Interpolation filters in delta-sigma DACs

Ivar Løkken, Jan. 2006

1 Motivation and specification

In a delta-sigma digital-to-analog converter, the input data is oversampled to a high-rate representation before being bit-reduced using a noise-shaped quantizer. This allows for the DAC to have few levels to resolve, but at the same time maintaining high resolution in the baseband. The oversampling ratio in delta-sigma audio DACs is typically 64 to 256.

Oversampling in principle consists of two processes. The first is to increase the sampling rate by inserting zeros between the existing ones, called zero-stuffing. Then, the signal is band limited using a digital low-pass filter. These are combined using an interpolation filter. The process is shown in the time and frequency-domain for a 2x oversampling in figure 1.

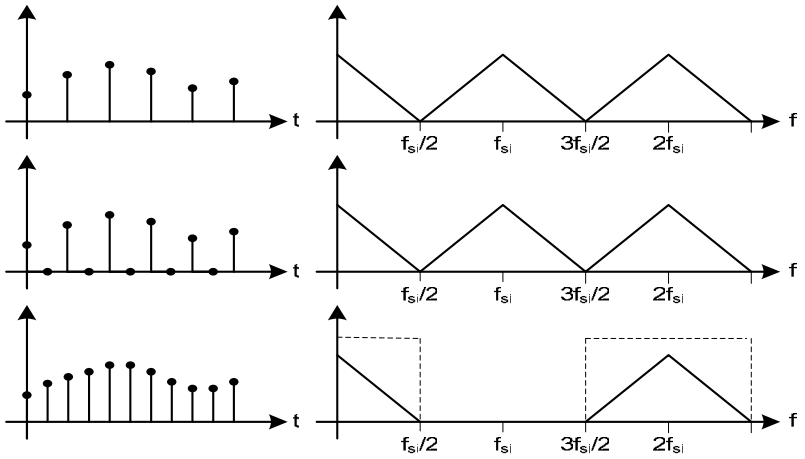

Figure 1: Oversampling in the time and frequency domain

Ideally, the output signal would have exactly the same spectral content as the input signal in its Nyquist-band and consequently the ideal interpolation-filter is characterised by:

$$H_L(f) = \begin{cases} 1 & , \ 0 < f < \dfrac{f_{si}}{2} \\ 0 & , \ \dfrac{f_{si}}{2} < f < \dfrac{Lf_{si}}{2} \end{cases}$$

The impulse response is given by the inverse fourier transform of the frequency characteristic and is hence:

$$h_L[n] = \mathrm{sinc}\left[\mathrm{n}\frac{1}{\mathrm{L}}\right] \qquad n \in \langle -\infty, \infty \rangle$$

An infinite impulse-response is of course not realizable, and real filters are designed based on the tolerated deviation from this ideal characteristic. Specified deviations include passband-ripple, stopband-attenuation and transition-band length (i.e. slope of the filter).

However, unlike for ADCs, where the high-sample rate signal is decimated and the lowpass-filter requirements, specifically the stopband attenuation, are strictly defined by the allowed

aliasing-error, a typical DAC application has no subsequent sampling of the signal after the interpolator, and the replicated spectra from the zero-stuffing will not fold into the baseband. Thus, the low-pass filter could theoretically be omitted. This would however have negative implications for the overall performance, as the large content of high-frequency energy would saturate the delta-sigma modulator, make the discrete-time to continuous-time interface extremely jitter-sensitive and cause the analog circuitry to slew. Non-linearity would also result in modulation products in the baseband.

Still, the requirement for low-pass filtering in an oversampled DAC is much more ambiguous than that of an ADC. How much high-frequency content the delta-sigma modulator tolerates for instance, is dependent on the modulator design. At its output, the DSM will also introduce much high-frequency content of its own, so in the case of the discrete-to-continuous time interface and subsequent analog filtering, the images will be of no influence if they are attenuated to a level where they are well below the DSM high-frequency noise-floor.

In typical high-end audio applications, the transition band is usually specified from the CD-Audio sample-rate of 44.1kHz. The passband is defined up to the presumably audible limit of 20kHz and the stopband usually specified from 24.1kHz. This gives a 4.1kHz transition band that is symmetric around half the input sample rate. If the same basic interpolation-filter is used for all input sample rates, the transition band will be given by table 1.

Table 1: Interpolator transition-band specification, based on CD-audio

Sampling rate	**Passband**	**Transition-band**	**Stopband**
44.1kHz	0Hz→20kHz	20kHz→24.1kHz	24.1kHz→∞
48kHz	0Hz→21.8kHz	21.8kHz→26.2kHz	26.2kHz→∞
96kHz	0Hz→43.5kHz	43.5kHz→52.5kHz	52.5kHz→∞
192kHz	0Hz→87.1kHz	87.1kHz→104.9kHz	104.9kHz→∞

It should be noted however, that the transition-band is sometimes increased for higher sampling-rates, allowing lower-order filters to be used. It should also be noticed that there is usually not a uniform stopband attenuation requirement all the way to infinity. At high frequencies, the delta-sigma modulator has a high noise-floor and the damping from the analog output filter will also be significant. This reduces the requirement for image suppression at frequencies much higher than the passband and can be exploited to make more efficient filters, which will be shown later.

