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:

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

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,

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

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 $\theta=\pi$ 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,

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

where,

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

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

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

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 $\pi/3$ 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$(ec_i)/(2^{qc}2^{qe}) \cong c_{Fi}$
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 LSBwould 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:

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

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,

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

;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

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

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

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

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,

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

The calculation we will actually perform on chip is,

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

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 ofg/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,

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

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

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,

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

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:

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

$$y_{des}(n) = \sum_{p=0}^{31} y_p(q) \quad ; n \leq> 48\text{kHz} \quad ; q \leq> 1.5\text{ kHz} \qquad \text{(II-12)}$$

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

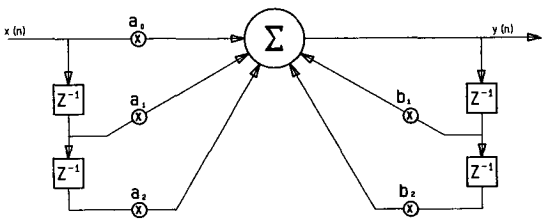

Fig. 1. For audio use. Direct Form I.

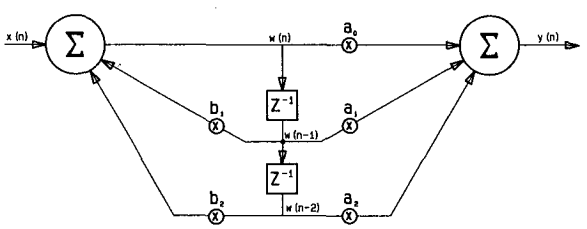

Fig. 2. Not for audio use. Direct Form II-Canonic.

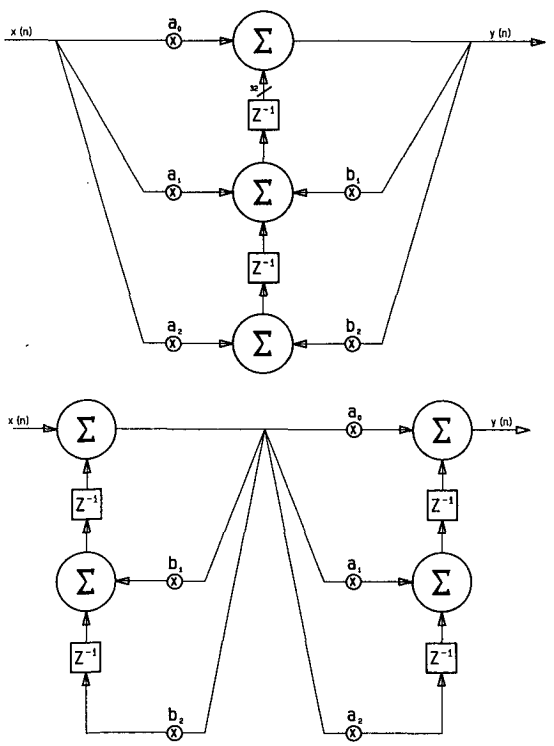

Fig. 3. a) OK for audio use. Direct Form II Transpose. b) Not for audio use. Direct Form I Transpose.

