Jackson [5] gives closed-form scaling and pairing procedures.)

The construction of higher order filters from cascades or parallels of second-order sections ("biquads") is the best approach because one soon discovers that the direct form implementation of high-order filters leads to geometric increases in the range of coefficient values, pole–zero sensitivity, and truncation noise recirculation. Based on the material presented in this paper, we will always want to choose the parallel and cascade forms I, as given by Jackson [5]. He notes that parallel realizations, while sometimes being less noisy than cascades, have very high zero sensitivity to coefficient quantization, and so cascades are often preferable. (Note that some of Jackson's analyses still account for truncation before accumulation.) Graphic equalizers are traditionally realized as parallel second-order stages, while parametric equalizers are cascade implementations having individual second-order stages forming a complete filter.

The only direct high-order topology which is known to elude both the scaling and the truncation noise recirculation problem at once is the Gray–Markel all-pole four-multiplier ladder [6] (Fig. 4e). For simplicity's sake, in this paper we will constrain all our filters to be biquadratic, and we want their magnitude responses to be deviations from unity gain. Therefore we will be able to show how to skirt the scaling issue totally. If we speak of a cascade of second-order sections, it will usually *not* be in reference to the construction of a high-order filter. Each second-order stage will itself be a complete filter.

Aside from this cursory discussion of higher order filters, all of the work in this paper will be limited to complete second-order digital filters, where all coefficient values are bounded in magnitude by 2 [Eq. (30)]. Later we will show examples of second-order filters having high gain and high Q factor. This comes about because of the proximity of the pole–zero pair in those designs.

We will now reintroduce a classic topology which inherently eludes the input scaling requirement, hence picking up some free SNR.

1 CIRCUIT TOPOLOGY: SCALING AND OVERFLOW

We will abruptly end our search for the best digital audio filter circuit topology with the most simple topology, the direct form I noncanonic implementation of the second-order digital filter. This circuit is shown in Fig. 1. The second-order difference equation which describes the output $y(n)$ is given in Eq. (3) and is straightforward to derive from Fig. 1,

$$y(n) = \Sigma a_i x(n - i) + \Sigma b_i y(n - i) \ . \tag{3}$$

Had we implemented this second-order difference equation directly in software, Fig. 1 would represent the topological choice we had unwittingly made. The direct form II filter circuit topology shown in Fig. 2 is chosen, historically, in preference to direct form I in Fig. 1 because it has superior truncation noise performance when both are implemented with truncation *prior* to accumulation. Otherwise, the direct form I is better. The direct form II also has two less storage (delay) elements, which yields the minimum number of storage elements for the second-order topology. Hence the term *canonic topology* (for "minimal" topology) is often used synonymously. The equations that describe the second-order filter in Fig. 2 are

$$\begin{aligned} w(n) &= x(n) + \Sigma b_i w(n - i) \\ y(n) &= \Sigma a_i w(n - i) \ . \end{aligned} \tag{4}$$

These simultaneous equations can be simplified to Eq. (3). Implementing these equations in software represents the choice of the canonic topology. The two other standard forms, which are the circuital transposes of the direct forms I and II, are shown in Fig. 3. There are many other topologies that embody the second-order section. The numerous topologies possess different characteristics with regard to the recirculation of truncation noise. These topologies can each be quantitatively characterized by SNRs of input signal to truncation noise power, and each topology is said to have a particular intrinsic "noise gain." Agarwal and Burrus [7] published an excellent article on this topic, and the topic is collectively referred to as *roundoff noise* [8]. In this paper we substitute the term *truncation* since it is this and not rounding which will be performed in our filter programs (for good reason).

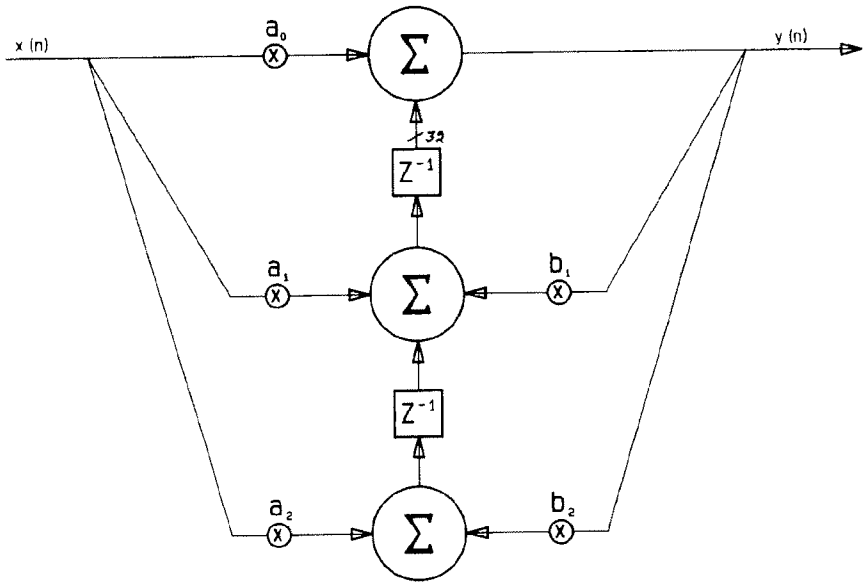
We choose direct form I because the truncation at the accumulator output is restricted to only the feedback paths; thus it will be easier to compensate. A casual scrutiny of this familiar topology reveals that there is only one accumulator, hence for truncation postaccumulation, one noise source. We choose the direct form I also because of its unique overflow properties. Notice in Fig. 1 that all the internal paths of this second-order section converge at this one accumulator. Further, the output of this mecca is attached directly to the output of the filter itself. Therefore by designing the filter for unity gain from input to output, we can ensure that the filter output $y(n)$ will not overflow under most real operating conditions. During the course of the calculations, that is, at intermediate steps in the accumulation of the final output, the accumulator is likely to overflow

many times. Jackson showed back in 1968 [3] that because 2's complement arithmetic is a modulo math, as long as the final result falls into the numerical range of the accumulator (the first modulo), intermediate accumulations may be allowed to overflow any number of times (traverse other moduli) without causing an erroneous accumulation. This means that the effective wordlength of the accumulator in the direct form I is much greater than its actual wordlength.

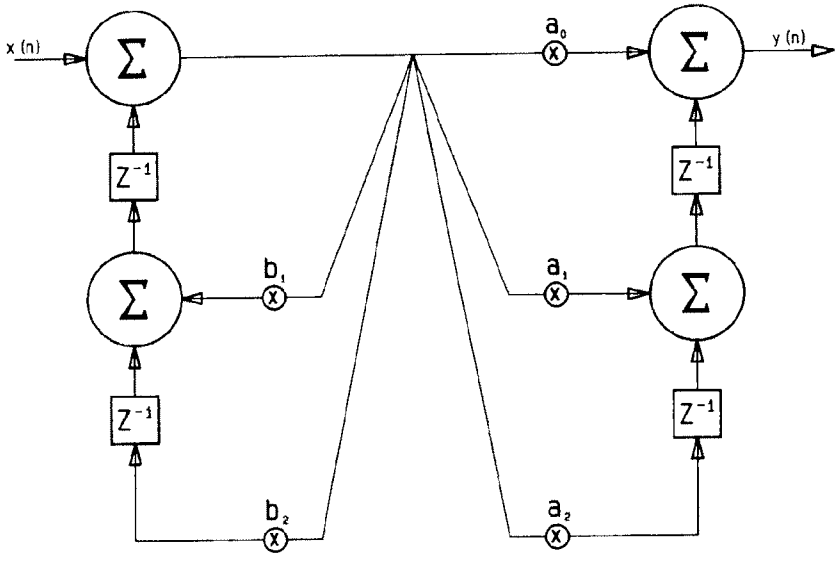
We have, therefore, infinite headroom at the internal nodes using the direct form I topology because all the internal nodes feed the one accumulator. The internal nodes which feed multipliers are not capable of overflow (by design), so there is no need to scale the input signal; hence no potential loss in the signal's SNR. Another way to look at this is that we have an infinite headroom accumulator.

To take full advantage of modulo arithmetic we will never want the accumulator to autonomously operate in the saturation mode. If overflow is detected *at the final output*, then saturation must be performed there, but only under program control.

These advantages cannot be capitalized on using the canonic topology and, for that matter, in most other topologies because they have at least two accumulators. The overflowed output of one of them will invariably feed back from an internal node into a multiplier input, which can only operate on fixed wordlengths (Fig. 4). In order to prevent this from happening, we would somehow need to ensure that the transfer from the input



(a)



(b)

Fig. 3. (a) Direct form II transpose; for audio use. (b) Direct form I transpose; not for audio use.

to each of the sensitive internal nodes did not exceed unity, thus precluding accumulator overflow. [Notice that the *transpose* of the direct form II (canonic) topology shown in Fig. 3(a) is equally suitable for our purposes and possesses all the desired characteristics, but we will only work with the direct form I in this paper.] Scaling the input with respect to the internal nodes is, therefore, mandatory for the canonic topology.

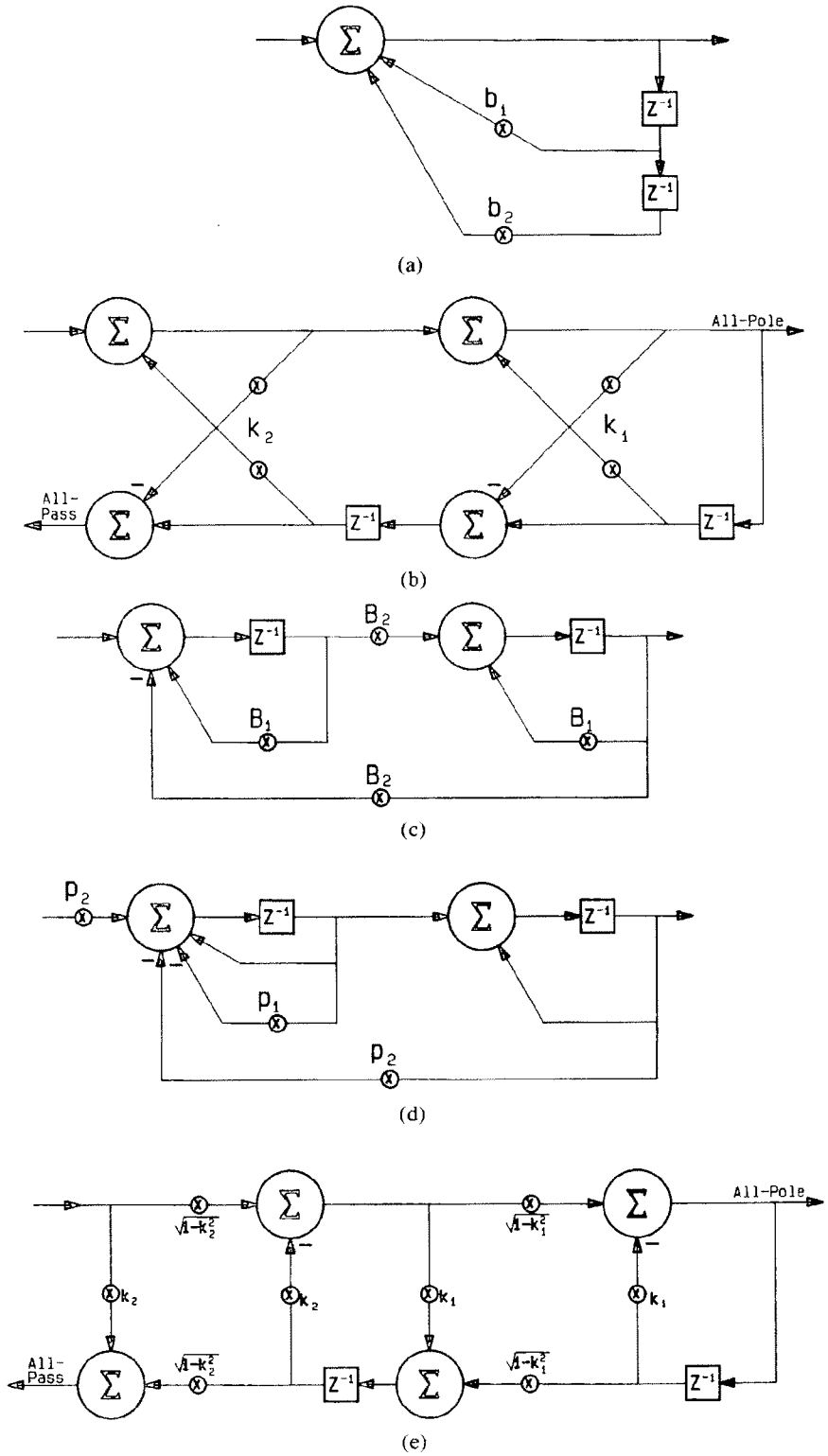


Fig. 4. Some second-order topologies. (a) Direct form I; has one accumulator. (b) Two-multiplier lattice; $k_1 - k_1k_2 = b_1$, $k_2 = b_2$. (c) Rader–Gold "coupled" or "normal" form in one guise. (d) Agarwal–Burrus. (e) Gray–Markel four-multiplier ladder.

So, we do not want to use it. Lower noise topologies such as the Agarwal–Burrus topology [7], [9] (which is a variation of the Rader–Gold topology) suffer from the same problem and thus require scaling.

1.1 Modulo Math

Two's complement addition can be thought of in rings or moduli. The circumference of the ring is dictated by the greatest magnitude representable by a given wordlength. As an example, consider the sum of two positive numbers that exceeds the available wordlength, which is represented pictorially in Fig. 5. The two positive numbers can each be represented at a location within the first ring. For the sum, we can visualize the result as having gone into the second ring, like the minute hand of a clock after one hour has elapsed. If a third number is now added which is negative and of the proper magnitude, then the clock hand will come back somewhere into the first ring and the result will be correct, even though the intermediate result was in error.

1.2 Unity Gain

By the term *unity gain filter* we mean a filter that is designed as a deviation from unity. Several unity gain filters are shown in Fig. 6. Although we implied in the preceding discussion that designing for unity gain will ensure no output overflow, there are three qualifications to this reasoning which compel us to perform overflow detection at the output on a per-sample basis and to saturate the output on detection of overflow there.

1.2.1 Boost Filters

Unity gain design implies that we cannot design boost filters. We will dispel this idea now. To design a boost filter (having a gain that exceeds unity) is fine as long as input signals falling in the frequency region of the boost are such that they do not exceed unity *after* boosting. If the request for a boost is because of a paucity of energy in the corresponding frequency region, then the threat of overdrive is small. So just as with analog filters, it is up to the user who requests a boost to ensure that the inputs are not overdriven. If this cannot be done conveniently, then the filter could be redesigned by dropping the absolute level of the transfer function. This would be equivalent to scaling the input by some desired amount. The point here is that there need be no inherent flaw in the filter implementation itself which demands the use of scaling at all times.