As mentioned, the requirements for the filter in a DAC are quite diffuse. The passband-deviation should be well within audible limits, while the stopband attenuation should be high enough for the high-frequency content not to affect the performance of the rest of the converter. It is in practice the designer's choice. A survey on existing high-end converters showed typical passband deviation to be in the range 0.0001dB to 0.001dB, while stopband-attenuation is found to be ranging from 75dB to 120dB [1], [2], [3].

Other interesting design parameters for audio interpolation filters include phase response and transient response. Due to the desire for a constant group delay, linear phase symmetrical FIR-filters are almost exclusively used in audio interpolators. However, it has been suggested from audiophiles and professionals that the pre-ringing, due to the impulse response being symmetric around the impulse, might smear the auditory arrival-time detection and thus negatively impact stereo imaging [4, 5, 6]. Unfortunately, a well-documented psychoacoustic evaluation has yet to be published.

2 Implementation of interpolation filters for audio DACs

2.1 Interpolator filter partitioning

We have established that most interpolation filters for audio DACs are linear-phase FIR interpolators with a certain transition-band, a certain passband-ripple and a certain stopband-damping, with the reservation that the latter can be relaxed for parts of the stopband. To approximate the ideal brickwall characteristic, the filter must be very long. Such high-order FIR-filters are usually designed with synthesis software using windowing, the Parks-McClellan method or another algorithm to determine the filter coefficients.

Figure 2 show a filter for 128 times oversampling with a transition band given by table 1, 0.005dB passband-ripple and 75dB stopband-damping, - a realistic specification for audio DAC oversampling. As we can see, due to the high sampling rate, the passband and transition band are extremely narrow compared to the full output nyquist bandwidth. This means the filter must be of exceedingly high order.

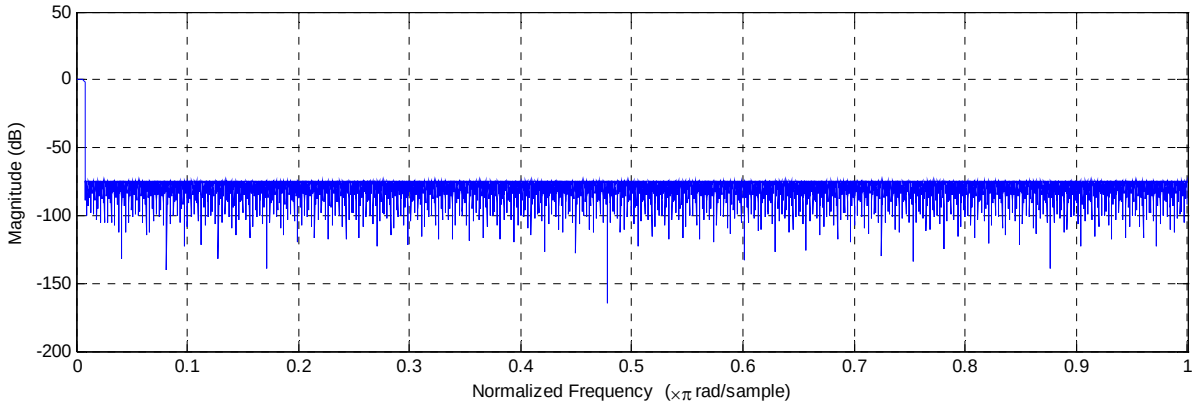

Figure 2: Frequency response, 128x OSR lowpass-filter.

The synthesized filter has an order of 5677. With so many coefficients, the filter will require much memory and many multiplications per sample, meaning the circuit will be very large and inefficient. Fortunately, there are methods to decrease the complexity significantly.

One way to reduce the number of computations is to use several filters in cascade. This is illustrated in figure 3 with 8x OSR lowpass-filtering being performed using three 2x OSR filters in cascade. As can be seen, the transition-band can be increased for each subsequent filter, as the aliases from the previous filtering are spaced further and further apart.