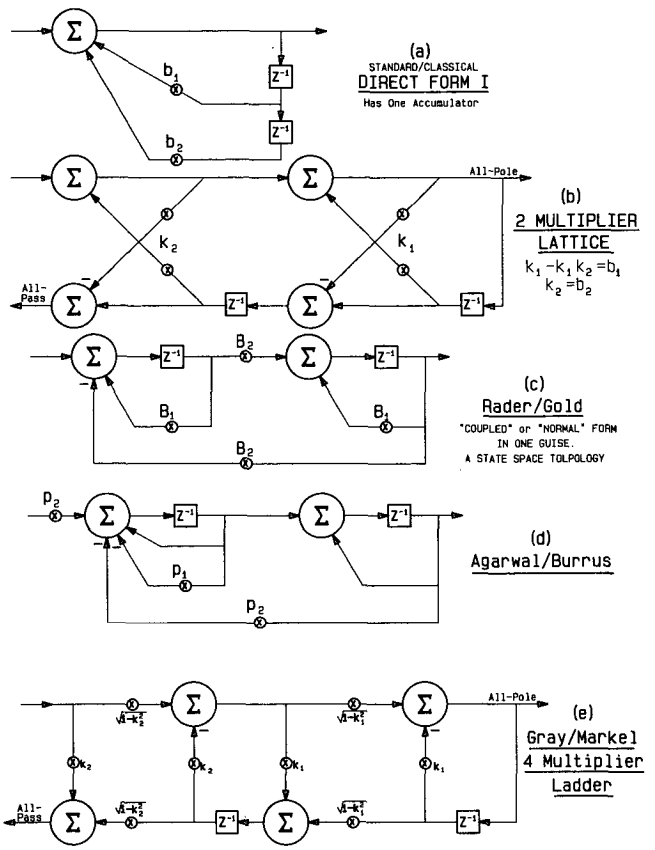

Fig. 4. Some second order topologies.

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.

EXAMPLE:

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

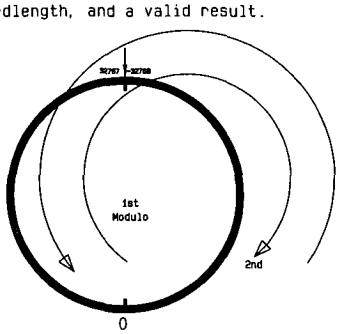

Fig. 5. Two's complement is a modulo arithmetic. The first modulo (ring) for 16 bits is shown. The arrows help visualize the traversal from the first into the second modulo, and then back again.

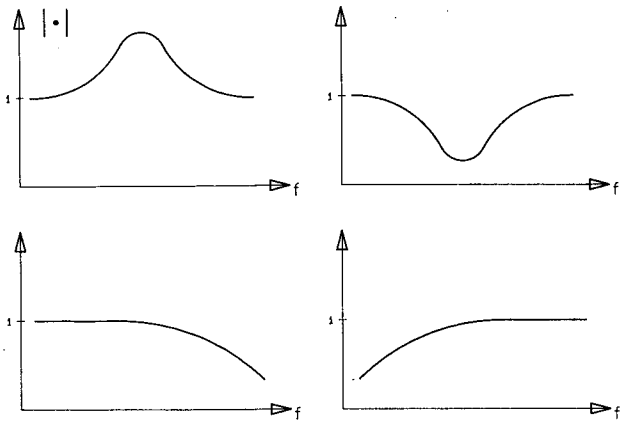

Fig. 6. Unity gain design.

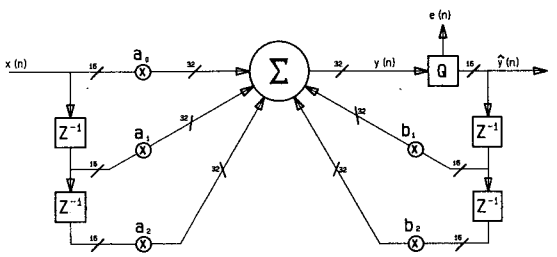

Fig. 8. Direct Form I having truncator.

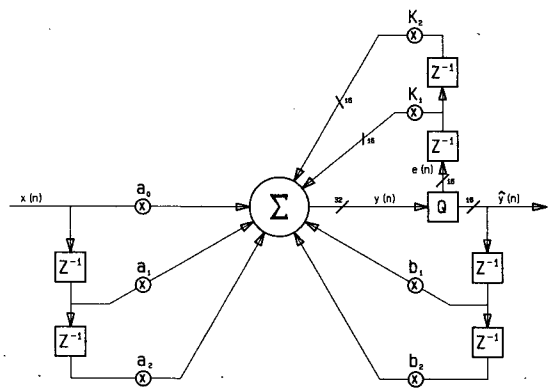

Fig. 11. Second order error feedback.