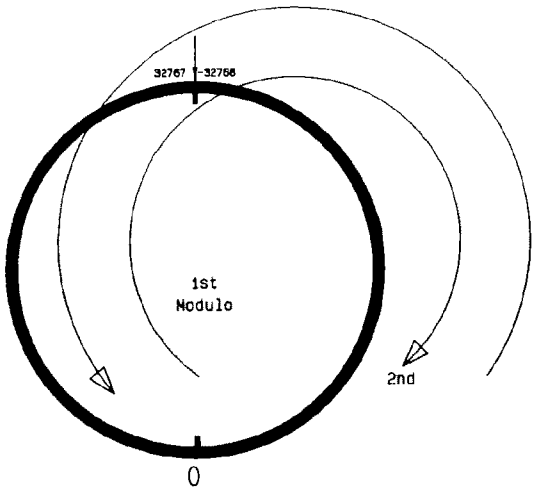
1.2.2 Obtuse Signals and the L_p Norm

It can be shown that there exist nonsinusoidal well-behaved input signals which will drive a unity gain filter output past unity. For example, the time-reversed impulse response of a filter, when used as input to that same filter, will drive the output past unity. (The interested reader is referred to the topic of "Matched filters" [10].) So designing a filter for a unity gain transfer function does not ensure the absence of overflow at the output for all input signals.

Input scaling criteria have been developed to prevent overflow at *any* node of a given topology, although the

EXAMPLE:

DESIRED RESULT	MODULO RESULT	
32512	32512	
+ 256	+ 256	
32768	-32768	(2nd Modulo)
- 768	- 768	
32000	32000	(1st Modulo)



Jackson's Rule: Any number of additions and/or subtractions may occur. Intermediate results and operands may fall into any modulo. As long as the final result is made to fall into the first modulo by design, it will be representable in two's complement at the chosen wordlength, and a valid result.

Fig. 5. Two's complement is a modulo arithmetic. The first modulo (ring) for 16 bits is shown. The arrows help visualize the traversal from the first into the second modulo and then back again.

sensitive nodes are only at multiplier inputs, as previously discussed. One of these criteria generates scale factors referred to as L_p norms, which are derived in the frequency domain in the following way:

$$||F||_p = \left(\frac{1}{\omega_s}\int_0^{\omega_s} |F(\omega)|^p \, d\omega\right)^{1/p} \quad (5)$$

where $F(\omega)$ is the transfer from the input to the sensitive node and ω_s corresponds to the sample rate in radians per second. A particular value of p is first chosen and the expression is evaluated for every sensitive node. L_p is then assigned as the inverse of the maximum $||F|_p$ with respect to all the nodes, and is then used as a scalar to the input of the filter section in question. The values of p usually adopted are 1, 2, or infinity. The case $p = 1$ ensures that the mean of the magnitude over frequency at the sensitive node will not exceed that of the input, while the case $p = 2$ ensures that the average power over frequency at the sensitive node will not exceed that of the input. When $p = \infty$, this norm is the most conservative of the L_p norms because in this case $||F||_p$ becomes simply the maximum of $|F(\omega)|$ with respect to frequency. Intuitively, we can understand why this is conservative because the assumption is being made that all the filter input energy resides at a single frequency [5]. This choice of p (infinity) is convenient, however, for test purposes.

Not being the most severe scaling criterion, the L_p norm will not guard against the time-reversed impulse response as input. In practice, though, such obtuse signals which can potentially overdrive filter outputs are infrequent [5], [7]. Because we use the direct form I, it is necessary to think about scaling the input with respect to only the output node. Therefore, while not precluding output overflow, the unity gain design criterion is usually sufficient corresponding to a p value of infinity in the frequency regions of precisely unity gain. The responsibility to ensure that boost filters are not overdriven is, again, left up to the user [35, p. 1008]. (Recording engineers typically work at approximately 8–24 dB below saturation to give themselves adequate clipping headroom.)

1.2.3 Overflow Prevention and Practice

Overflow detection must be performed at each output stage in a cascade *on every sample output*, and upon detection, saturation of the affected output (appropriately positive or negative) must take place. Regardless of the precautions taken to prevent output overflow, detection is mandatory under absolutely any and all conditions in a real-time operating environment. If output saturation is not performed, the filter output will be a hard nonlinearity, due to 2's complement wraparound, which is never audibly pleasant. If not performed on a per-sample basis, then upon recovery from the overflow condition, the filter can enter into weakly nonlinear modes which only show up as subtle but constant deviations from nominal THD + N measurements performed on the filter output for an input sinusoid. A theoretical explanation of these overflow oscillations is not given here, but can be found in the standard literature. We stress that saturation is implemented only for the final output of each stage, as pictorially represented in Fig. 7 (ignore the truncator Q box for now), and not during intermediate calculations.

The proper way to detect output overflow is discussed in Appendix 2.

1.3 Summary

By designing unity gain filters using direct form I structures, we can skirt the input scaling issue altogether by allowing intermediate accumulator overflow. The choice of the direct form I topology does not exclude us from considerations of the effects of recirculating output truncation noise, however, so we will have to deal with this noise problem in a different manner than by judicious choice of topology. For a good survey of standard topologies see [12].

2 TRUNCATION NOISE AND ITS PROPAGATION

The currently available DSP processors, such as the ADSP-2100 and TMS32010, routinely provide double wordlength products and accumulations. This is a tremendous advantage to the DSP algorithm designer because it obviates the need to consider truncation noise effects at any node other than the output node in the direct form I digital filter structure.

Accepting the fact that the direct form I is by no means the most quiet topology, we must now deal with truncation noise generated at its output. We must consider the effect that this noise has on filter operation when unavoidably fed back into that filter. First-time designers of digital filters tend to ignore this aspect of the implementation because it is considered a second-order effect. If the application is one that involves fidelity of reproduction or accurate frequency synthesis [23], then this aspect should not be ignored. On the other hand, if the primary interest is an implementation ending up in a low-end sound effects processor, then this topic need not be of concern.

Fig. 8 explicitly shows the direct form I topology having the nonlinear truncation operator (Q for quantizer) at its output. Fig. 7 shows how the truncator and the saturator are used in the same circuit. We omit the saturator in the remaining figures for clarity.

2.1 Low Signal Level Aberrations

The most apparent manifestation of the effect that output truncation has on the output signal of a second-order digital filter occurs when low-level signals are input. An anomaly emerges as a discrepancy between the transfer function of the filter for the full-level (unity amplitude) input as compared with the transfer function for an attenuated input signal. For ideal linear filters, the transfer function is supposed to be independent of the input signal level. Let us say that a sinusoid at an amplitude approximately 30 dB below unity is input to a 16-bit digital filter having a modest gain and low

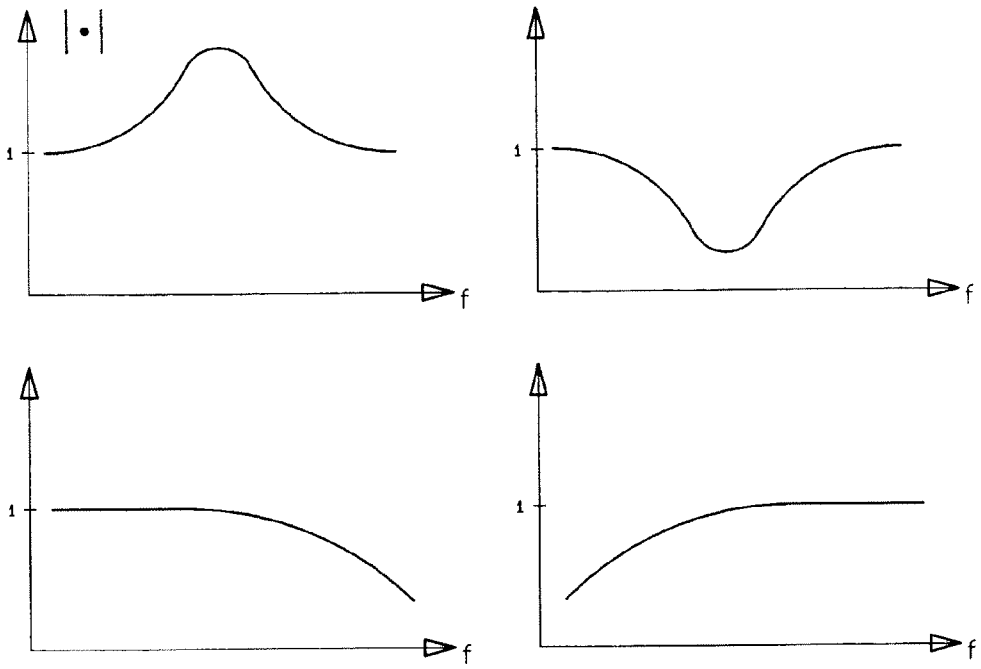


Fig. 6. Unity gain design.

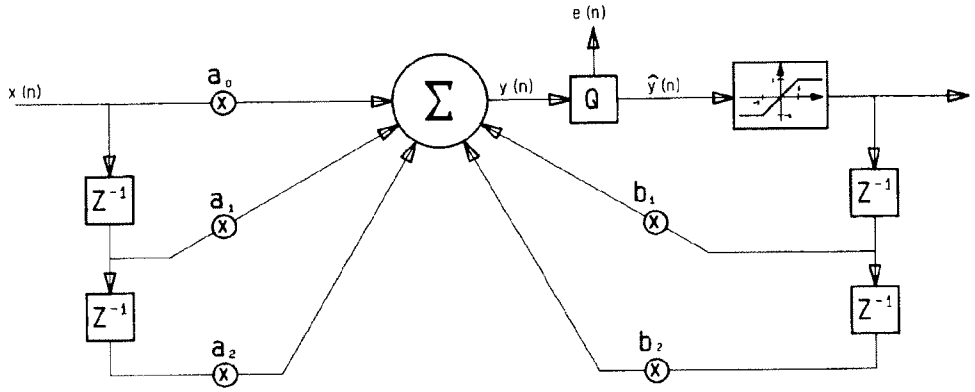


Fig. 7. Truncator Q and saturator.

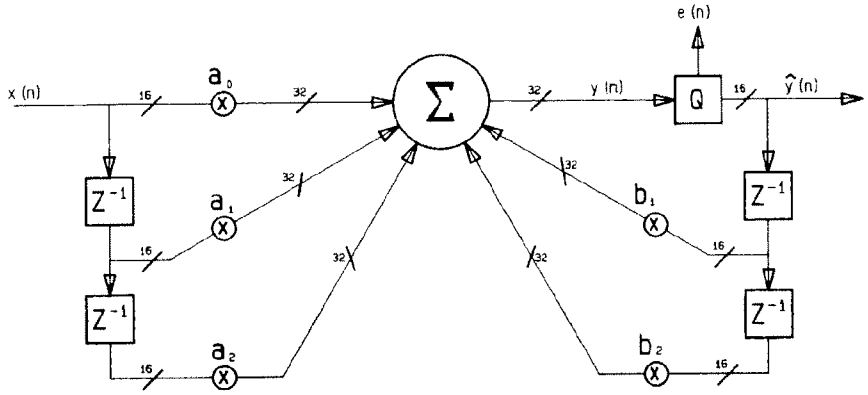


Fig. 8. Direct form I having truncator.

critical frequency. Then independent of the resolution of the coefficients, the output sinusoid will exhibit, to the eye, a variety of amplitude aberrations when viewed on an oscilloscope. These are a direct consequence of the nonlinearity introduced by truncating the accumulator output.

This effect is not hard to reproduce in the laboratory. The easiest one to see is called the *jump phenomenon*. This aberration was named by Kristiansson in 1973 during his study of it [13]. The phenomenon appears as abrupt shifts in the amplitude or dc level of a digitally filtered sinusoid.

The truncation operation is unavoidable because the inputs to the multipliers must be 16 bits while the accumulator accepts and produces 32-bit results. Also, the outside world is most likely 16 bits, so that regardless of the accumulation wordlength, the filter output must pass on 16-bit signal data. Unfortunately the amplitude of the transfer function at any one point along the curve is not a constant but is rather some function of time as caused by this nonlinearity. Needless to say, as the engineers of these digital filters, this performance falls short of our expectations.

The aberrations are even more pronounced at low input frequencies, less than 100 Hz, where one does not have to go as far down as −30 dB to start seeing these effects. They can be seen at input levels of 0 dB. If a common pen plotter is used to characterize a digital filter over frequency, then small ripples will be seen in the plot that are not the same among multiple plots of the same transfer. Further, in the case of 0-dB mid-frequency signals, even if the amplitude aberrations are not apparent on an oscilloscope or a pen plotter, then a THD + N analyzer picks up the problem by calculating a figure upward of 10 dB worse than nominal. Indeed, the gentlest filters will cause at least a 10-dB degradation in THD + N. If the amplitude aberrations are so bad as to be visible, then the degradation to THD + N will be much more than 10 dB.

There are at least two workable solutions to this problem, but let us first perform a simple analysis of the truncation error in an attempt to put a handle on it. Once this is done, the solutions will be straightforward.

2.2 Truncation Error Math

Referring to Fig. 8, we have designated the 32-bit quantity $y(n)$ as the fixed-point output of the digital filter while $\hat{y}(n)$ is the truncated output. We would like to derive a frequency domain expression for the truncated (actual) output signal $\hat{y}(n)$ in terms of the input and truncation error signals. To do this we may define the ideal output $y(n)$ as a 16-bit integer $\hat{y}(n)$ plus a signed fractional part, the truncation error signal $e(n)$,

$$y(n) \triangleq \hat{y}(n) + e(n) . \tag{6}$$

The input signal, generally a 16-bit integer, is called $x(n)$, allowing us to write

$$y(n) = \Sigma a_i x(n - i) + \Sigma b_i \hat{y}(n - i) . \tag{7}$$

This is the equation that would be used in a program to perform the filtering function. The 32-bit output $y(n)$ is then truncated by taking the integer part and storing it as $\hat{y}(n)$. We can gain more insight into Eq. (7) by first using Eq. (6) to rewrite it as

$$y(n) = \Sigma a_i x(n - i) + [\Sigma b_i y(n - i) - \Sigma b_i e(n - i)] . \tag{8}$$

Since we know the source of the error signal $e(n)$, it is not truly a random variable, neither is it uncorrelated to the output signal. Indeed, the error signal is deterministic because for the same input signal and coefficient set; it can be replicated exactly. Therefore we may justifiably assume that it has a frequency domain representation.