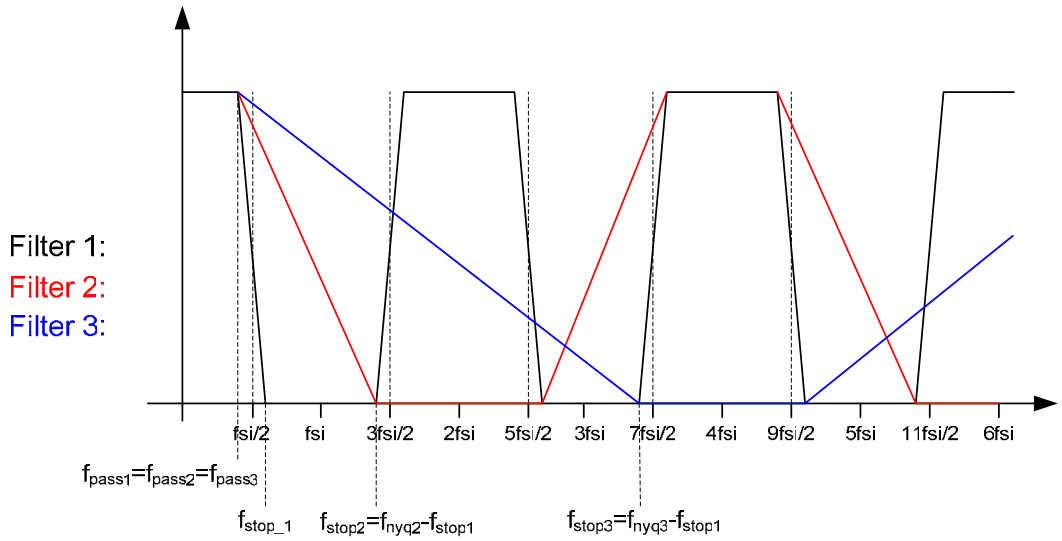

Figure 3: Interpolation with cascade of lower OSR filters

If the 128x OSR filter was implemented with a cascade of seven 2x OSR-filters, the order of these, synthesized for the same requirements as in figure 2, would be as given by table 2.

Table 2: Required filter order with seven cascade 2x OSR filters for 128x OSR

Filter	Filter1	Filter2	Filter3	Filter4	Filter5	Filter6	Filter7
Order	89	15	7	5	5	3	3

As we can see, the total number of coefficients has been reduced from 5677 for the direct implementation to 126 for the cascaded implementation. This is much more feasible to implement in CMOS. Still, the requirement for number of multiplications is still large. In the first example, only every 128th input sample would be non-zero, meaning that the filter must have performed 5677 multiplications per input sample or 45 per output sample. In the cascaded case, only every second sample in each filter is non-zero, and the rate doubles for each successive filter. The number of multiplications per input sample then becomes 569. A significant saving, but still the filter is computationally heavy.

Another much used method to further relax the filter requirements is to use a lower oversampling FIR-filter followed by a simple hold or linear interpolator as seen in figure 4.

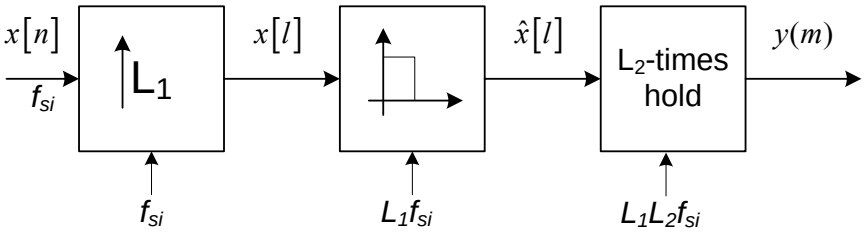

Figure 4: Lower OSR FIR-filter followed by hold-circuit

The hold-operation is characterised by a sinc frequency-response. This means it will have zeros at multiples of $L_1 f_{si}$ or where the replicated spectra of the first filter are centered.

$$y(m) = \hat{x}[l] * rect\left(\frac{1}{L_1 \cdot f_{si}}\right)$$

$$Y(f) = \hat{X}(f) \cdot \mathrm{sinc}\left(\frac{f}{L_1 \cdot f_{si}}\right)$$

Figure 5 illustrates a 128x OSR filter constructed from a 8x FIR interpolator and a 16x hold-circuit in the frequency domain. It shows the characteristic of the 8xFIR interpolator (red trace) from f_{s_in} to $64f_{s\_in}$ (the output nyquist frequency) along with the frequency response of the 16x hold interpolator (blue trace).