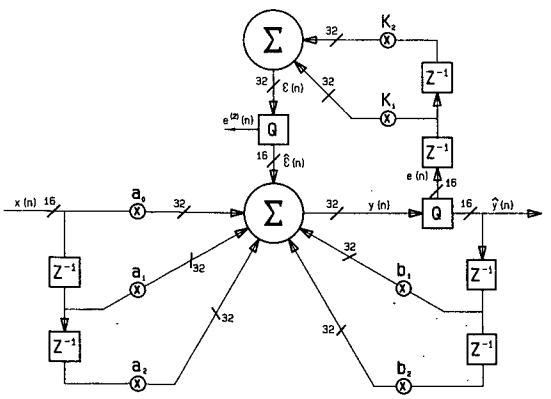

Fig. 11-II. Second order truncation error cancellation showing all truncation errors.

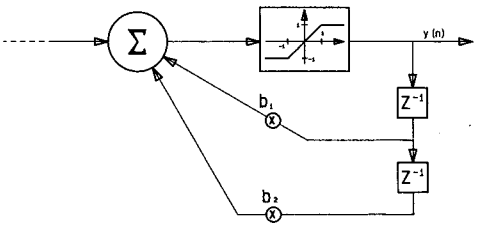

Fig. 15. Showing the saturator explicitly, forced overflow analysis.

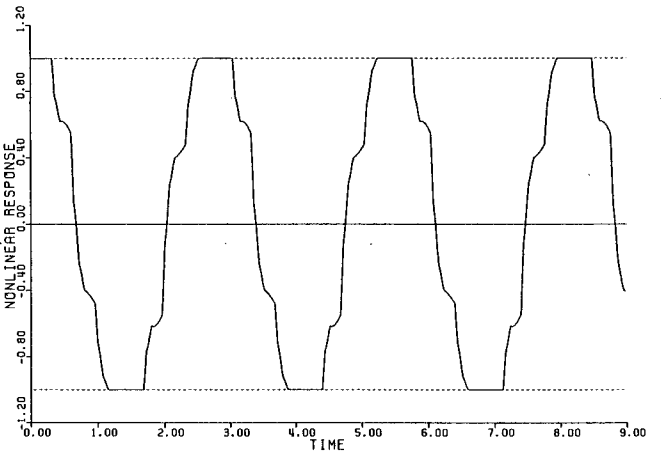

Fig. 16. Typical forced overflow response for a sinusoidal input.

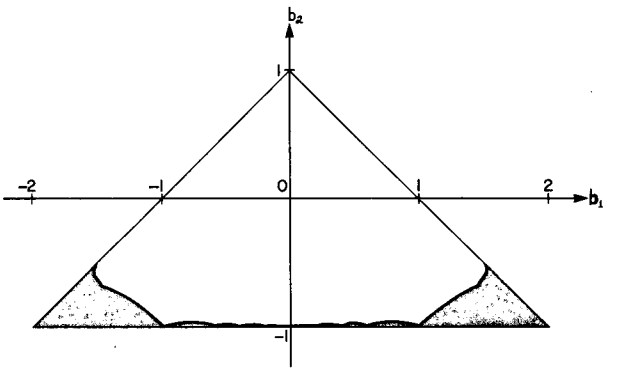

Fig. 17. Stability triangle for positive topology.

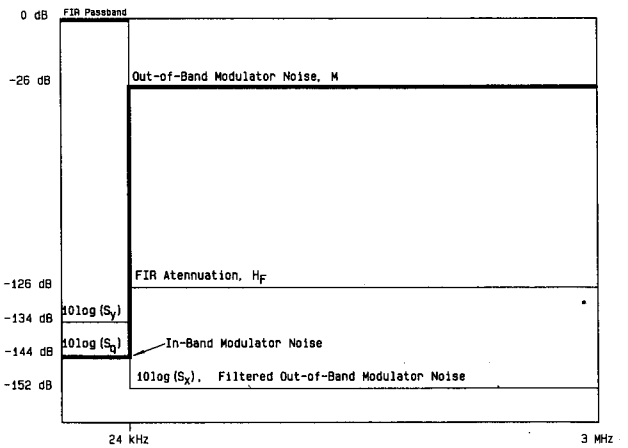

Fig. 19. Relationship of M, S_Q, and H_F to achieve S_y, the in-band spectral density after decimation.