Then, using Eqs. (6) in (7) in the frequency domain,

$$\hat{Y}(z) = X(z) \frac{\Sigma a_i z^{-i}}{1 - \Sigma b_i z^{-i}} - E(z) \frac{1}{1 - \Sigma b_i z^{-i}} . \tag{9}$$

This important Eq. (9) states that the truncated-output spectrum of the classical direct form I digital filter topology is equal to the ideal filter transfer times the input spectrum plus some "error function," multiplying the truncation error spectrum $E(z)$. The truncation error spectrum is amplified by the poles of the filter transfer function *whether the filter boosts or cuts*, and is therefore highly correlated with the resonances of the filter. This is bad news because the gain introduced by the poles can be as large as 90 dB for some practical cases. It is interesting to note that $E(z)$ is not affected by the feedforward paths in this circuit. This is one of the reasons we chose this topology since this is an advantage.

Eq. (9) will serve as our reference. Any improvements made to minimize the effects of truncation error will be gauged to the last term in Eq. (9) since this is what we are left with if we do nothing about the truncation error problem.

2.3 Truncation Error Feedback Math

The first solution to impact the truncation error spectrum was suggested to us by a Philips Corporation document [14], which describes the DSP involved in the design of their oversampling D/A converters which have found wide use in Compact Disc players. Fig. 9 shows the first-order *error feedback* scheme. Here $\hat{y}(n)$ is again the 16-bit truncated (integer) value which goes to the outside world, while $y(n)$ is the 32-bit nontruncated output signal. The error signal $e(n)$ is a signed quantity formulated as implicit by Eq. (6),

$$e(n) = y(n) - \hat{y}(n) . \tag{10}$$

We like to define the error in this sense because it is sometimes a facile method from the programmer's point of view, where positive-signed inputs to the accumulator

will result later. This means that the error signal is formed by subtracting the 16 bits that form the integer part of the 32-bit accumulator output, from the accumulator itself. This assigns the signed fractional value to the error signal whose magnitude can, thus, never exceed 1.

In all our error feedback schemes we will never use rounding to form $\hat{y}(n)$ simply because it uses more program steps and gains us little advantage. Recall that statistically, the difference between rounding and truncation is a ½ LSB dc offset in the time average of the error signal. Hence the output signal spectrum also experiences a small dc offset when truncation is used. Rounding is used when the absolute dc value of a quantity is of concern; dc is of little concern in the audio signal.

Notice in Fig. 9 that the error signal $e(n)$ is delayed and then fed back to the accumulator. We show a multiplier $K = \pm 1, \pm 2, \pm 4$ one value (to be determined) multiplying the delayed error signal. In order to avoid the use of the hardware multiplier, K is constrained to a nonzero trivial binary integer. Although we will determine the optimal value for K given this constraint, later we will relax this constraint at the expense of more intensity. The 16-bit input to the filter structure is $x(n)$. We wish once again to find a frequency domain equation that expresses the truncated output signal in terms of the input and truncation error signals. As before,

$$y(n) = \Sigma a_i x(n - i) + \Sigma b_i \hat{y}(n - i) + Ke(n - 1) . \quad (11)$$

This is the equation we would use in our program. Notice the new term in Eq. (11) as compared with Eq. (7). $\hat{y}(n)$ is formed within the machine by truncating $y(n)$, while $e(n - 1)$ is formed using Eq. (10).

Using Eq. (6) in Eq. (11) we get

$$y(n) = \Sigma a_i x(n - i) + [\Sigma b_i y(n - i) - \Sigma b_i e(n - i)] + Ke(n - 1) \quad (12)$$

and in the frequency domain, using Eq. (6) on the left-hand side of Eq. (11),

$$\hat{Y}(z) = X(z) \frac{\Sigma a_i z^{-i}}{1 - \Sigma b_i z^{-i}} - E(z) \frac{1 - Kz^{-1}}{1 - \Sigma b_i z^{-i}} . \quad (13)$$

Eq. (13) has the same form as Eq. (9), except that the error function, the last term in Eq. (13), has been modified slightly due to the new error feedback path. What we wish to know at this point is whether the modification to the error function has diminished the impact of the truncation error $E(z)$ at all. We have already seen how the poles of the filter amplify the truncation error, as shown by Eq. (9). Does the introduction of a zero in the numerator of this new error function make it smaller in some way? The optimum choice of a zero location would be somewhere right upon the unit circle in the z plane because this would bring the magnitude of the error function down to absolute zero at some audio frequency. The error function now has a zero at $z = K$. But since K is constrained to be a nonzero binary integer, then the only two choices lay upon the real axis at $z = \pm 1$. A zero at $z = 1$ corresponds to 0 Hz, whereas $z = -1$ would correspond to a zero at the Nyquist. The choice at first appears to be arbitrary from a theoretical point of view. The acquisition of a zero in the error function will certainly diminish its impact in either the very low or the very high frequency region. We choose a positive-valued K (= 1) for empirical reasons discussed below.

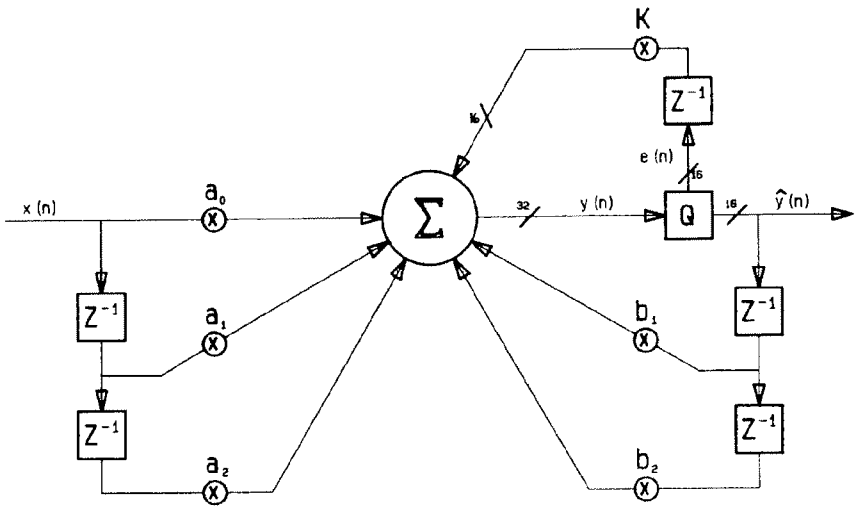


Fig. 9. First-order error feedback.

Fig. 10 shows sketches of the various error situations assuming $K = 1$. The horizontal line in Fig. 10(a) represents the magnitude of $E(z)$, here assumed to be white. Fig. 10(b) shows the error function of Eq. (9) multiplying $E(z)$, where $E(z)$ is amplified by the poles of some arbitrary second-order filter. Fig. 10(c) shows the new $1 - z^{-1}$ term by itself, from the error function of Eq. (13). Notice that there is a boost above unity in the high-frequency region in this graph. This graph shows that the new zero term will attenuate $E(z)$ at some frequencies, but will boost it at others. The attenuation at dc is complete. Fig. 10(d) shows the result of putting the zero into the numerator of the error function of Eq. (9), which results in the error function of Eq. (13) multiplying $E(z)$.

We believe that the situation shown in Fig. 10(d) is an improvement over that in Fig. 10(b), but it is not obvious why. Fig. 10 might persuade us that since the zero in the numerator acts to boost high-frequency truncation noise, we should not use this "error feedback" technique. But we have yet to contradict the classical assumption which states that the truncation noise spectrum $E(z)$ is white. A contradiction would have bearing on our decision. Our observations in the laboratory tell us that much of the energy in $E(z)$ lies in the low-frequency region. O'Donnell [15] independently observed a nonwhite truncation error spectrum on other second-order topologies and filter types years earlier. So the graph of the truncation error $E(z)$ might be more realistically portrayed in Fig. 10(a) as a low-pass signal. Knowing this, we choose to place the zero introduced by the first-order error feedback scheme at 0 Hz, which is the choice nearest to the frequency region where most of the truncation noise is found in practice.

There is a far more practical reason to choose to put the error function zero at dc, that is, if we know ahead of time that the boost of $E(z)$ by the poles of our filter predominates in the low-frequency region. Had we designed a filter whose critical frequency were nearer Nyquist than dc, we may well have decided to try $K = -1$ for an error function zero *at* Nyquist, our knowledge of $E(z)$ notwithstanding.

2.3.1 First-Order Error Feedback Summary

The degree of amplification of $E(z)$ by the poles of the filter function is now mitigated by the introduction of a new term in the numerator of the error function of Eq. (9), as shown in Eq. (13). It has been found that this trivial feedback scheme really helps to linearize an inherently nonlinear digital circuit. The amplitude aberrations that were seen in low-level filtered signals not having the benefit of error feedback, actually disappeared when the feedback was introduced, to the extent that the impact on a reference THD + N measurement was negligible. (The reference THD + N measurement was made bypassing the digital filters altogether.) Further, the scheme only requires three additional one-cycle TMS32010 instructions per stage, making it computationally attractive.

2.4 Second- and Higher Order Error Feedback

Fig. 11 shows a more powerful error feedback scheme, this time using two trivial multipliers for second-order feedback. Using reasoning similar to that above, the optimal K coefficient set places zeros on the unit circle in the error function closest to the anticipated frequency region of high noise boost by the filter poles [16]. These zeros do not adversely affect the filter *transfer* function in any way. Table 1 summarizes the viable choices of trivial multipliers and

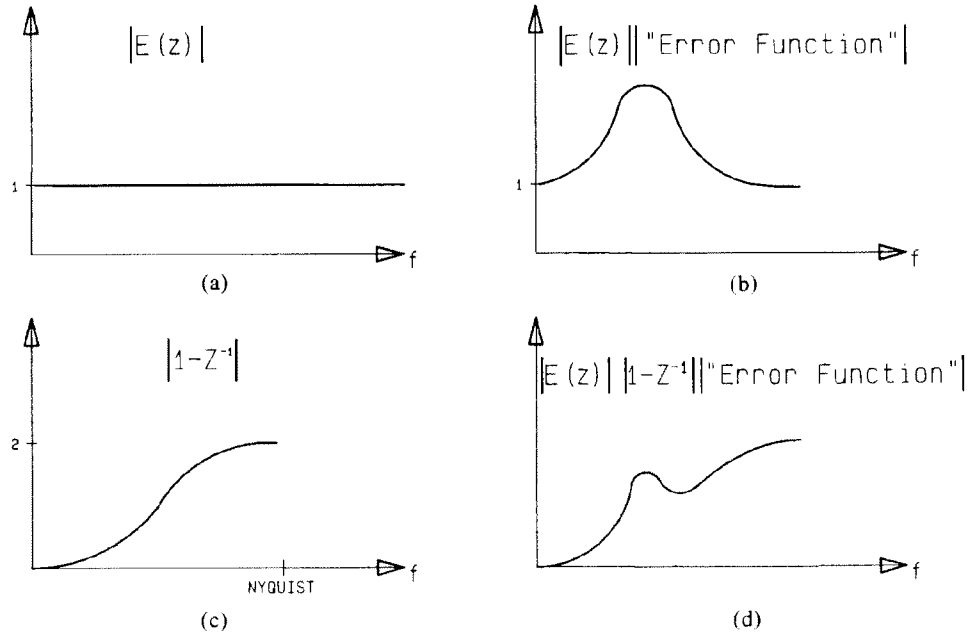


Fig. 10. Presumedly white truncation noise (a) is passed through a digital filter having an error function as in (b) and an error feedback network having the transfer (c), resulting in truncation error spectrum shaping (d).

their associated frequency regions, normalized to 2π.

While not unequivocally optimal, these error feedback schemes are economical, and their power should not be underestimated. Truncation noise recirculation is worst for filters of high Q or extreme critical frequency. As it turns out in audio filter design, you get the double whammy for extreme critical frequencies because poles need to be quite close to the unit circle to counteract the proximity effects of the mirrored (conjugate) pole. This is the reason why these feedback techniques are effective. They reduce the truncation noise in the filters that need it the most.

Still more elaborate feedback circuits have been devised such that the error function spectral amplitude is pushed predominantly into the high-frequency region [17] while squelching the lower frequency noise toward zero over any desired bandwidth. Of course, these schemes proportionately boost, by many times, the power of the high-frequency noise, but it is assumed that some other mechanism will be designed to filter that noise out.

These error feedback schemes have become collectively known in the literature as error spectrum shaping (ESS) [11], [18], [19]. All these DSP techniques accomplish the same goal: they selectively shape the truncation error $E(z)$. ESS techniques were developed with an eye toward trivial error feedback coefficients, but there is no necessity for this constraint if we are willing to pay the price. We now explore another ESS scheme which is optimal over the entire band, but just a bit more intensive than the error feedback schemes discussed so far.

2.5 Truncation Error Cancellation

We constrained the error feedback coefficients in our derivations above to be integers. We relax this constraint here. In Higgins and Munson [20] it is shown that the truncation error cancellation scheme, which follows, can be considered to be the optimal ESS filter, but with no constraints imposed on the feedback coefficients. Further, using a classical linear statistical analysis they show that the 16/32-bit truncation error cancellation scheme is equivalent in noise performance to a double-precision (32/64-bit) realization and outperforms linear state space (state variable) topologies. It is only when we go to a 24-bit processor employing truncation error cancellation that performance exceeds the double-precision realization.

We point out two advantages of the following scheme over a standard double-precision implementation: 1) only the signal feedback paths require double-precision bit widths, and 2) the multiplier inputs are always of the signed variety.

2.5.1 Wordlength Considerations

This new error "cancellation" scheme was presented in a thesis by Rothaar [21], where wordlength of coefficients and signal paths were of great concern. The criterion in that thesis was that no degradation past the 16-bit noise floor of the input signal would be tolerated over the entire audio band for all filter settings. It was

Table 1. Error feedback zeros.