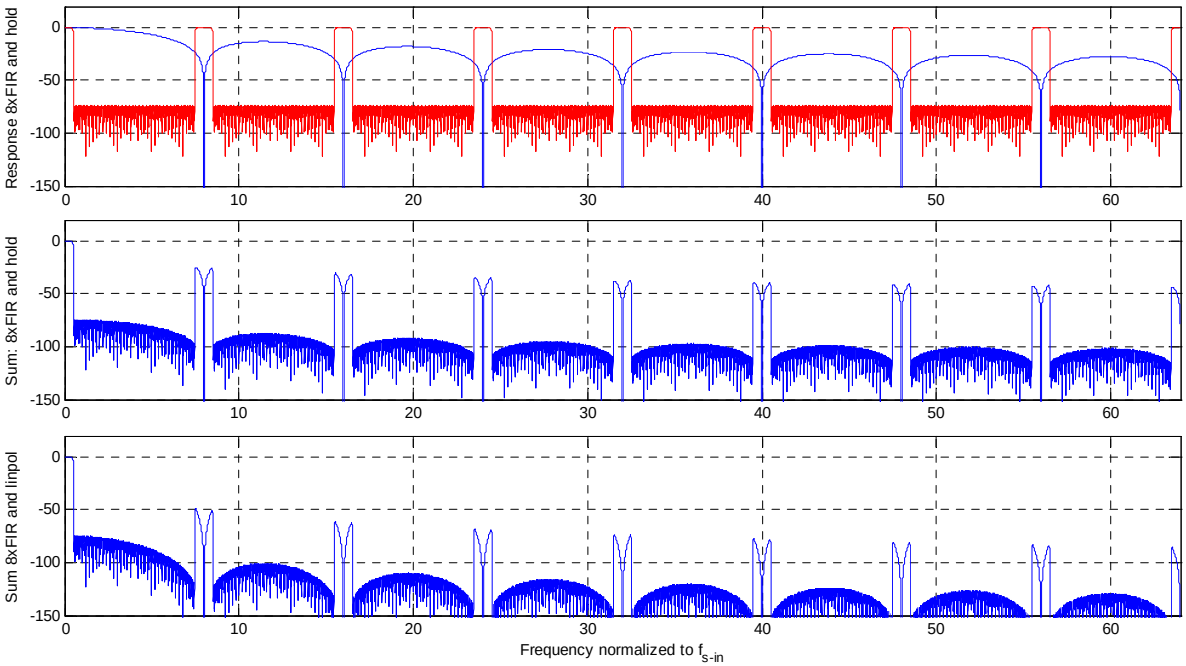

Figure 5: Response for 8xFIR and 16x hold (top), summed response (middle) and same with linear interpolation instead of hold (bottom)

We can see that the sinc-function has very narrow zeros and that the summed filter response peaks at multiples of eight times the input sample frequency. However, this is far into the stopband, and as remembered from the reasoning in section one, these peaks can be tolerated in many applications. If the FIR interpolator was 16x, the peaks would be spaced even further apart. If so high peaks in stopband-damping are not tolerable, the hold-interpolator could be replaced with a linear interpolator, which is also computationally very simple. The linear interpolator has a sinc^2-response. Then, the highest peak is 50dB down, and likely to be well below the noise-floor of the delta-sigma modulator. It should be noted that both the hold and linear interpolator has a slight roll-off in the passband, which is often compensated by the final stage FIR-filter [7, p223].

We have in this chapter shown that by partitioning of the interpolation-filter, the computational load can be significantly reduced. The relaxed requirements far into the stopband can also be exploited for simplification. Next, we will look at structures for realising these filters.

2.2 Interpolator filter structures

A FIR digital filter is described with its finite length impulse response h[n]. Given that the filter is causal, we have in the z-domain and frequency domain for a filter of length N:

$$H(z) = \sum_{n=0}^{N-1} h[n] \cdot z^{-n}$$

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h[n] \cdot e^{-j\omega n}$$

The filter is applied to the input by convoluting the impulse responses, which correspond to multiplying in the frequency- or z-domain:

$$y[n] = x[n] * h[n] = \sum_{k=0}^{N-1} h_L[n-k] \cdot x[k]$$

$$Y(z) = X(z) \cdot H(z)$$

$$Y(e^{j\omega}) = X(e^{j\omega}) \cdot H(e^{j\omega})$$

The simplest form of implementation is to directly realize the non-recursive difference equation that is the convolution sum. This is called the direct-form structure and is shown in figure 6. The zero-stuffing is added at the beginning indicating the filter being part of an interpolator with oversampling factor L:

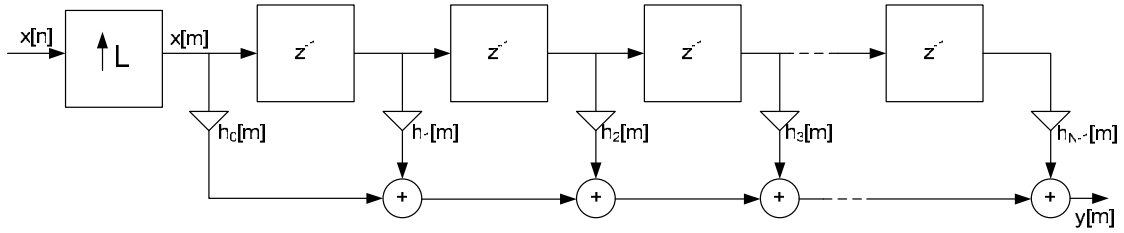

Figure 6: Direct form implementation of FIR filter.

However, the direct-form implementation is disadvantaged in terms of efficiency when used in interpolators. It requires N multiplications per output sample y[m], even though we know the input signal is zero-stuffed. For L times oversampling, only every L^{th} input value is nonzero. Then it is evident that implementing every factor h[m] in the difference equation leads to many redundant multiplication operations.