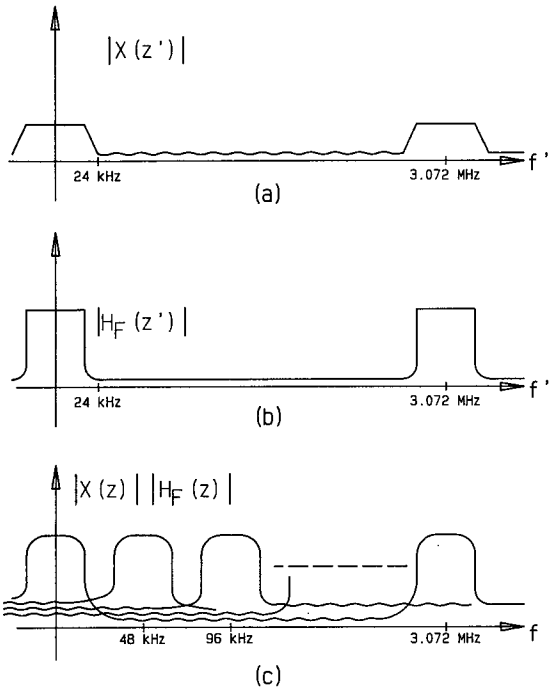

Fig. 18. Decimation in the frequency domain.

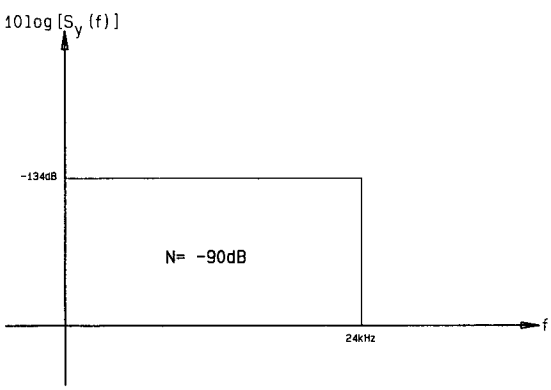

Fig. 20. Required power spectral density of noise power, N, to achieve 90 dB S/N.

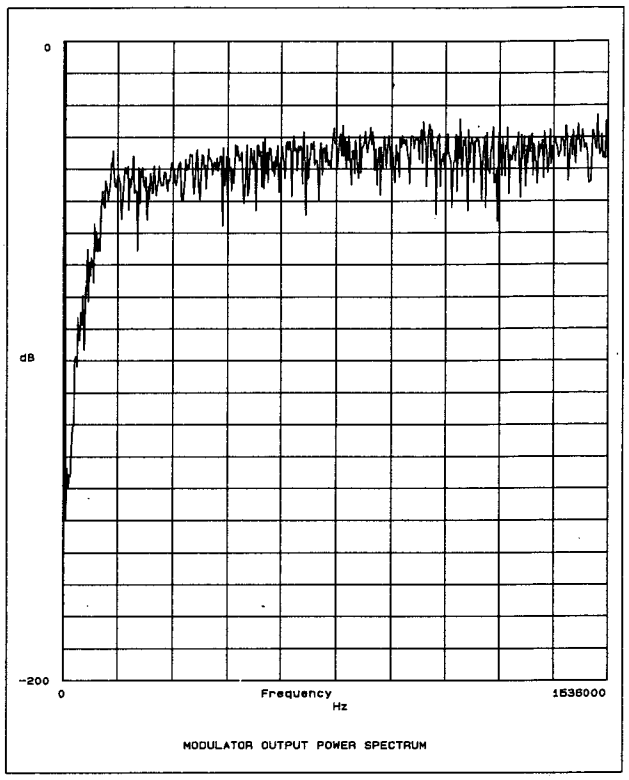

Fig. 21.