K_1	K_2	Region θ
+2	−1	0 Twice
−2	−1	π Twice
0	+1	0 and π
+1	−1	$\pi/3$ Twice
−1	−1	$2\pi/3$ Twice
+1	0	0 Once
−1	0	π Once

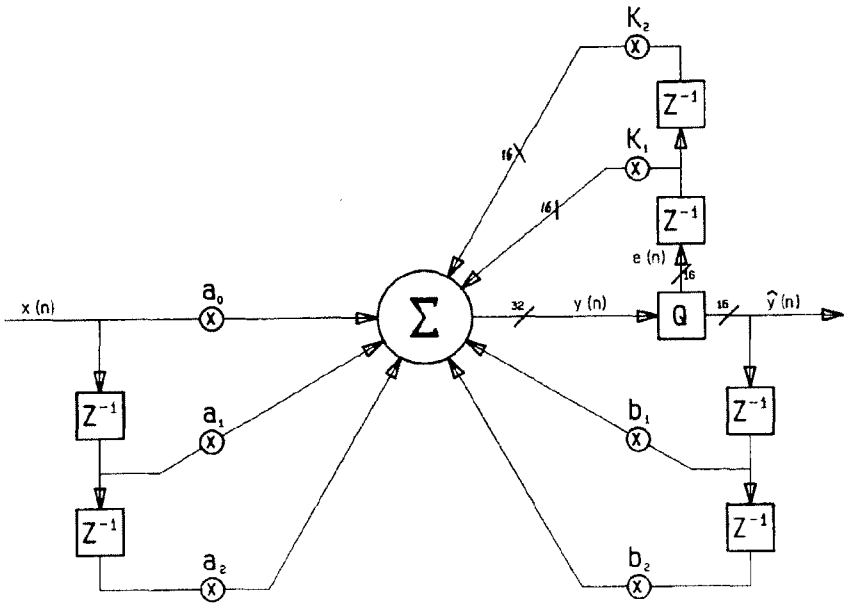


Fig. 11. Second-order error feedback.

shown that to achieve that performance from a standard second-order section *without* error cancellation required at least a 34/54-bit (multiplier/accumulator) architecture. This means 34-bit signal paths. (54 bits was the limitation on the simulator.) It was illustrated how filters having low critical frequencies place the most demands on the second-order topology. A wordlength of 24 bits was ultimately settled on for the coefficients; 16 bits was not enough. The signal data paths followed suit, resulting in a 24/48-bit system with error cancellation.

Rothaar's 24/48-bit system using error cancellation is ideal for the most demanding high-fidelity audio digital filtering applications since the 24-bit internal signal paths guarantee that the filters themselves operate with about a 138 dB SNR. It is advantageous to have the interstage signal paths exceed the 16-bit input wordlength because in this manner each stage's output-truncation noise propagating through a cascade can be kept well below the signal noise floor, since only the *last* filter in the chain need truncate its output at the 16-bit level. Assuming that truncation noise would accumulate through a cascade of N error-canceling second-order stages according to the absolute worst-case classical assumption of correlated noise sources [10 log $(N^2\sigma)$], this still means that hundreds of filters could be cascaded before the DSP designer would even need to *consider* its effects on a 16-bit signal. On the other hand, and in the best case of the classical assumption concerning *un*correlated truncation noise propagation [10 log $(N\sigma^2)$] the passing of 16-bit signal data between stages results in an immediate buildup above the −90-dB signal noise floor, approximately 3 dB after one stage, 6 dB after two, 9 dB after four, 12 dB after eight, and so on. (The best case, that is, uncorrelated noise sources, is usually the classical assumption taken, and although the least conservative of the classical analyses, it is the one most proven in practice. The worst case would double the decibels in the preceding list. For more on classical noise analysis, see [22, chaps. 3, 7], [5, chap. 11] [4, chaps. 9, 8].)

2.5.2 Signal Justification

If we agree that the coefficients need be 24 bits wide for sufficient control over low critical frequency filters, then given a 24/48-bit system, the question arises as to where one places the 16-bit input signal within the 24-bit register. We have infinite accumulator headroom as discussed earlier, so we need not worry about overflow. Since it is those bits to the right of the binary point that determine noise performance, we want to position the input signal with full left justification because the binary point follows the LSB of the 16-bit (Q0) input signal and we want as many fractional places as possible. We might think of this left justification as a scaling upward, by 48 dB, of the 16-bit input signal with no loss to the original quantization SNR of 90 dB. Now if the input signal absolutely must be scaled downward, as in a high-order filter design, there is some room. Regardless of the justification, we would always want to pass the 16-bit integer plus 8-bit-fractional output signal to the next stage in a cascade for the reasons discussed above.

There is a trend in DSP chip design that provides for headroom bits in the accumulator, that is, the accumulator bit width exceeds the product register width. The section of this paper on scaling and overflow has shown this feature to be superfluous for IIR digital filtering applications.

We now show that the technique of truncation error cancellation will keep the noise level of each individual 16/32-bit second-order stage at the 16-bit boundary. The only significant noise introduced by each second-order stage itself will come when the one truncation is performed at its output.

2.6 Truncation Error Cancellation Math

Referring again to Fig. 11, we wish to write a frequency domain equation for the truncated output in terms of the input and truncation error signals. This time we will not place any constraints upon the error feedback coefficients K_i.

$$y(n) = \Sigma a_i x(n - i) + \Sigma b_i \hat{y}(n - i) + \Sigma K_i e(n - i) . \tag{14}$$

This is the equation that we program. The multiplications involving the fractional quantity $e(n - i)$ are performed using no scaling because it is well represented numerically, having no leading zeros. The decimal point in the error accumulation is higher up in the accumulator, though, and it must be shifted to the right before combining it with the signal accumulation.

Using Eq. (6),

$$y(n) = \Sigma a_i x(n - i) + [\Sigma b_i y(n - i) - \Sigma b_i e(n - i)] + \Sigma K_i e(n - i) . \tag{15}$$

If we set the K_i equal to the b_i, then they cancel each other and Eq. (15) becomes

$$y(n) = \Sigma a_i x(n - i) + \Sigma b_i y(n - i), \qquad K_i = b_i . \tag{16}$$

The 16-bit error feedback paths in Fig. 11 now go to 32-bit widths after the multiplications because they are no longer trivial. In the frequency domain, using Eq. (6) in Eq. (16),

$$\hat{Y}(z) = X(z)\left(\frac{\Sigma a_i z^{-i}}{1 - \Sigma b_i z^{-i}}\right) - E(z) . \tag{17}$$

Comparing Eq. (17) to Eqs. (13) and (9), this is the best situation for the truncation noise that we have yet encountered since the error function is essentially gone as there is no amplification of $E(z)$ due to the poles of the filter. Eq. (17) says that the truncated output spectrum is equal to the ideal filter function of the input

spectrum plus the (negative) truncation error spectrum. The error function is decoupled from the filter transfer function.

This is indeed the best we can ever do with respect to truncation noise performance. It just does not get any better than this, no matter what circuit topology is used. Since the ideal output $y(n)$ is a fixed-point number having a nonzero fractional part, when we convert the double-precision output of the accumulator $y(n)$ to its 16-bit single-precision truncated value $\hat{y}(n)$, the truncation error spectrum unavoidably remains. But this is all that remains. The truncation error does not feed back. If we can accommodate the added computation time due to the two new feedback paths in Fig. 11, then this result is fabulous because we have a way to implement extremely quiet filters. We mention that this technique would be indispensable for high-purity digital oscillator applications [23].

Notice that the requirements for perfect error cancellation in Eq. (15) are 1) the signal feedback coefficients must equal the error feedback coefficients; nothing is said concerning the precision of those coefficients; and 2) no rounding may be performed in the formulation of the error accumulation.

2.7 Truncation Noise Spectrum $E(z)$

If $E(z)$ would add only 3 dB to the filter noise floor when $\hat{y}(n)$ is formed, then why should we be so concerned with it? The reason is because these second-order filters will most likely end up living in a cascade. (Higher order filters are often formed this way.) One interesting application is that of cascading a number of unity gain second-order filters to configure what is called a *parametric equalizer* in which each second-order filter exerts influence over a disparate part of the audio band. Typical configurations will see four cascaded second-order stages that could add a total of 9 dB [10 log $(N\sigma^2)$] of truncation noise to the −90-dB filter noise floor. We need in that case to consider the use of a larger wordlength processor (>16 bits) at a premium.

The assumption that the truncation error spectrum $E(z)$ is white is, more often than not, false. We reiterate that the preponderant error spectrum energy was empirically found to be concentrated in the low-frequency region. It has been said that the output truncation noise could be much like that of analog-to-digital quantization noise. This might be erroneously rationalized following the garden path: The output $y(n)$ of our second-order filter could be made ideal by using error cancellation, as shown by Eq. (16). If the input to an ideal filter is a (sampled) sinusoid, then because it is a linear system, the output must also be a pure sinusoid. Although the input sinusoid $x(n)$ was described using all integer values, the ideal $y(n)$ must be described by the filter using fixed-point (not necessarily integer) values. A truncation to an integer is required to form $\hat{y}(n)$ from $y(n)$, which should be very much like the quantization performed by an A/D converter in a successive approximation when it gets down to the LSB estimate. . . .

2.7.1 The Barnes Criterion

Barnes et al. [24] make the argument that quantization noise and multiplier roundoff noise are very different phenomena. Although he limits his scope to the output of an isolated multiplier followed by a quantizer, we should be able to extrapolate his results to the present case which sees two multipliers input to an accumulator where the quantizing operation follows the summation; see Fig. 12. (In the standard direct form I structure we have seen that the truncation error $E(z)$ is impacted only by the two feedback paths, the b_i. This is one reason why we chose that structure.) We will make no attempt here to perform a statistical analysis that would support this intuitive leap.

Barnes points out that while the conditions under which quantization noise can be considered spectrally white are quite mild, the corresponding conditions for multiplier roundoff noise are comparatively much more restrictive and highly dependent on the multiplier coefficient value. Specifically, he states that roundoff (or truncation) error will be white, uniformly distributed, and uncorrelated with the signal, in general, only if the standard deviation of the input signal is greater than approximately one half the full amplitude (unity) signal level.

Narrow-band and low-level signals each violate the criterion. In contrast, if the dynamic range of a wide-band signal exceeds only a few quanta, then its quantization noise can be accurately modeled as white in keeping with the traditional analysis. But it was demonstrated in [25], which dealt only with the topic of quantization noise, that even for full-level (unity gain) sinusoids, the ratio of sample rate to sine frequency can easily be adjusted to produce a range of quantization error characterization from purely harmonic to white spectra, the point being that the spectrum of quantization noise is highly signal dependent. Therefore we have an argument that quantization noise is not *always* white.

We hope to have supported the hypothesis, by these arguments, that the character of the truncation noise $E(z)$ is not generally white, but is highly dependent on both the input signal and the values of the feedback coefficients. It stands to reason, then, that in the particular case of a cascade of N second-order stages, where the feedback coefficients are made to be disparate, we might reasonably expect that the truncation noise would amass according to the usual classical assumption [that is, the least conservative assumption, 10 log $(N\sigma^2)$] of uncorrelated noise sources, because the contribution to the total error spectrum energy from each stage is likewise disparately splattered about the audio band.

So, let us see how our 16/32-bit truncation error feedback system fares in a little experiment.

2.8 Physical Measurement Example

In the implementation that follows we use 32-bit double-precision coefficients, as opposed to adequate 24-bit, only because it is convenient to do so. As it stands with the TMS32010, we are restricted to a 16/

32-bit system with error cancellation. So we are forced to perform the coefficient multiplications using double-precision arithmetic, which might be tricky due to the lack of an unsigned multiplication in the instruction set. But an elegant method of double-precision multiplication will be illustrated when *residual coding* of the filter coefficients is discussed.

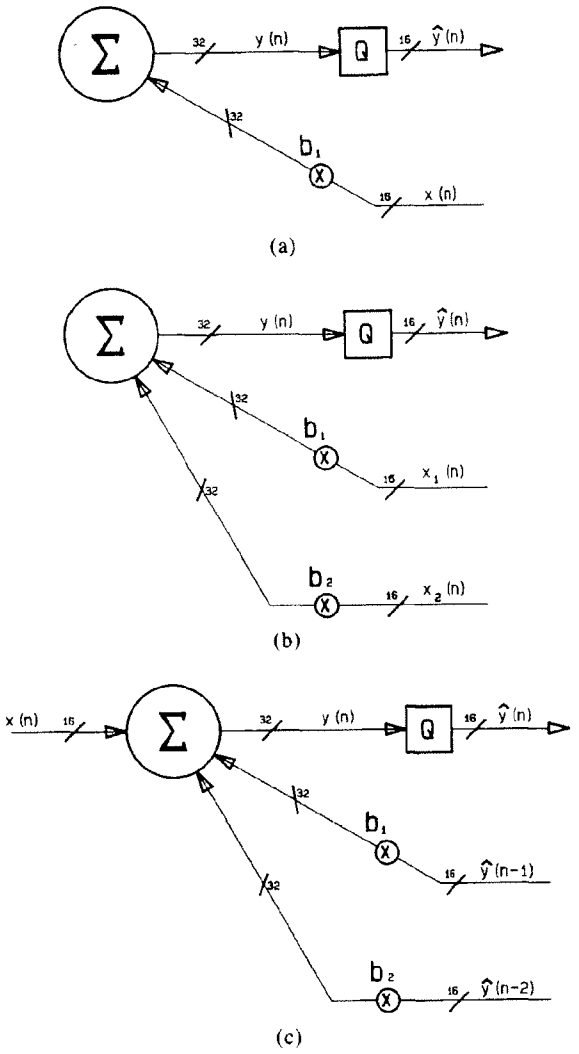


Fig. 12. (a) Barnes's equivalent formulation. (b) Logical extension. (c) Intuitive leap.

We would now like to substantiate the preceding work with actual measurements. Noise plus distortion was measured using a Tektronix 5000 series audio analyzer and oscillator. The A/D/A conversion process was provided by a Sony PCM 701. Table 2 shows the THD + N measurements for three digital filter test setups. Each setup transfer is sketched in Fig. 13. The first and third setups are cascades of four second-order parametrics, while the second setup consists of a single second-order section implementing a 60-Hz notch filter having an incredibly high Q of 1200 and a gain of −60 dB. Both cascade setups arbitrarily sequenced the second-order stages from lowest to highest critical frequency. All these filters are minimum-phase unity gain designs [1], use truncation error cancellation and double precision (32-bit) coefficients via residual coding, as presented later in this paper. Only the error cancellation coefficients are single precision, which could be a liability. The TMS320 code actually used is given in Appendix 1.

The reference THD + N levels were measured with the digital filters completely out of the measurement path, that is, bypassed. Since there is no signal processing going on during these reference measurements, we can attribute all the noise above the theoretical −90-dB noise floor of a 16-bit system to quantization noise introduced by the A/D/A conversion process. Any noise later introduced above the reference level is most likely then due to either the truncation process or amplification of the quantization noise floor by boost stages. Since each THD + N measurement of a sinusoidal input encompasses the full audio bandwidth, the measurement is repeated with various input frequencies only to provide diverse situations.