The fact that the input signal consists mostly of zeros can be exploited using the polyphase decomposition. To illustrate this principle, we look at a three-branch polyphase decomposition of an impulse response of length 9:

$$H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[4]z^{-4} + h[5]z^{-5} + h[6]z^{-6} + h[7]z^{-7} + h[8]z^{-8}$$

If we regroup the equation, we get:

$$H(z) = \left(h[0] + h[3]z^{-3} + h[6]z^{-6}\right) + \left(h[1]z^{-1} + h[4]z^{-4} + h[7]z^{-7}\right) + \left(h[2]z^{-2} + h[5]z^{-5} + h[8]z^{-8}\right)$$

This can be rewritten as:

$$H(z) = \left(h[0] + h[3]z^{-3} + h[6]z^{-6}\right) + z^{-1}\left(h[1] + h[4]z^{-3} + h[7]z^{-6}\right) + z^{-2}\left(h[2] + h[5]z^{-3} + h[8]z^{-6}\right)$$

Naming the coefficient in the first parenthesis E_0, the next E_1 and the last E_2, we see that these three functions are only nonzero every third sample instant, and we can define H(z) as:

$$H(z) = E_0\left(z^3\right) + z^{-1}E_1\left(z^3\right) + z^{-2}E_2\left(z^3\right)$$

Generally, the L-branch polyphase decomposition can be defined by:

$$H(z) = \sum_{m=0}^{L-1} z^m E_m\left(z^L\right)$$

The general flowchart of the L-branch polyphase filter is shown in figure 7. The transposed version is seen to the right, and can easily be shown to be completely equivalent. Initially it's difficult to see the advantage of dividing the filter this way, but we will soon see that it is very efficient when realizing interpolators.

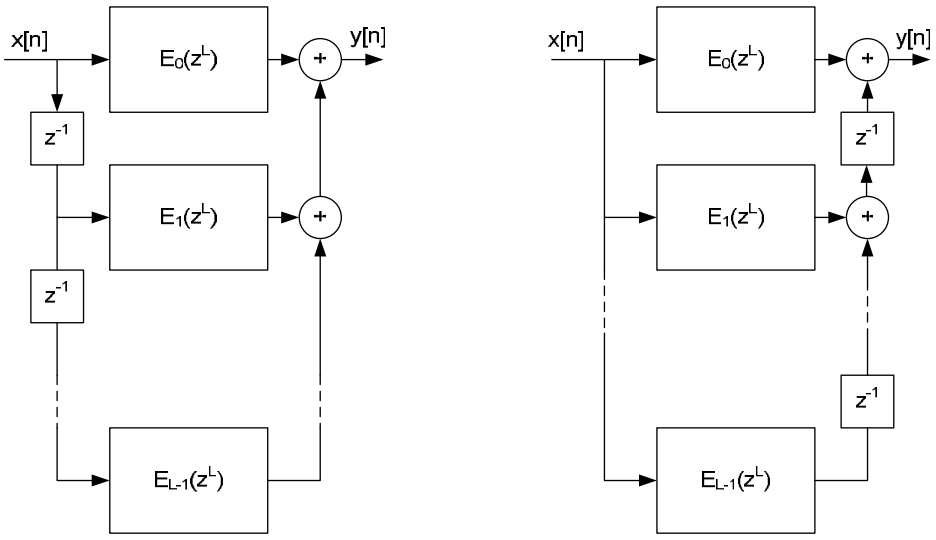

Figure 7: L-branch polyphase FIR-filter and its transpose

To understand how the polyphase decomposition can be used to implement very efficient interpolators, first assume we apply an L-branch polyphase filter applied to an L times

oversampled signal. Due to the zero-stuffing, only every L^{th} input sample is non-zero. This means that the signal flow through each branch is on the form:

$$Y_m(z) = X(z^L) \cdot E_m(z^L) \quad , f_s = Lf_{si}$$

This means we can run the branch filters at the input sampling rate, replacing each z^{-3} delay with a z^{-1} delay

$$Y_m(z) = X(z) \cdot E_m(z) \quad , f_s = f_{si}$$

Now the zero-stuffing can be removed and each branch filter E_m only has to do one computation for every non-stuffed input sample. The multiplication redundancy has been eliminated. Since each branch filter only produces a valid output every L^{th} output sample, the delay-chain at the output can be removed too, replaced with a rotating subfilter selector operating at full speed. This is illustrated in figure 8.