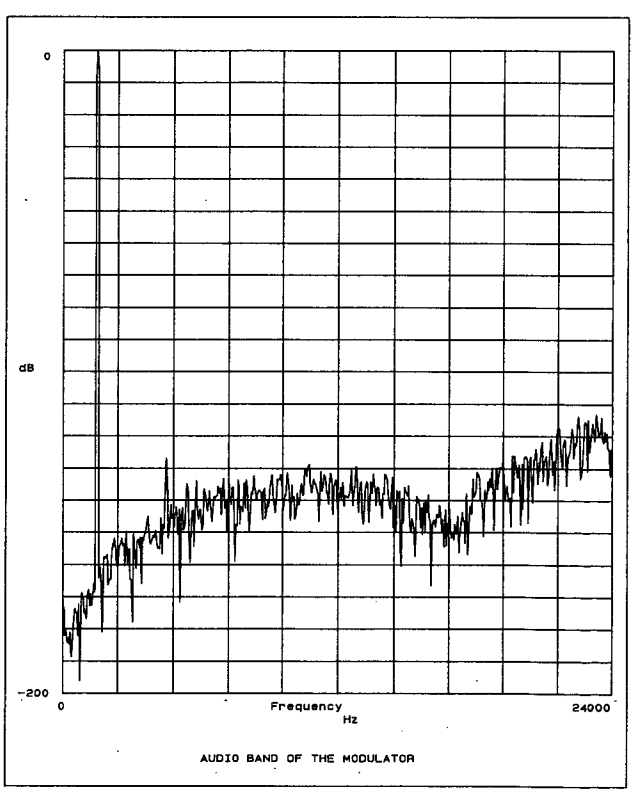

Fig. 22.

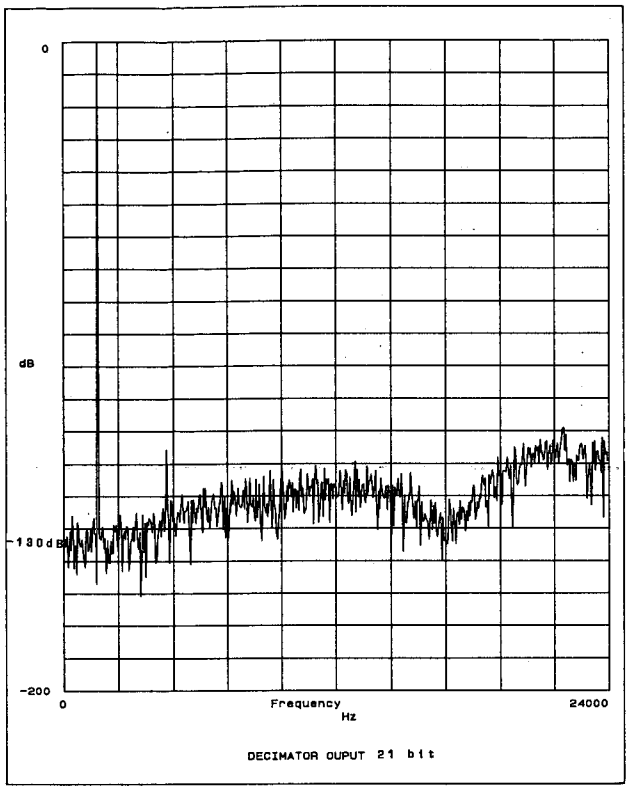

Fig. 23A.