2.8.1 Cut Filter Measurement

For test setups 1 and 2, the cut filter cases, the THD + N level recorded in Table 2 was arrived at by subtracting the actual level of attenuation of the unity amplitude input sinusoid from the (negative) instrument

Table 2. Digital filter noise plus distortion measurements.

		THD + N (dB)		
Sinusoid input frequency (Hz)	Noise reference level (dB) (Filters bypassed)	Q = 20 A = −18 dB f_c = 50, 500, 5000, 15 000 Hz (Test setup 1)*	Q = 1200 A = −60 dB f_c = 60 Hz (Test setup 2)*	Q = 20 A = +18 dB f_c = 50, 500, 5000, 15 000 Hz (Test setup 3)*
50	−80.0	−82.2	−81.8	−79.5
100	−81.2	−80.8	−81.6	−73.0
500	−82.7	−82.8	−82.0	−78.5
1000	−83.0	−82.7	−82.3	−78.0
5000	−82.8	−82.0	−82.2	−79.0
10 000	−82.5	−82.3	−82.0	−77.0
15 000	−82.6	−83.0	−82.1	−80.0

$F_{S_{Actual}}$ = 44 470.45 Hz. "Unity" ≙ 0 dB. All setups use error cancellation.

* Test setup 1—four cascaded presence filters having negative gain; Test setup 2—notch filter; Test setup 3—four cascaded presence filters having positive gain.

reading. This legitimate compensation serves to make the measurements independent of the particular filter transfer under test. We do this because we are interested in the *S*/(THD + N) at the filter output from *unity* to the noise floor. We presume that the noise floor is the variable here.

Looking at Table 2, we see that the attenuating cascade and the notch are doing quite well. They are introducing little noise above the reference levels in this particular system. This is shown by Table 2, which compares the noise measurements for test setups 1 and 2 with the reference noise levels. If the reference levels were closer to −90 dB, we might run into a problem here with this 16/32-bit filtering scheme. Remember that regardless of whether the filters under test boost or cut, they still *add* truncation noise. Some of the truncation noise may be getting filtered out by succeeding stages in the cascade.

We note in passing that this −60-dB notch filter is stable and that it would be difficult to implement in analog due to the required component tolerances. The poles of this filter lie precariously close to the unit

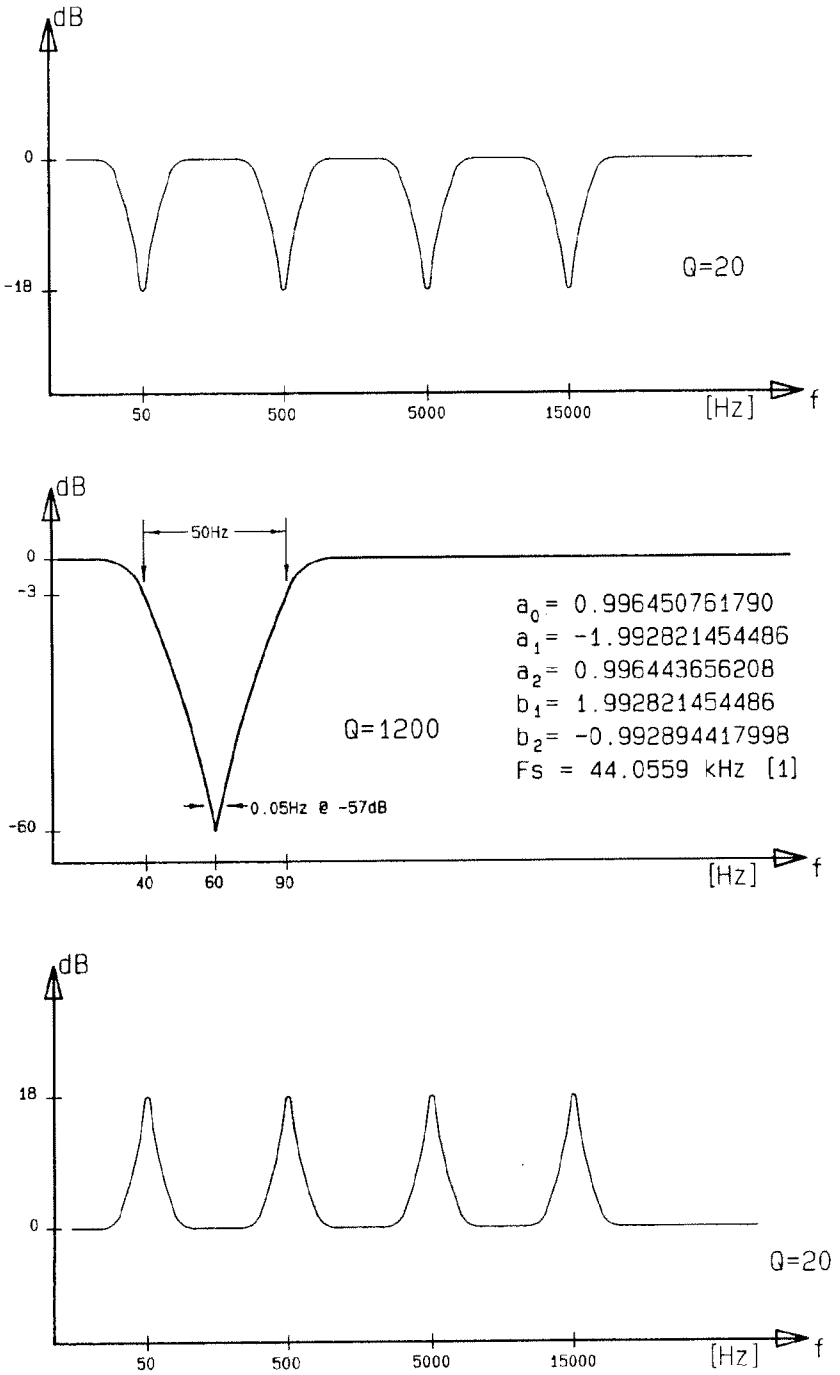


Fig. 13. Filter test setups 1, 2, 3.

circle, as this is an extreme-case design [1], [26]. The noise gain of this topology, not using the truncation error cancellation network, would be 85 dB, the gain of the poles, making this filter unusable (THD + N $\approx$ -5 dB) without error cancellation. (See Eqs. (30) for $b_1 = 1.992821454486$ and $b_2 = -0.992894417998$ at a sample rate of 44.0559 kHz [1].) As it stands, we are seeing a THD + N that is less than 1 dB beyond the system noise reference. This filter has been described as "surgical," seeing that its bandwidth (at -57 dB) is only 0.05 Hz at the null, while the (-3-dB) bandwidth at the skirts is 50 Hz. All the critical parameters of this filter were verified to within the resolution of the measurement apparatus. Resolution in frequency = 0.01 Hz, resolution in level = 0.1 dB.

2.8.2 Boost Filter Measurement

In the case of test setup 3, the amplifying cascade, the input signal level was adjusted downward below unity until the output signal level reached unity. The data in Table 2 indicate that test setup 3 is not doing too well. We're seeing increases of up to 8.2 dB above reference. But it must be remembered that the boosting is above unity and *any* noise falling beneath each respective parametric will be amplified. Even if there were no truncation noise, the system noise floor might be boosted, thus interfering with our desired measurement of the increase in truncation noise. If we calculate the worst-case total boost of white noise across the audio band by all four parametrics, it is about 7.6 dB. Since all four boost filters have the same Q factor (already quite high at 20), the majority of the noise boost comes from the higher critical frequency filters because they span more bandwidth. We might say that it must be the quantization noise floor that is getting boosted, but the THD + N is varying so much with input frequency that we are forced to conclude that it must be truncation noise that is getting boosted. This is unfortunate because it means that we need to squelch the truncation noise further. The only way this can be accomplished with the topology we have chosen is to use a 24-bit processor, but we still need the truncation error DSP.

2.9 Going Further

Had we not used truncation error cancellation in the experiment, all the THD + N readings would have been horrific. The experimenter can easily verify that the filter transfers we chose are actually quite difficult designs to implement. We were more concerned about whether we needed to use a 24-bit processor than whether we needed truncation error cancellation. A bit of experience tells us that we just cannot get along without some form of error feedback, and so that was taken for granted.

Probably the best way to measure the noise produced by a particular digital filter is to implement it using a DSP chip, and then monitor the output with a noise measurement device. Computer simulations are sometimes cheaper and there are programs available. One such program is DINAP II: A Digital Filter Analysis Program [27] from Purdue University. This program performs noise analyses of user-specified topologies in the time or frequency domain.

2.9.1 Midsummary

Thus far we have been able to alleviate any input-level scaling requirements through a discerning choice of filter topology, and the truncation noise problem has been grasped. We examined digital signal processing techniques known as error spectrum shaping (ESS), which encompass both truncation error feedback and cancellation. The primary purpose of ESS is to lower the operational noise floor within each filter that we implement. We decided that it would be best to use a 24-bit processor and some DSP in the form of ESS to improve truncation noise performance for the most demanding applications. If we have such a processor at our disposal, then 24-bit coefficients are necessary and adequate for sufficient control over low critical frequency filters. In this case we can skip the next section on residual coding.

If we are using a 16-bit processor, then we will need to find a way to gain precise control over the placement of poles and zeros in the low-frequency region. Once found, then we will have tackled the three major obstacles standing in the way of high-quality digital filtering. In the section that follows we will show how to encode double-precision coefficients on any DSP chip that does not have the ability to perform unsigned multiplications.

3 RESIDUAL CODING OF COEFFICIENTS FOR EXTREME CRITICAL FREQUENCY DESIGNS

This method of coefficient encoding is intended primarily for the TMS32010, TMS32020, or any DSP chip where unsigned multiplications are problematic. The reason for its contrivance is to elegantly perform double-precision multiplications. Specifically, we would like to use filter coefficients having 24-bit wordlengths, but our multiplier only accepts signed data having wordlengths of 16 bits. It will be convenient to use 32-bit wordlength coefficients, so we will do so.

3.1 Residual Coding In a Nutshell

Consider the purely numerical problem stated as follows:

$$y = ax$$

where a is a double-precision and x a single-precision binary integer. If we express a as having a high-order part a_H and a low-order part a_L at the bit level, then we can say

$$y = a_H x(\text{with suitable shift}) + a_L x$$

$$[\text{signed}] = [\text{signed}][\text{signed}] + [\text{unsigned}][\text{signed}] \ .$$

The second multiplication requires an unsigned input,

so we cannot use a multiplier that only performs signed arithmetic. We could force the MSB of a_L to be zero by shifting it 1 bit to the right before input to the signed multiplier, but we lose precision, albeit 1 bit. Instead, we redefine a_L to make it signed:

$$a_L \triangleq a - a_H .$$

Now a_L is no longer set to the low-order bits of a double-precision binary integer; it is a signed quantity. In the detailed explanation that follows, keep in mind that the complexity of the implementation is no better or worse than that of a conventional double-precision implementation using unsigned arithmetic.

3.2 The Details

Let us name the generic single-precision coefficient c_i. This is the fixed-point decimal representation of the ideal floating-point coefficient, which we have seen in our filter topology. The ideal floating-point filter coefficient c_{Fi} is calculated by a filter design routine. We define the signed difference between the ideal and the single-precision coefficient as ec_i and dub it the *residual* coefficient.

We formulate c_i in fixed-point decimal using the following:

$$c_i = \frac{\text{round}(c_{Fi}2^{qc})}{2^{qc}} \quad . \tag{18}$$

where the round function means round to the nearest integer. We use the round function here because it will later bound the residual coefficient to $\pm 0.5/2^{qc}$ which will be advantageous when we encode it. The quantity qc is the number of binary places following the binary point in the coding of the single-precision coefficient c_i.

To "code" a coefficient means to find a 2's complement representation of the fixed-point decimal number. To code c_i,

$$\text{binary code}(c_i) = c_i 2^{qc} = \text{round}(c_{Fi}2^{qc}) \ . \tag{19}$$

The resulting encoded integer, often expressed in hexadecimal, is used directly in assembly code, representing a 16-bit fixed-point binary number having qc binary places after the binary point. Therefore the sign and integer part must be representable in $[16 - qc]$ binary places. If qc were 12, then we would say that c_i is a Q12 coded number (= [4.12]); a Q12 number, for short.

The residual coefficient is formulated in fixed-point decimal as

$$ec_i = c_{Fi} - c_i \ . \tag{20}$$

The residual coefficient is encoded using an additional scale factor that is equal to 2^{qc}, because it is too small to be represented adequately in a 16-bit fixed-point format. Of couse, the filter program must account for the scaled residual coefficient in the calculations. Due to the rounding procedure, after scaling, the residual coefficient will have a decimal magnitude no greater than 0.5,

$$\text{binary code}(ec_i) = \text{round}(2^{qc}\ (ec_i 2^{qe})) \ . \tag{21}$$

[Numerically, it has been found that it is better *not* to simplify Eq. (21) by substituting previous equations and then canceling like terms.]

The residual coefficient has qe binary places following the binary point, and qe is not necessarily equal to qc. A good choice for qe might be 15. In that case we would say that ec_i is coded as a Q15 number, meaning that there are 15 binary places following the binary point, one sign bit, but no integer bits [1.15]. For stable second-order filter sections having unity gain and minimum phase, a good choice for qc would be 14 (that is, Q14 = [2.14], having one integer bit) since the magnitude of the coefficients is bounded by 2. Due to idiosyncrasies of the TMS32010, we will set qc equal to 12 in order to save four program steps per filter stage, and qe to 14 for maximum headroom in the separate residual coefficient accumulation.

This notational method of working with fixed-point binary numbers in terms of Q follows those conventions set forth in the TMS32010 user guide [28]. The Q notation follows the same rules as exponents in a by-hand multiplication. For example, if the multiplier were a 16-bit Q12 number and the multiplicand were a 16-bit Q14 number, then the product would be a 32-bit Q26 number, that is, the sum of the two Q factors. (Note the redundant sign bit in the product of any fractional multiplication.)

3.3 Filter Calculations Using the Residual Coefficients

Now that we have encoded the double-precision coefficients, we are ready to perform the digital filtering itself. We tally five double-precision coefficients for the standard second-order section plus two single-precision coefficients for the truncation error cancellation network. This is equivalent to 12 single-precision coefficient multiplications in terms of computation time. The price we have paid for these hi-fi filters is an amount of computation that exceeds that for the basic second-order section by a factor of roughly 12/5. (In comparison, for a 24/48-bit architecture the expense factor would only be 7/5 because all the coefficients would be in single precision which, of itself, entails even less housekeeping.)