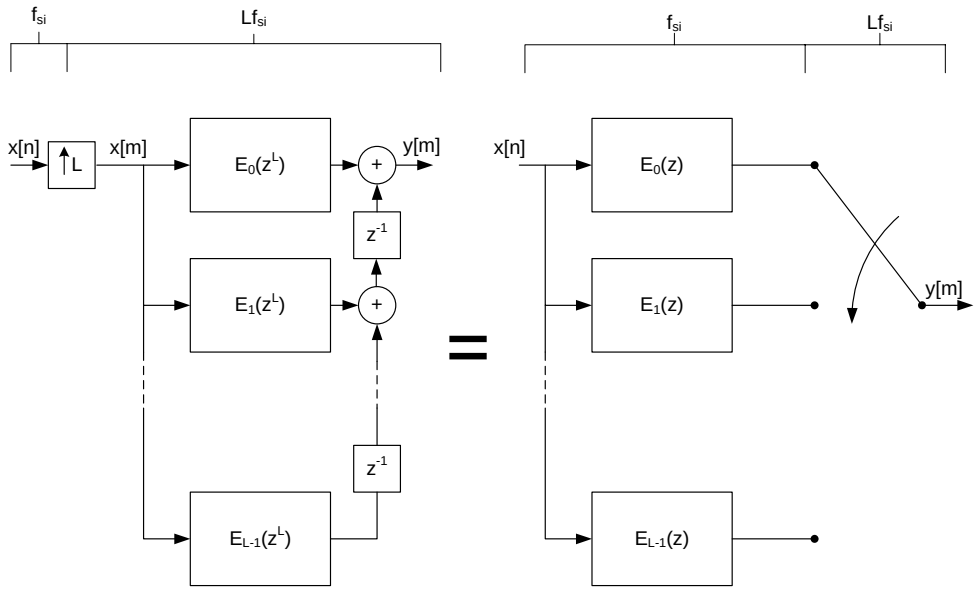

Figure 8: Efficient realization of polyphase interpolator

For an interpolator with oversampling rate L, the polyphase decomposition leads to an L times reduction in the number of computations per second, eliminating the computational redundancy of the direct form implementation when applied to a zero-stuffed signal. The number of coefficients is the same as before, given by the filter specification and design as described in chapters 1 and 2.1.

We have already shown that the number of coefficients can be reduced significantly by partitioning a large OSR filter into a cascade of subfilters with lower OSR. In the case of using 2x subfilters, the number can be reduced further using the half-band filter design.

Half-band-filters is a sub-class of linear-phase FIR filters characterised by a symmetric frequency response, i.e. that the transition band is at exactly half the nyquist frequency and ripple is equal in the passband and stopband. An ideal half-band brickwall filter is characterised by the impulse response:

$$h_{1/2}[n] = \mathrm{sinc}\left[\mathrm{n}\frac{1}{2} \right] \qquad n \in \langle -\infty, \infty \rangle$$

This means that every even coefficient, except $h_2[0]$, is zero. The observant reader will note from comparing with eq.2, that this is also the ideal filter for a 2x interpolator. Implemented half-band filters are of course not ideal, and synthesized based on the same requirements as regular filters, but the transition band always centres as half the nyquist-frequency and the even coefficients are always zero. This means the filter has only about half the number of computations compared to an arbitrary filter of the same length. Half-band filter transfer functions can generally be classified by:

$$H_{1/2}(z) = cz^{-K} + H_0(z^2).$$

We have now summarized how efficient structures can be used to reduce the complexity of interpolators. A cascade of 2x interpolators gives a significantly lower number of coefficients than a direct implementation of the entire oversampling ratio. If these interpolators are designed using the half-band subclass of filters, the number of coefficients can be reduced further by a factor of two. Additionally, by exploiting the polyphase structure, the number of computations per input sample can be reduced by a factor L for an L-times oversampling filter. If we compare the direct implementation from figure 2, with a cascade of half-band 2x interpolators exploiting the half-band structure, we get a significant complexity reduction, as given from table 3.

Table 3: Storage and computation requirements for direct vs optimized implementation

Filter topology	**Total number of coefficients**	**Multiplications per input sample**
Direct implementation	5677	726656
Polyphase, halfband 2x cascade	71	443

As can be seen, the savings in computational load and storage requirements are very significant. Also, since the filters are symmetric, two and two taps, at equal distance from the centre tap, can share the same coefficient. This means the numbers of stored coefficients can be halved once more.

2.3 Coefficient quantization

We have now shown how to do efficient partitioning of the interpolator and efficient structures for realization of the interpolation filters. However, in a real implementation, the design procedure must also include determining the precision of the filter coefficients. Infinite-precision filters would require infinite wordlength coefficients, which are clearly not realizable. Even with a given structural complexity, the resource usage of a filter may vary greatly, depending on the wordlenghts used in the computations. What number of bits to quantize the coefficients to is thus a trade-off between the required accuracy and the available chip area or DSP resources. Also, it might be necessary to quantize some internal variables in the filter, to avoid the wordlengths from growing and growing throughout the structure. How the filters respond to internal signal quantization is greatly dependent on the chosen filter structure.

The usual way to quantize filter coefficients is to truncate each one to a specific number of bits. As the coefficients are multiplied with the data, their wordlenght is also determined by the bit budget for the internal data paths.

2.3.1 Fixed-point vs. floating-point arithmetic

A critical choice in the quantization process is if the data should be implemented in fixed-point or floating-point arithmetic. Floating-point gives better results, but is more computationally heavy.

Fixed-point quantization is done using either truncating, disregarding the LSBs exceeding the wordlength, or rounding. In the case of truncation, the error for a B-bit number in two's complement form relative to the full scale is given by:

$$0 \le \varepsilon < 2^{-B}$$

This means the error is offset with a non-zero expectance value. In the case of rounding, the error is given by:

$$-2^{-B/2} \le \varepsilon < 2^{-B/2}$$

In the case of floating-point numbers, only the mantissa is quantized. The relative error in terms of the numerical values of a quantized floating-point number $Q(x) = 2^E Q(M)$ and the unquantized number $x = 2^E M$ is given by:

$$\varepsilon = \frac{Q(x) - x}{x} = \frac{Q(M) - M}{M}$$

E is the exponent and M is the mantissa. The precision is in other words better for small values where the mantissa is large and the exponent small.