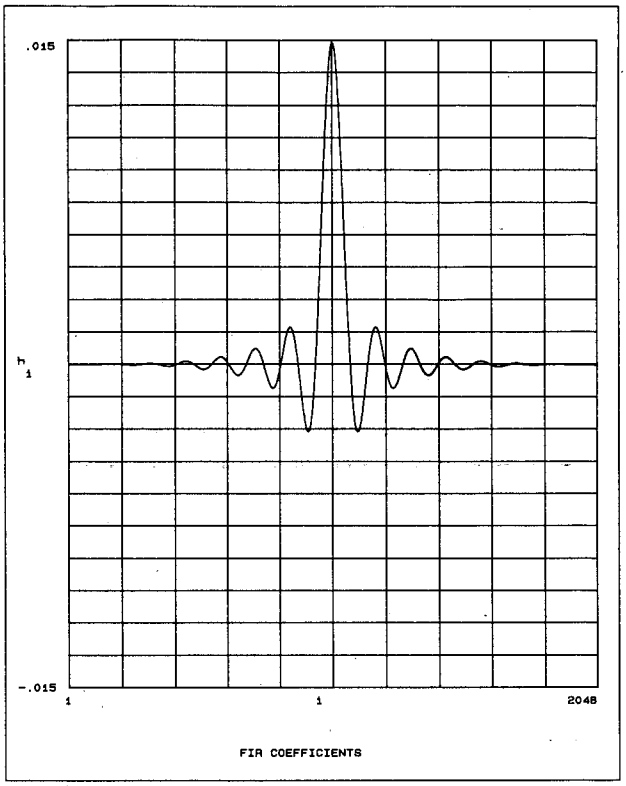

Fig. 24.

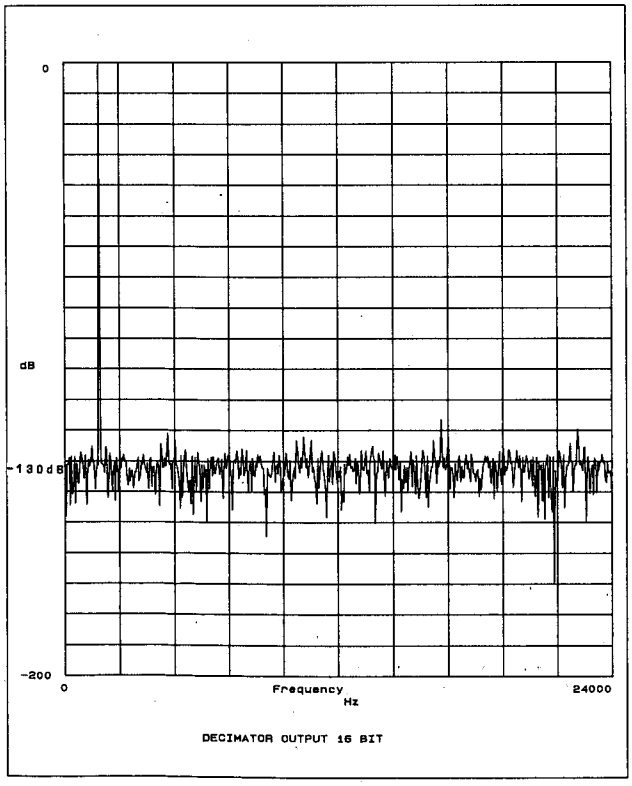

Fig. 23B.