To justify the use of single-precision coefficients in the error cancellation network, refer to Fig. 14. There,

$$\begin{aligned} y(n) = {} & [\Sigma a_i x(n - i) + \Sigma e a_i x(n - i)] \\ & + [\Sigma b_i \hat{y}(n - i) + \Sigma e b_i \hat{y}(n - i)] \\ & + \Sigma b_i e(n - i) \end{aligned} \tag{22}$$

where for the sake of less nomenclature, we call the double-precision coefficient the same as the floating-point coefficient since

$$c_i + ec_i \approx c_{Fi}$$

(double precision on the left, floating point on the right).

Eq. (22) is the one that is actually implemented in the code in Appendix 1.

Using Eq. (6), we can rewrite Eq. (22) as

$$\begin{aligned} y(n) = {} & \Sigma a_i x(n - i) + \Sigma e a_i x(n - i) \\ & + [\Sigma b_i y(n - i) - \Sigma b_i e(n - i)] \\ & + [\Sigma e b_i y(n - i) - \Sigma e b_i e(n - i)] \\ & + \Sigma b_i e(n - i) \ . \end{aligned} \tag{23}$$

Using Eq. (20) and simplifying, we get

$$\begin{aligned} y(n) = {} & \Sigma a_{Fi} x(n - i) + \Sigma b_{Fi} y(n - i) \\ & - \Sigma e b_i e(n - i) \ . \end{aligned} \tag{24}$$

If we compare Eq. (24), which is the single-precision case of truncation error cancellation, with Eq. (16), aside from the use of double-precision coefficients in Eq. (24), the only difference is the (last) error term in Eq. (24). This excess error term arises because we chose *not* to use double-precision (that is, the same) coefficients in the error cancellation network. Ideally, we would like to use double-precision error cancellation coefficients. Keep in mind that perfect truncation error cancellation, as in Eq. (15), is not dependent on the precision of the error feedback coefficients. The signal feedback coefficients must be *equal* to the error feedback coefficients. In that case, the error feedback becomes error "cancellation."

If we look at Eq. (24) in the frequency domain, using Eq. (6) to derive an expression for $\hat{Y}(z)$, then we get an excess error term at the truncated output that looks like this:

$$E(z) \frac{1 - \Sigma(b_{Fi} - eb_i)z^{-i}}{1 - \Sigma b_{Fi} z^{-i}} \ . \tag{25}$$

If the eb_i are small, then the numerator of this error function should approximately cancel the denominator, which comprises the filter poles. The use of single-precision coefficients in the error cancellation network is probably sufficient in most cases except those demanding high precision, which encompass filters of high Q factor or (extreme) critical frequency close to dc or Nyquist. Recalling that the eb_i are bounded in magnitude by $(0.5)2^{-qc}$, then the eb_i will be smallest when the b_i have the greatest possible precision (that is, the highest resolution). So we now have a theoretical reason to prefer the greater precision Q14 representation of the single-precision part b_i of the double-precision coefficient b_{Fi} rather than the Q12 representation that is used in our TMS code at the present time.

3.4 Residual Coefficient Implementation

When we get down to writing the actual program code for Eq. (22), since the residual coefficients are encoded using a scale factor, they must be accumulated separately. The error cancellation term must also be accumulated separately, but for a different reason. It

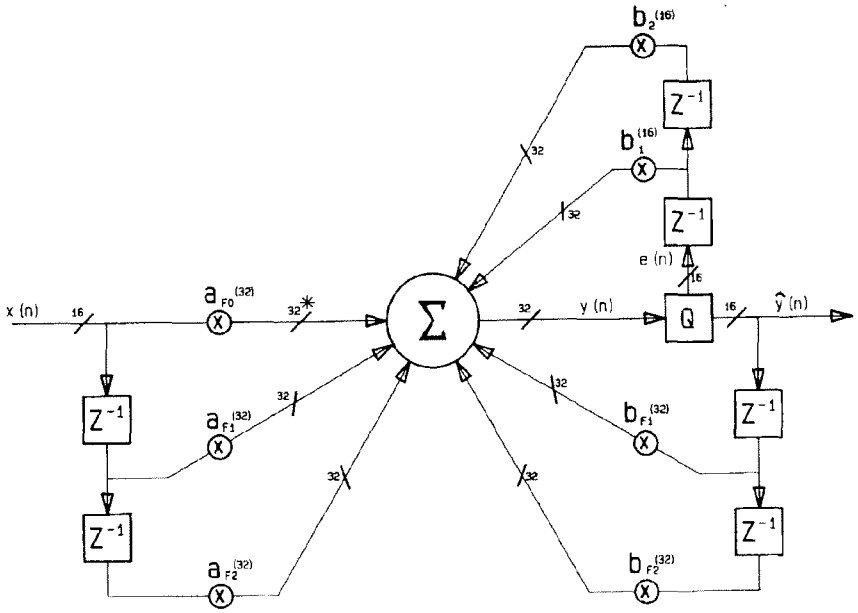


Fig. 14. Truncation error cancellation and residual coefficient coding. See TMS code in Appendix 1 for implementation of path alignment.

is because the error signal $e(n)$ is not Q0 as are the input and truncated output signals; it is Q12, the same Q value as the single-precision coefficients. The most efficient order of calculations is to first group all the residual coefficients, then perform the truncation error cancellation, and finally the single-precision coefficients.

Both the error cancellation and the residual accumulation must provide for worst-case headroom in the accumulator. Overflow is not acceptable at these stages in the calculations because they are nonlinear. In the case of the residual accumulation, since the input and output signals are Q0 and the magnitude of each of the five residual coefficients never exceeds 0.5, a potential overflow factor of 2.5 requires that there be two headroom bits. Thus the residual coefficients need to be in Q14 format ($qe = 14$). After the residual coefficient scale factor 2^{qc} is accounted for, the accumulator ends in Q26 format.

In the case of the error cancellation, the Q12 error signal has a maximum magnitude of 1.0, whereas the Q12 feedback coefficients are bounded in magnitude by 2.0 and 1.0, respectively. Therefore a potential overflow factor of 3.0 again requires two headroom bits. But the accumulator will be in Q24 at the end of the error cancellation accumulation. This means we can toss up to five of the MSBs; in the code we discard four for the sake of convenience.

When these two separate accumulations are combined with the accumulation of the single-precision coefficients, we may then take advantage of the infinite accumulator headroom and disregard intermediate overflow, as discussed earlier.

Refer to the TMS320 code in Appendix 1 for an explicit presentation of all the principles discussed in this paper.

3.5 Going Further Still

The purpose of residual coefficient coding is to make the actual frequency response more closely resemble the shape of the theoretical frequency response, while one purpose of truncation error feedback is to make the actual frequency response less dependent on absolute input signal level. The most straightforward way to predict the effects of coefficient quantization on transfer function is to use the desired precision coefficients in the theoretical computation of the transfer. This technique is a valid first-order approximation of the truth and will show the trends of shift in frequency response due to the drift of the pole–zero locations. While coefficient resolution primarily affects the shape of the filter transfer and stability, its impact on noise performance and other signal quantization effects is second order. Coefficient quantization alone does not make the digital filter a nonlinear system. The best method to predict all quantization effects on frequency response is to take the Fourier transform of the actual impulse response of the filter under study. Programs to do this are available; one such program is included in a filter design package from Signix Corp. [15], which can calculate the FFT of up to 8192 points (version 1.5). Very low critical frequency filters may need more points to capture the complete impulse response.

4 FORCED OVERFLOW OSCILLATIONS

In the first three sections we have covered the most important topics concerning the implementation of digital filters. The current topic concerns the nonlinear behavior of second-order digital filters once output overflow with saturation has occurred and the input remains nonzero. While it is important that the practicing engineer be made aware of this potential problem, this topic is not essential to the understanding of the other material. This phenomenon, it must be stressed, is characteristic of many digital filter topologies and is not peculiar to our particular implementation. All the error feedback techniques we have shown have been reported to minimize this problem which we are about to discuss [29]. The simplest means of circumventing forced overflow oscillations is to ensure somehow that the filter output will not exceed unity.

4.1 Statement of the Problem

Recall that our direct form I structure is immune to overflow problems at *internal* nodes as long as the output is constrained to be less than unity. When the saturating output tries to exceed unity, all bets are off—linear analyses no longer apply and the filter output can be observed, under the proper circumstances, to "lock up" in a mode of oscillation which follows the input frequency but not its amplitude or phase. At the instant this lock-up response occurs, the output phase can be observed to jump suddenly to a new value, which is frequency dependent but difficult to predict. The input signal is present during this form of instability, which is the reason for the term *forced* overflow oscillation. The oscillation will cease when the input signal is taken away (brought to zero). Unfortunately our venerable error feedback techniques cannot help us because the output is no longer representable at the allotted bit width, causing the truncation errors to be themselves in numerical error.

If the input signal is merely reduced in amplitude, then the filter will return to its linear behavior after the input amplitude has dropped to a level that can be considerably lower than the level that elicited the response. (This return to linear behavior will happen, assuming that output overflow detection is performed on a per-sample basis, as previously discussed.) Conceptually, there is an input level hysteresis-loop function that describes this lock-up mode of the filter. The amount of hysteresis is primarily dependent on the pole positions.

A rule of thumb for sinusoidal input signals is this: if the input frequency is below the pole frequency, then there will be *no* forced overflow oscillation *hysteresis*, that is, the oscillations will stop as soon as the input stops overdriving the filter. But when the reverse is true, when the pole frequency is lower than the input

frequency, there *will* be hysteresis. We will, shortly, be more precise in our definition of exactly when the problem occurs, but this rule of thumb is good for filters having poles at less than half Nyquist.

The topic of forced overflow oscillations should not be confused with the more widely researched topics of overflow oscillations and limit cycles. These are the only two types of low-level autonomous oscillations possible in digital filters. We briefly present them here.

4.1.1 Overflow Oscillations

Autonomous overflow oscillations will occur if output saturation is not implemented upon the detection of overflow there. (Remember we do not saturate intermediate results.) This type of instability is self-sustaining once activated, and is due to the wraparound (modulo) characteristics of 2's complement arithmetic. Output saturation eliminates the problem for second-order filters, and so we can dispense with it entirely [30]. It would be nice if saturation solved the forced overflow oscillation problem, but it does not. The fundamental difference between this and forced overflow oscillation is that these oscillations will sustain under zero *and* nonzero input conditions, while forced overflow oscillations only occur under nonzero (large-signal) input conditions.

4.1.2 Limit Cycles

Limit cycles, on the other hand, are a problem due to the inherent nonlinear nature of digital filters provoked by numerical inaccuracies in the signal paths. Limit-cycle oscillations can be sustained under the condition of zero input and are dependent on initial conditions. Activation of limit cycles does not require an overflow mechanism; this is the primary conceptual difference. Nonzero inputs usually break them up, once activated. The cause of these oscillations can be understood intuitively if we realize that even when digital filters are operating in their "linear mode," they are still nonlinear devices to a second degree because of the quantization of the internal signals and the truncation errors. "Digital" filters, by definition, operate on signals that are discrete in both time and amplitude. Therefore no matter what number of bits are being used in the signal path, there will always be the possibility of limit-cycle oscillations in the standard direct forms. The amplitude of these oscillations can be quite large, although typically small and annoying, and the problem worsens when rounding as opposed to truncation is used. Indeed, Chang [31] has shown that limit-cycle oscillations are impossible in first-order digital filters when truncation is used. This limit-cycle problem has usually been solved in the past by increasing the number of bits in the signal paths, which reduces the amplitude of the oscillations to "acceptable levels."

In the error cancellation and feedback schemes we have presented, we diminished the impact of the truncation errors by constraining them to appear only at the output node. We accomplished this by canceling or minimizing the truncation errors which fed back into the circuit, and by retaining full precision multiplications. Hence the internal signal paths appear much wider than they actually are because we succeeded in preventing a buildup of truncation noise along those paths. The net effect is the same as it would be had we actually increased the signal bit width. The amplitude of any potential limit cycle is greatly minimized compared to that using no form of error cancellation. Thus using cancellation/feedback (ESS) techniques, we succeed in linearizing the digital filter to a much higher degree than do conventional implementations. Indeed, Higgins and Munson [16] showed that these ESS schemes quickly approach statistical equivalence to double-precision implementations, which in itself guarantees significantly improved limit-cycle suppression. Going one step further, Chang [31] showed that for specific values of the error feedback coefficients, limit cycles could be eliminated completely. Those values include the case of truncation error cancellation discussed.

In summary, using ESS techniques, one is able to suppress limit cycles to well below the 16-bit level. We will not examine limit cycles further here because thorough treatments can be found in most standard DSP texts.

4.2 Region of Forced Overflow Oscillation

Fig. 15 shows the second-order digital filter section under study here. Only the feedback paths are shown because the feedforward paths do not contribute to this problem. The saturation arithmetic shown at the output of the accumulator is in keeping with the assumptions we have made thus far. Fig. 7 shows our implementation of the truncator and saturator operators as found in the code in Appendix 1. We believe that their ordering is not critical; this figure is only included here for reference. The truncator is not critical in this analysis and so it is omitted in Fig. 15.

Fig. 16, taken from [32], shows a typical forced overflow oscillation at the output when a sinusoid overdrives the circuit. We talked about the observation that the forced overflow oscillation seems to be worse when the input frequency is above the pole frequency, that is, it is harder to recover from it. Let us hypothesize a possible explanation for this by guessing that recovery time is somehow related to the impulse response of the filter. Let us loosely define "recovery time" as the time it takes for the circuit to stop oscillating once the input signal has been removed.

If we talk about an input sinusoid whose amplitude is just slightly above unity, then it is really just the peaks of that sinusoid which overdrive the unity gain filter. Then let us guess that the input amplitude hysteresis phenomenon has to do with the fact that a high-frequency input sinusoid has peaks which occur at a rate that is too fast for the overflowed filter to recover from. Because the filter does not have adequate time to recover from this nonlinear mode, it is necessary to bring the input amplitude far enough below unity so

that the corresponding amplitude of the filter's impulse response becomes attenuated. This in turn reduces the time required before the internal computations (convolutions) are once again representable at the allocated bit widths. Conversely, when the input sinusoid has a frequency that is below the pole frequency and its amplitude is just above unity, then the overflowed filter has plenty of time to recover and it is not necessary to bring the input amplitude back down much below unity.