An alternative to the computational burden of floating-point arithmetic is to use a hybrid called block floating-point arithmetic [8]. This is based on the recognition that coefficients near the centre tap are large compared to coefficients far from it. Thus, one can divide the filter into a number of segments and use the same exponent within these segments. This basically corresponds to a fixed-point representation within each segment and allows for the usage of fixed-point multipliers. The only extra computational burden is a shift-operation on each block as they have to be aligned at the output.

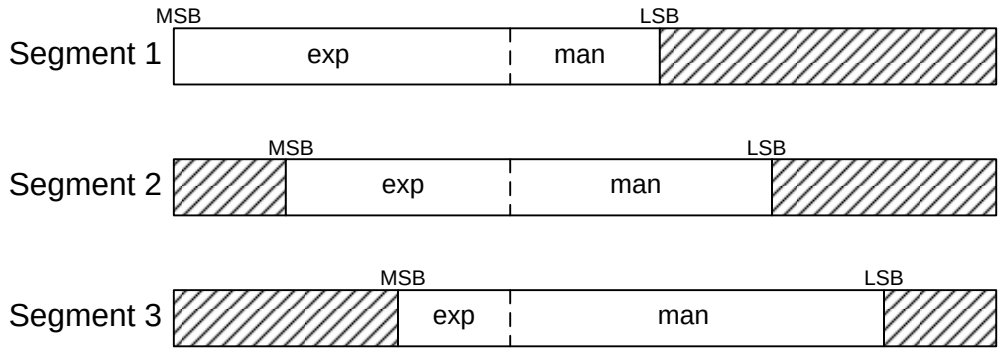

Figure 9: Block-floating point principle, each segment is a subset of the coefficients

2.3.2 Consequences of coefficient quantization

Quantization of the filter coefficients will cause the poles and zeros to move dependent on the error for the singularity which is given by a certain coefficient. Thus the frequency response is altered. Additionally, in IIR filters, the poles can move outside the unit circle and make the filter unstable. The effect can be estimated looking at the distribution of all possible poles or zeros for the given filter structure, with a given degree of coefficient quantization. Figure 10 shows all possible pole locations for a second-order IIR-structure with 4-bit coefficient quantization for a direct form and coupled form implementation respectively [7, p. 674]. It is evident that if the critical part of the filter, especially the transition-band, is at low frequencies, the direct-form realization will be significantly more sensitive to coefficient quantization. However, the errors at high frequencies will be largest for the coupled-form filter.

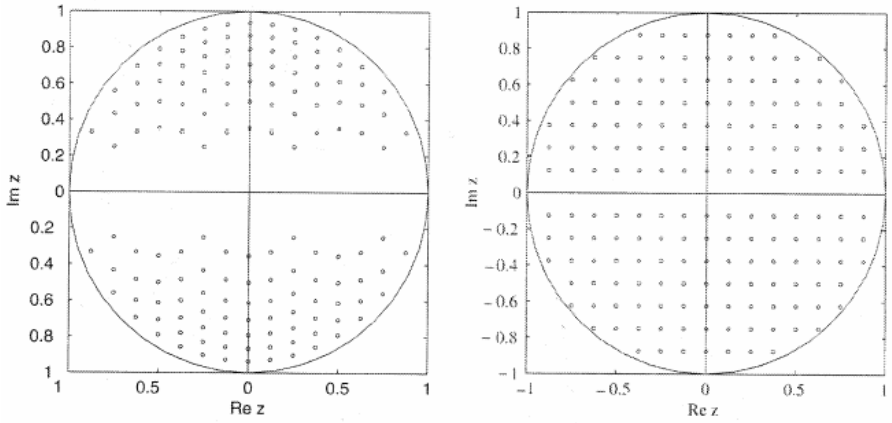

Figure 10: Possible pole-loactions for second order IIR-filter with 4-bit coefficients. Direct form (left) and coupled form (right) [7, p.674]

For high-order IIR-filters, the coefficient quantization effects are difficult to estimate analytically. For FIR-filters, general approximations of quantized coefficient effects can quite easily be found [7]:

An N'th order ideal FIR transfer function is given by:

$$H(z) = \sum_{n=0}^{N} h[n] z^{-n}$$

For a filter with quantized coefficients, we get:

$$\hat{H}(z) = \sum_{n=0}^{N} \hat{h}[n] z^{-n} = \sum_{n=0}^{N} \big(h[n] + e[n]\big) z^{-n} = H(z) + E(z)$$

This means the FIR-filter with quantized coefficients can be modelled as two parallel filters; the ideal filter and a filter with coefficients consisting of the different quantization errors respectively. If we look at the frequency response:

$$\hat{H}\left(e^{j\omega}\right) = H\left(e^{j\omega}\right) + E\left(e^{j\omega}\right)$$