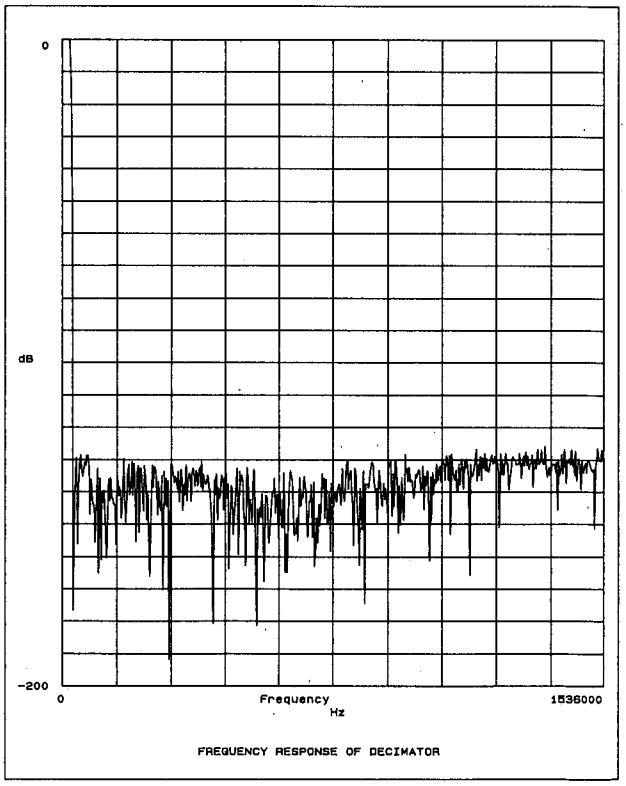

Fig. 25.

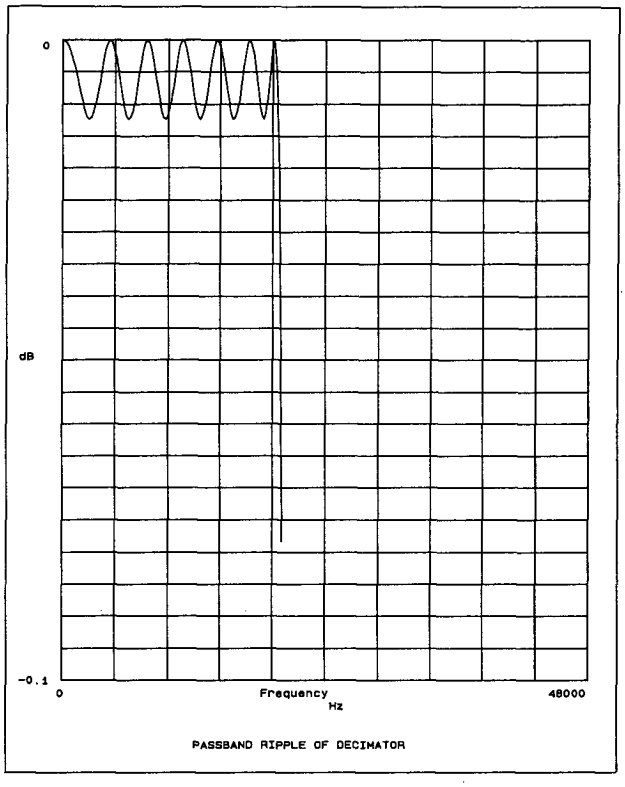

Fig. 26.

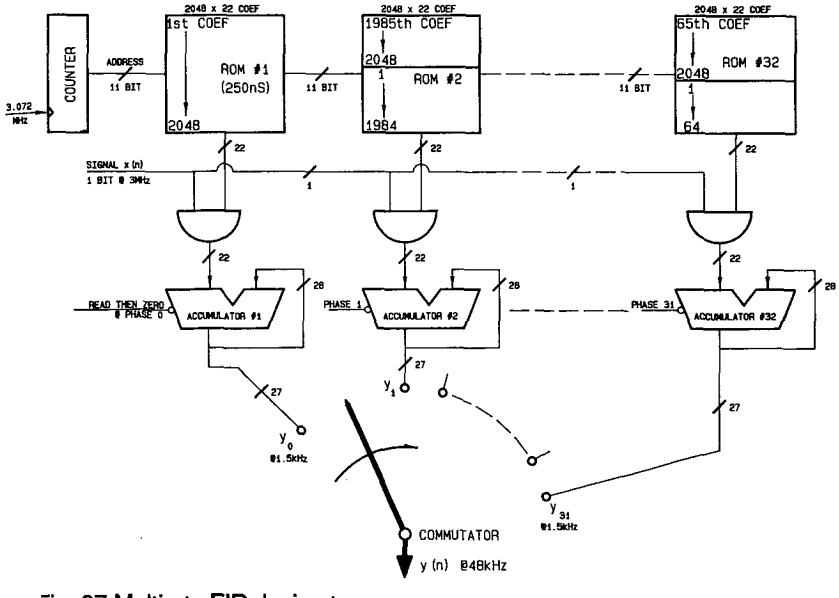

Fig. 27.Multirate FIR decimator.