Although this explanation may be oversimplified, having used the linear impulse response in the explanation of a nonlinear phenomenon, Claasen and Kristiansson [33], [34, p. 515] proved in 1975 that a necessary and sufficient condition for the existence of forced overflow oscillations is

$$|b_2 h(n)| > 1, \qquad \text{for some } n = 0 \rightarrow \infty \tag{26}$$

where b_2 is the second-degree feedback coefficient and $h(n)$ is the impulse response of the ideal linear second-order digital filter.

Using the residue theorem, the impulse response of the filter shown in Fig. 15 can be derived from its transfer

$$H(z) = \frac{1}{1 - b_1 z^{-1} - b_2 z^{-2}} = \frac{1}{(1 - pz^{-1})(1 - p^* z^{-1})} \tag{27}$$

as

$$h(n) = \frac{p^{n+1} - (p^*)^{n+1}}{p - p^*}, \qquad n = 0 \rightarrow \infty \tag{28}$$

where $p = re^{j\theta}$ is the pole and p^* is the conjugate pole of this second-order system. (Note that consideration of zeros in the numerator of Eq. (27) would only change

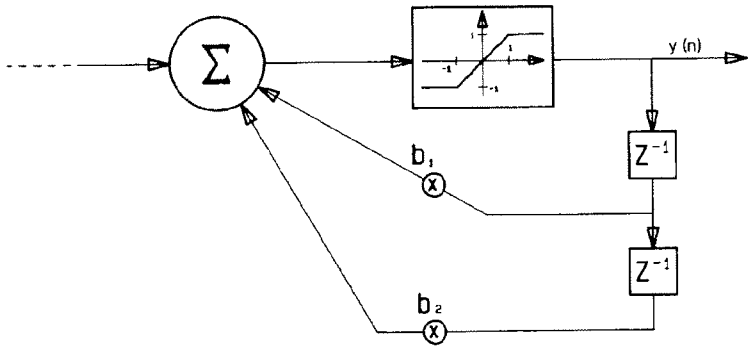


Fig. 15. Saturator, forced overflow analysis.

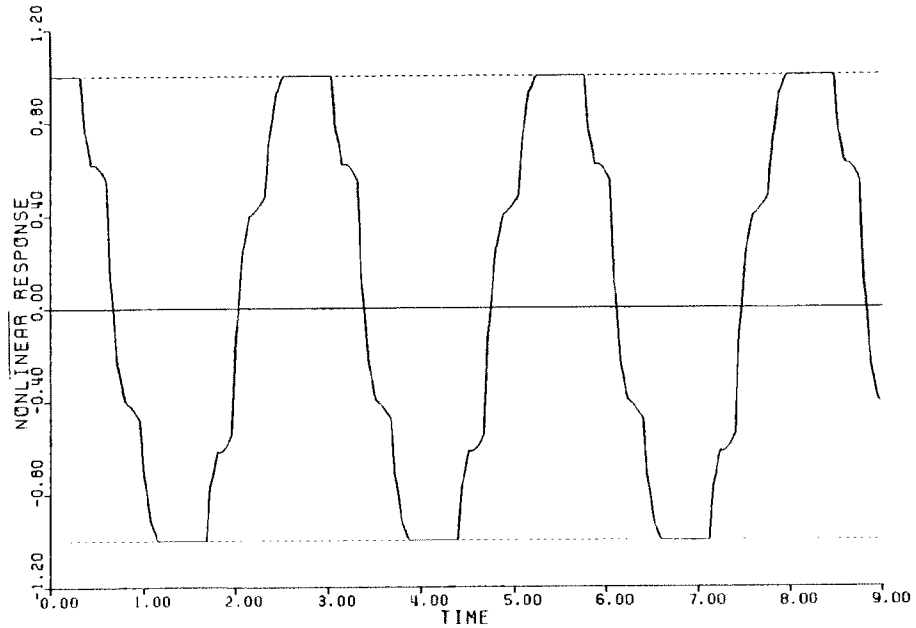


Fig. 16. Typical forced overflow response for sinusoidal input. (© 1983 IEEE [32].)

the absolute magnitude of the consequent impulse response, but not its shape.)

We can rewrite Eq. (28) in its polar form as

$$h(n) = r^n \frac{\sin (n + 1)\theta}{\sin \theta} \tag{29}$$

where r is the pole radius in the z plane and θ is the pole angle in the upper half-plane. The pole radius r must be within the unit circle for stability, as is well known.

For θ very close to 0 or π,

$$h(n) \simeq (n + 1)r^n , \qquad \theta \text{ near dc or Nyquist} .$$

Relating the poles within Eq. (27) to the actual filter coefficients, it is straightforward to show that for the positive topology

$$b_1 = 2r \cos \theta$$

$$b_2 = -r^2 . \tag{30}$$

The filter stability expressed in terms of the filter coefficients b_1 and b_2 is illustrated by the stability triangle in Fig. 17. All coefficients within the triangle produce stable filters.

Now it is easy to see that there may easily be forced overflow oscillations for poles near dc or the Nyquist since there, by substituting Eq. (30) and the approximation to $h(n)$ into the criterion, Eq. (26),

$$|(n + 1)r^{n+2}| > 1 ,$$

$$\text{for some } n = 0 \rightarrow \infty , \qquad \theta \text{ near 0 or } \pi$$

it is not hard to find an n for which it is true.

Fig. 17, taken from [32], [34], shows the coefficient regions shaded within the stability triangle for which the second-order digital filter is likely to have a forced overflow problem. The trend of the problem area is for values of b_2 close to -1, which implies that the problem is more severe for filters whose poles are close to the unit circle. The trend as it pertains to b_1 shows problems for filters whose poles lie near dc or Nyquist. Notice that the problem areas are symmetric with respect to the b_2 axis, and that there appear nulls for particular values of b_1 for which no forced overflow oscillations are possible.

Fig. 17 suggests possible recovery schemes that involve the modification of the coefficients to bring them into safe regions. This solution implies that the filter transfers could be changed momentarily to solve the problem when it occurs. This may not be attractive in some circumstances. It may be more desirable to scale input amplitude. A nonlinear approach such as that described by Claasen and Kristiansson in [29] may be more effective. The technique they describe employs a feedback circuit that only comes into play when overflow occurs. Assuming there is some headroom in the accumulator, they feed back the difference between the saturated output and the actual value of the overflowed signal. Under normal circumstances of linear operation, this difference is zero, so nothing gets fed back.

4.3 Forced Overflow Summary

Our goal is to make the DSP engineer aware of this problem. Interestingly enough, there are only a handful of research papers on this subject, and after all is said and done, the solution typically offered is to simply reduce the input amplitude; this is now the best solution

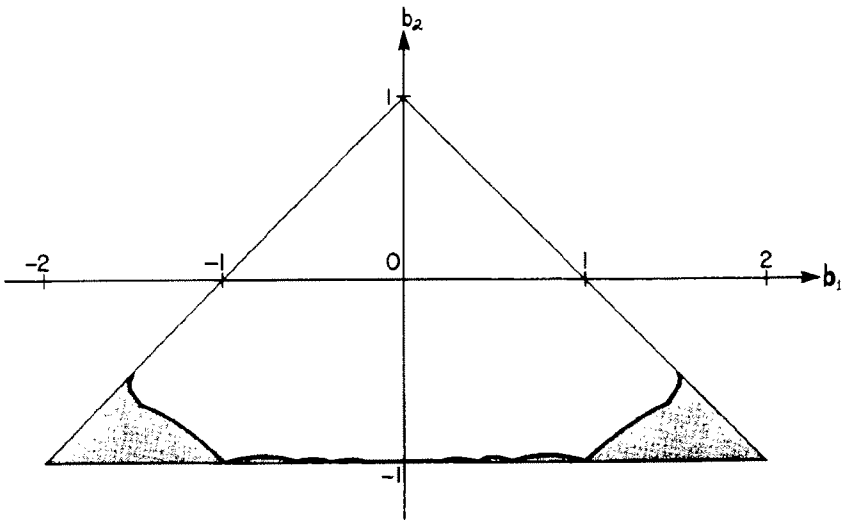


Fig. 17. Stability triangle for positive topology. Shaded region shows where forced overflow oscillations can occur. (© 1982, IEEE [32], [34].)

[35]. But whose responsibility is it to do so, the filter architect's or the end-user's?

5 CONCLUSIONS

One thing that has become eminently clear during the course of this research is that, in terms of noise performance, we could have extremely high fidelity (transparent) digital filtering if we used 24-bit signal paths. For the most demanding applications, the 16/32-bit processors just will not do, unfortunately. 24-bit signal paths will not, however, obviate the use of truncation error cancellation. 32/64-bit processing would be needed to perform brute-force digital filtering using no truncation error cancellation. But note that 24/48-bit processing with truncation error cancellation will outperform the 32/64-bit system and can be up to 8 bits (48 dB) better because the 32/64-bit system can only hold the noise down to the 16-bit boundary. We have also learned that 24-bit coefficients have sufficient accuracy for most audio frequency filtering requirements. From this perspective, a 24/48-bit version of the TMS320 series of DSP chips would be ideal.

Too often the lesson has been learned that the number and orders of the filters designed are determined not by the desired transfer characteristics, but by the sheer processing power of the implementation. For this reason, commercial filter design packages should be amended so that the principal constraint is the filter order rather than band tolerances. The program would then yield the best possible design for the given filter order.

The primary contribution of the topics we have covered is to dispel some popular myths, particularly those concerning input scaling, internal overflow, and the assumptions of "white" noise ("random") error sources. We have found that our truncation noise reduction techniques also serve to eliminate autonomous limit-cycle oscillations, while output saturation ensures freedom from autonomous overflow oscillations. Having hurdled these barriers, we have nearly reached the ideal in digital filtering. There is always room for more work, and we suggest that the most elegant solution to forced overflow oscillation is still unclear. The solution may surface in the area of chaotic system theory.

6 TENETS OF DSP FILTER IMPLEMENTATION

1) Overflow is not always a bad thing.

2) Truncation noise can be controlled at the expense of computational intensity. The degree of intensity determines the amount of control.

3) Signed multipliers *are* useful for double-precision calculations without loss of accuracy or efficiency.

4) Floating-point mathematics alone in a DSP chip does not solve the truncation noise problem [36], [37]. Numerical resolution determines noise performance. Resolution is determined by the mantissa bit width. Truncation of the mantissa to less than the required $2N$ bits prior to accumulation increases truncation noise recirculation.

5) Dynamic range is not the same as signal-to-noise ratio (SNR). Dynamic range is the ratio of the largest to the smallest *perceptible* signal level. Since signals well below the noise floor are perceptible by humans, the dynamic range specification often exceeds SNR. It is often made commensurate with the 6-dB-per-bit PCM criterion, but even this can be exceeded by proper dithering [38]. Dynamic range, then, is determined by the digital signal processing. SNR is the ratio of the largest signal representable to the noise floor with no signal present. In digital systems it is probably more meaningful to have a term for that ratio when signal is present: this we call THD + N. Included in it are harmonic distortion, the noise floor, and all other artifacts introduced by the digitization of the signal itself.

6) Coefficients of second-order sections may be linearly interpolated for smooth transitions to target filters without intermediate instabilities. The same is not guaranteed for direct higher order filters (except ladders). The interpolated coefficients must be encoded in double precision for smooth transitions.

7) The propagation delays of digital filters are no worse, in general, than for the corresponding analog implementations.

8) DSP chips should have the capability of shifting the entire accumulator in place and at once in either direction to facilitate the alignment of the intermediate accumulations in the realization of the techniques presented herein. The program-controlled realignment of the product register into the accumulator is also useful. (TMS32020 and TMS320C25 have these capabilities.) DSP chips must be designed to handle the access of long delay lines of any modulo and with nonequispaced taps. DSP chips need to be able to detect overflow out of several of the high-order accumulator bits below the MSB, not just from the MSB.

7 ACKNOWLEDGMENT

The author is indebted to the following people for their contributions to this paper: Richard Cabot of Audio Precision, Hayley Greenberg of Analog Devices, Tom Hegg of Lexicon, and Dana Massie of Next.

8 REFERENCES

[1] J. A. Moorer, "The Manifold Joys of Conformal Mapping: Applications to Digital Filtering in the Studio," *Audio Eng. Soc.*, vol. 31, pp. 826–841 (1983 Nov.).

[2] E. M. Cherry, "Transient Intermodulation Distortion—Part I: Hard Nonlinearity," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-29, p. 137 (1981 Apr.).

[3] L. B. Jackson, J. F. Kaiser, and H. S. McDonald, "An Approach to the Implementation of Digital Filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-16, p. 413 (1968 Sept.).

[4] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing* (Prentice-Hall, Englewood Cliffs, NJ, 1975).

[5] L. B. Jackson, *Digital Filters and Signal Processing*, 1st ed. (Kluwer Academic Publishers, Boston, MA, 1986).

[6] J. D. Markel and A. H. Gray, Jr., "Fixed-Point Implementation Algorithms for a Class of Orthogonal Polynomial Filter Structures," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-23, p. 486 (1975 Oct.).

[7] R. C. Agarwal and C. S. Burrus, "New Recursive Digital Filter Structures Having Very Low Sensitivity and Roundoff Noise," *IEEE Trans. Circuits Sys.*, vol. CAS-22, p. 921 (1975 Dec.).

[8] L. B. Jackson, "Roundoff Noise Bounds Derived from Coefficient Sensitivities for Digital Filters," *IEEE Trans. Circuits Sys.*, vol. CAS-23, p. 481 (1976 Aug.).

[9] K. W. Martin and M. T. Sun, "Adaptive Filters Suitable for Real-Time Spectral Analysis," *IEEE Trans. Circuits Sys.*, vol. CAS-33, p. 218 (1986 Feb.).

[10] A. B. Carlson, *Communication Systems, An Introduction to Signals and Noise in Electrical Communication*, 2nd ed. (McGraw-Hill, New York, 1975).

[11] P. S. R. Diniz and A. Antoniou, "Low-Sensitivity Digital-Filter Structures Which Are Amenable to Error-Spectrum Shaping," *IEEE Trans. Circuits Sys.*, vol. CAS-32, p. 1000 (1985 Oct.).

[12] H. T. Nagle, Jr., and V. P. Nelson, "Digital Filter Implementation on 16-Bit Microcomputers," *IEEE Micro Mag.*, p. 23 (1981 Feb.).

[13] L. Kristiansson, "Jump Phenomenon in Digital Filters," *Electron. Lett.*, vol. 10, p. 14 (1974 Jan. 24).