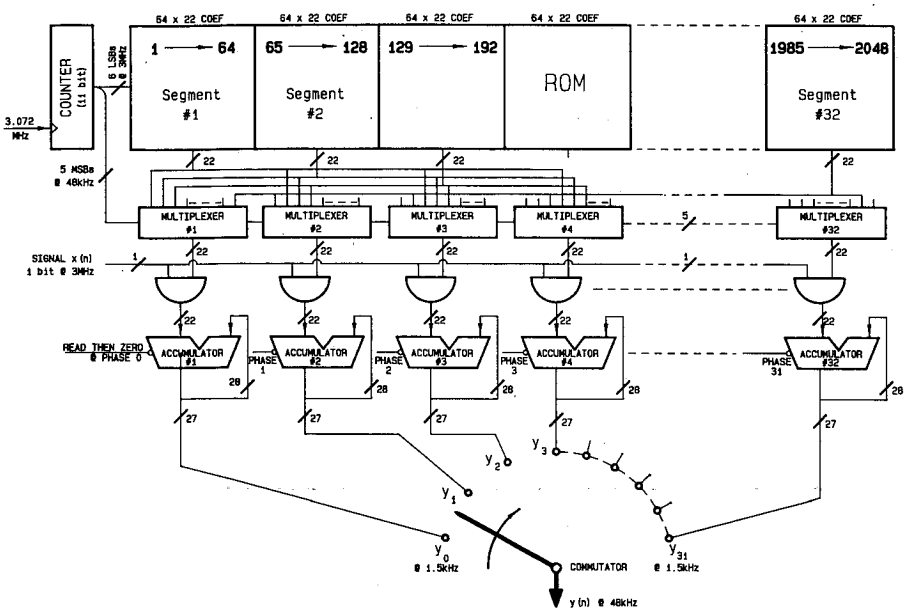

Fig. 28.ROM collapse.

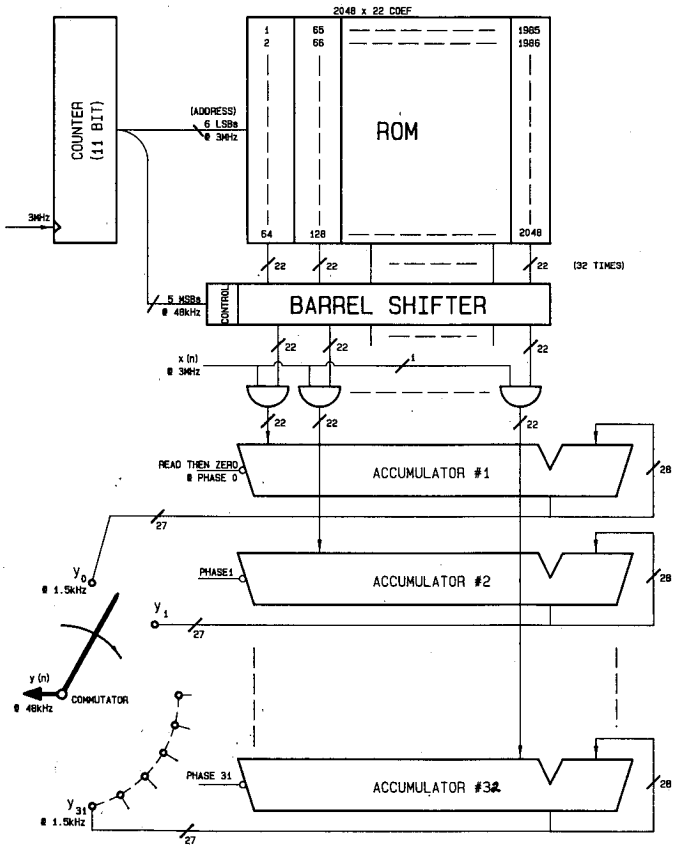

Fig. 29.Physical organization — MUX collapse.