[14] Philips Corp., "News Report on Philips 16-Bit Digital to Analog Conversion System," Elcoma Marketing Communications Group, Bldg. BA, Eindhoven, The Netherlands, News Rep. 82906 (1982 Apr. 16).

[15] J. O'Donnell, "Digital Signal Processing Software—For the Evaluation of Digital Filters," Signix Corp., 19 Pelham Island Road, Wayland, MA 01778, DISPRO version 1.5 (1986).

[16] W. E. Higgins and D. C. Munson, Jr., "Optimal and Suboptimal Error Spectrum Shaping for Cascade-Form Digital Filters," *IEEE Trans. Circuits Sys.*, vol. CAS-31, p. 429 (1984 May).

[17] H. A. Spang III, and P. M. Schultheiss, "Reduction of Quantizing Noise by Use of Feedback," *IRE Trans. Communic. Sys.*, vol. CS-10, p. 373 (1962 Dec.). Reprinted in *Waveform Quantization and Coding* (IEEE Press, New York, 1976).

[18] C. T. Mullis and R. A. Roberts, "An Interpretation of Error Spectrum Shaping in Digital Filters," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-30, p. 1013 (1982 Dec.).

[19] D. C. Munson, Jr., and B. Liu, "Narrow-Band Recursive Filters with Error Spectrum Shaping," *IEEE Trans. Circuits Sys.*, vol. CAS-28, p. 160 (1981 Feb.).

[20] W. E. Higgins and D. C. Munson, Jr., "Noise Reduction Strategies for Digital Filters: Error Spectrum Shaping versus the Optimal Linear State-Space Formulation," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-30, p. 963 (1982 Dec.).

[21] B. C. Rothaar, "A Digital Audio Tone Control," M.S. Thesis, Dept. of Computer Science, University of Utah, Provo (1982 June).

[22] G. R. Cooper and C. D. McGillem, *Probabilistic Methods of Signal and System Analysis* (Holt, Rinehart & Winston, New York, 1971).

[23] A. I. Abu-El-Haija and M. M. Al-Ibrahim, "Improving Performance of Digital Sinusoidal Oscillators by Means of Error Feedback Circuits," *IEEE Trans. Circuits Sys.*, vol. CAS-33, p. 373 (1986 Apr.).

[24] C. W. Barnes, B. N. Tran, and S. H. Leung, "On the Statistics of Fixed-Point Roundoff Error," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-33, p. 595 (1985 June).

[25] J. M. Halbert and R. A. Belcher, "Selection of Test Signals for DSP-Based Testing of Digital Audio Systems," *J. Audio Eng. Soc. (Engineering Reports)*, vol. 34, pp. 546–555 (1986 July/Aug.).

[26] G. T. Yan, "New Digital Notch Filter Structures with Low Coefficient Sensitivity," *IEEE Trans. Circuits Sys.*, vol. CAS-31, p. 825 (1984 Sept.).

[27] S. C. Bass, J. W. Grundmann, and S. E. Belter, "DINAP II: A Digital Filter Analysis Program," School of Elec. Eng., Purdue University, West Lafayette, IN, TR-EE 78-14 (1978 Mar.).

[28] Texas Instruments, "TMS32010 User's Guide," Digital Signal Processor Products Division, Houston, TX, Doc. SPRU001B (1985).

[29] T. Claasen and L. Kristiansson, "Improvement of Overflow Behaviour of 2nd-Order Digital Filters by Means of Error Feedback," *Electron. Lett.*, vol. 10, p. 240 (1974 June 13).

[30] P. M. Ebert, J. E. Mazo, and M. G. Taylor, "Overflow Oscillations in Digital Filters," *Bell Sys. Tech. J.*, vol. 48, p. 2999 (1969 Nov.).

[31] T. L. Chang, "Suppression of Limit Cycles in Digital Filters Designed with One Truncation Quantizer," *IEEE Trans. Circuits Sys.*, vol. CAS-28, p. 107 (1981 Feb.).

[32] H. Samueli and A. N. Willson, Jr., "Nonperiodic Forced Overflow Oscillations in Digital Filters," *IEEE Trans. Circuits Sys.*, vol. CAS-30, p. 709 (1983 Oct.).

[33] T. Claasen and L. Kristiansson, "Necessary and Sufficient Conditions for the Absence of Overflow Phenomena in a Second-Order Recursive Digital Filter," *IEEE Trans. Acoust., Speech, Signal Proc.*, vol. ASSP-23, p. 509 (1975 Dec.).

[34] H. Samueli and A. N. Willson, Jr., "Almost Period P Sequences and the Analysis of Forced Overflow Oscillations in Digital Filters," *IEEE Trans. Circuits Sys.*, vol. CAS-29, p. 510 (1982 Aug.).

[35] P. K. Sim and K. K. Pang, "Effects of Input-Scaling on the Asymptotic Overflow-Stability Properties of Second-Order Recursive Digital Filters," *IEEE Trans. Circuits Sys.*, vol. CAS-32, p. 1008 (1985 Oct.).

[36] J. A. Moorer, "The Audio Signal Processor: The Next Step in Digital Audio," *Collected Papers*

from the AES Premiere Conference on Digital Audio (Rye, NY, 1982 June 3–6), pp. 205–215.

[37] B. Liu and T. Kaneko, "Error Analysis of Digital Filters Realized with Floating-Point Arithmetic," *Proc. IEEE*, vol. 57, p. 1735 (1969 Oct.).

[38] J. Vanderkooy and S. P. Lipshitz, "Resolution Below the Least Significant Bit in Digital Systems with Dither," *J. Audio Eng. Soc.*, vol. 32, pp. 106–113 (1984 Mar.).

APPENDIX 1
TMS320 CODE

A tested real-time TMS32010 source code follows. This code runs at 25.60576 MHz, ±0.01%, on a TMS320C10-25 for a 50.4-kHz maximum sample rate. There are two filters maximum, executable per sample having double-precision coefficients, single-precision truncation error cancellation, and overflow detection and saturation. This leaves a headroom of 15 instruction cycles out of a maximum of 127 to perform noncritical functions in a fragmented outer executive loop.

```
*************************************************************
****************** filtering ****************************
FILTER
      BIOZ    FILTER          * JOND - 1987

      IN      XNL,2

      IN      INDEX,3         * GET COEFFICIENT
      LAC     INDEX           *  FOR OUTER LOOP
      AND     MSK4
      SACL    INDEX
      LAR     AR0,INDEX
      IN      *,2

      OUT     YNR,2           * 14 INSTR.

********** LEFT CHANNEL ****************
* PERFORM RESIDUAL COEFFICIENT CALCULATIONS
      ZAC
      LT      XNM2L           * Q0
      MPY     EA2L            * Q14
      LTA     XNM1L
      MPY     EA1L
      LTA     XNL
      MPY     EA0L
      LTA     YNM2L
      MPY     EB2L
      LTA     YNM1L
      MPY     EB1L
      APAC                    * 26 INSTR.
*                             * WANT HEADROOM IN ACCUMULATOR
      SACH    RESID           * DIVIDE BY 2**12 => RESID IS Q10
* END RESIDUAL COEFFICIENT CALCULATIONS

      ZAC             *** PERFORM ERROR FEEDBACK CALCULATIONS
      LT      EONM2L          * STORED SUCH THAT IS NEGATIVE Q12
      MPY     B2L             * Q12
      LTD     EOFNL           * REALLY EONM1L
      MPY     B1L             * P REGISTER RESULT IS Q24
      APAC
      SACH    TEMP,4          * INTERMEDIATE ERROR TERM Q12


                      *** NOW DO SINGLE PRECISION COEFFICIENTS
      LAC     TEMP            * Q12    * 35 INSTR.
      LT      XNM2L           * Q0
      MPY     A2L             * Q12
      LTD     XNM1L
      MPY     A1L
      LTD     XNL
      MPY     A0L
      LTA     YNM2L
      MPY     B2L
      LTA     YNM1L
      MPY     B1L
      APAC                    * 46 INSTR.

              *** FORMULATE NEW TRUNCATION ERROR AND OUTPUT SAMPLE
      ADD     RESID,2         * Q12
      SACH    YNL,4           * YNL Q0, ACCUMULATOR Q12
      SACH    OVERFL          * STORE MSBs OF ACCUMULATOR
      SUB     YNL,12
      SACL    EOFNL           * Q12    * 51 INSTR.

*     ************** CHECK FOR OUTPUT OVERFLOW *****************
      LAC     OVERFL,1        * SIGN EXTENSION HERE.
      SACH    TEMP,4          * 5 MSBs OF Q12 ACC NOW ISOLATED.
      SACH    SIGN            * BEST BET AS TO OUTPUT SIGN.
      ZALS    TEMP            * NO SIGN EXTENSION HERE.
      XOR     SIGN            * ARE MSBs DIFFERENT?
      BZ      IIRL            * RESULT IS 0 IF ALL SAME.
      LAC     SIGN
      XOR     CEILIN          * MAGIC NUMBER => -32767 (NOT 8)
      SACL    YNL             * 61 INSTR., SATURATE
IIRL

********** RIGHT CHANNEL (in cascade with LEFT) *****************
* PERFORM RESIDUAL COEFFICIENT CALCULATIONS
      ZAC
      LT      YNM2L           * Q0
      MPY     EA2R            * Q14
      LTA     YNM1L
      MPY     EA1R
      LTA     YNL
      MPY     EA0R
      LTA     YNM2R
      MPY     EB2R
      LTA     YNR             * REALLY YNM1R
      MPY     EB1R
      APAC                    * 73 INSTR.
*                             * WANT HEADROOM IN ACCUMULATOR
      SACH    RESID           * DIVIDE BY 2**12 => RESID IS Q10
* END RESIDUAL COEFFICIENT CALCULATIONS

      ZAC             *** PERFORM ERROR FEEDBACK CALCULATIONS
      LT      EONM2R          * STORED SUCH THAT IS NEGATIVE Q12
      MPY     B2R             * Q12
      LTD     EOFNR           * REALLY EONM1R
      MPY     B1R             * P REGISTER RESULT IS Q24
      APAC
      SACH    TEMP,4          * INTERMEDIATE ERROR TERM Q12

                      *** NOW DO SINGLE PRECISION COEFFICIENTS
      LAC     TEMP            * Q12    * 82 INSTR.
      LT      YNM2L           * Q0
      MPY     A2R             * Q12
      LTD     YNM1L
      MPY     A1R
      LTD     YNL
      MPY     A0R
      LTA     YNM2R
      MPY     B2R
      LTD     YNR             * REALLY YNM1R
      MPY     B1R
      APAC                    * 93 INSTR.

              *** FORMULATE NEW TRUNCATION ERROR AND OUTPUT SAMPLE
      ADD     RESID,2         * Q12
      SACH    YNR,4           * YNR Q0, ACCUMULATOR Q12
      SACH    OVERFR          * STORE MSBs OF ACCUMULATOR
      SUB     YNR,12
      SACL    EOFNR           * Q12    * 98 INSTR.

*     ************** CHECK FOR OUTPUT OVERFLOW *****************
      LAC     OVERFR,1        * SIGN EXTENSION HERE.
      SACH    TEMP,4          * 5 MSBs OF Q12 ACC NOW ISOLATED.
      SACH    SIGN            * BEST BET AS TO OUTPUT SIGN.
      ZALS    TEMP            * NO SIGN EXTENSION HERE.
      XOR     SIGN            * ARE MSBs DIFFERENT?
      BZ      IIR             * RESULT IS 0 IF ALL SAME.
      LAC     SIGN
      XOR     CEILIN          * MAGIC NUMBER => -32767 (NOT 8).
      SACL    YNR             * 108 INSTR., SATURATE
*
********************** UNIVERSAL RETURN *********************
IIR
      RET                     * 110 INSTR. + 2 (call) => 15 LEFT/127
*                             *                       FOR OUTER LOOP
*************************************************************
*                  * AT 25.6MHz/4/50.4 KHz => 127 INSTR. MAX
```

APPENDIX 2
HOW TO DETECT OUTPUT OVERFLOW

We store the Q0 output $\hat{y}(n)$ from a 32-bit accumulator which is in Q12 format. The accumulator is Q12 because we chose to perform the single-precision (Q12) coefficient accumulation last. (See the TMS320 code in Appendix 1, right before "CHECK FOR OUTPUT OVERFLOW.") Since $\hat{y}(n)$ is a 16-bit Q0 number, we are left with 4 bits of visible headroom in the accumulator above the sign bit of $\hat{y}(n)$. If no output overflow has occurred, then these "headroom" bits should each be a copy of the sign bit of $\hat{y}(n)$. This suggests a simple way to detect output overflow, then. If any of the 5 MSBs in the accumulator differ from the others at the time that $\hat{y}(n)$ is stored, then output overflow must have occurred. Since we really do not have a clue as to the extent of the overflow, our best guess is that the proper sign of the output should be the same as the MSB of the accumulator. Using that information, we should saturate the output under program control. The 4 headroom bits will allow us to detect output overflow up to 24 dB past unity.

The TMS320 code to perform output overflow detection/saturation is shown in the program. Notice that the use of the XOR operating with the magic number CEILIN (= 32766) on the sign of the Q12 accumulator obviates the need for conditional branching to saturate

at the proper polarity. This detection/saturation routine takes 10 TMS320 instruction cycles per filter stage. This is a significant portion of the code and is almost as long as the basic IIR filter calculation itself. Unfortunately this code is inextricable for high-quality audio work. The ADSP-2100, in contrast, has 8 hard overflow bits annexing its 32-bit accumulator. It has a single instruction monitoring the sameness of the 9 MSBs. We should expect quite a code reduction using it.

THE AUTHOR

Jon Dattorro was born in Providence, Rhode Island, in 1954. He studied piano from the age of eight with Salvatore Fransosi. From 1972 to 1977 he attended the New England Conservatory of Music where he studied piano and electronic music. While in Boston he studied composition with Pozzi Escot and was a soloist with the Boston Symphony Orchestra for a series of Children's Concerts. He holds a B.S.E.E. degree with honors from the University of Rhode Island (1981) and an M.S.E.E. with emphasis on digital signal processing from Purdue University (1984).

Mr. Dattorro was with Lexicon Inc. until 1987, where he was the principal engineer for the Model 2400, their premier time-compression device. His responsibilities also included the digital filter design for OPUS, their recently announced digital audio workstation.

Mr. Dattorro is now employed by Ensoniq Corp. in Malvern, Pennsylvania, where he has been the project engineer for the Model SQ-80 synthesizer, and where he is currently working on new product research and development. He is a member of the AES and the IEEE.