The error must be bounded by:

$$\left|E\left(e^{j\omega}\right)\right| \le \sum_{n=0}^{N} \left|e[n]\right|$$

For rounding $\left|e[n]\right| \le \Delta/2$ and as a result:

$$\left|E\left(e^{j\omega}\right)\right| \le \frac{(N+1)\Delta}{2}$$

This is however a very conservative bound as it will be reached only if *every* coefficient quantization error has the absolute maximum value. For an uniformly distributed error where within $-\Delta/2 \le e[n] < \Delta/2$, the normal method of analyzing quantization errors, it can be found [7, p.681] that the error variance is bounded by:

$$\sigma_E{}^2(\omega) \le \frac{(2N+1)\Delta^2}{12}$$

Figure 11 shows an 87th order FIR-filter synthesized for 100dB stopband-damping without coefficient quantization and with 16-bit fixed-point quantization. It can be seen that the stopband-damping is reduced to a little over 80dB. The lower graphs show the pole-zero plot, with the unquantized version to the left.

As can be seen from the figure, it is in the stopband the zeros are moved the most, as they are there very small and the relative error is large. This confirms the reasoning behind the block floating-point approach.

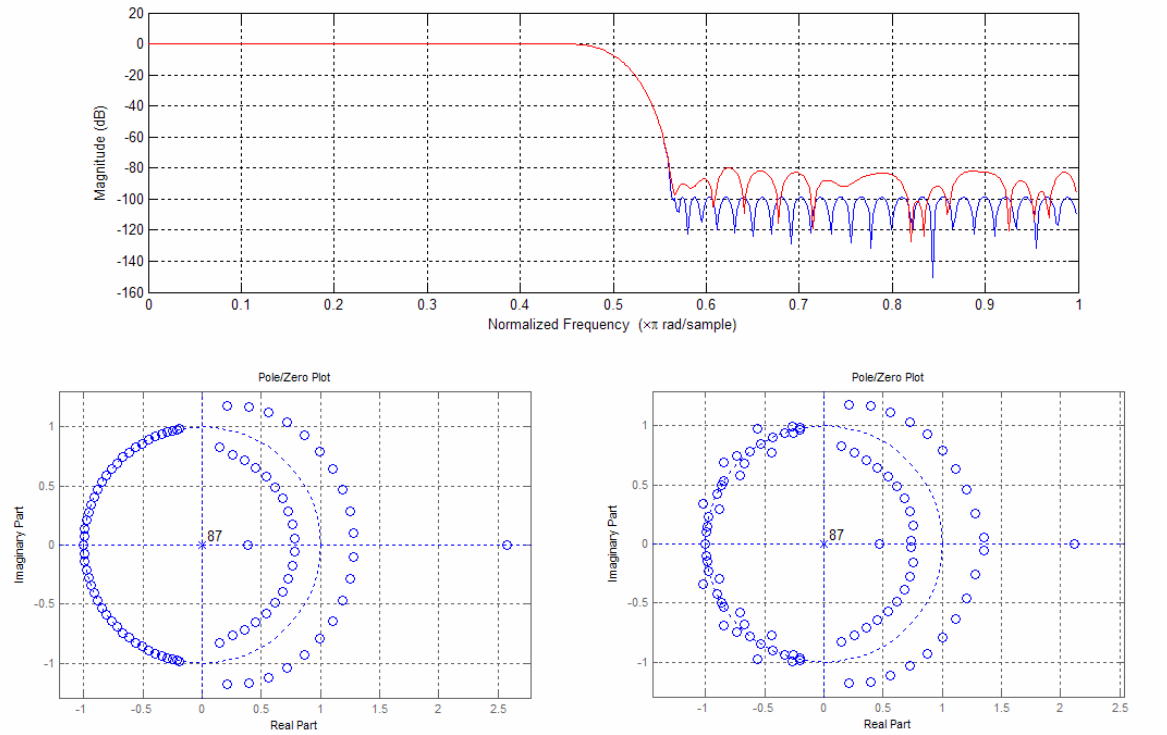

Figure 11: Frequency response and zero-location change, 16-bit coefficient quantization

2.3.3 Error-shaping of the coefficient quantization error

We have previously argued that the frequency-response in a delta-sigma DAC is less critical far into the stopband that in and around the transition region. This can be exploited by applying error-shaping to the filter coefficients. A simple method published by Gray [9] suggests to apply the coefficient vector, with the centre coefficient first, as the input to a delta-sigma modulator, with few-bit or one-bit quantization. The output error will then be shaped in the frequency domain, producing little error in the baseband and a large error far into the stopband. This is especially the case if the filter has a narrow passband, or in other words a large OSR. It should be noted that the delta-sigma modulator is not implemented as a part of the chip, it is only used in simulations to find the desired few-bit coefficients.

This method used by Scott to design an efficient 64-times OSR single-stage modulator, where all filter coefficients were truncated to three levels (-1,0 or 1) [10]. Then the filter can be realized with no multiplications whatsoever. The internal wordlength is also very-well controlled as no coefficients increase it. Furthermore no coefficient storage is needed either. Due to the spectral shaping and the narrow bandwidth, Scott argued that the filter still had good performance in the critical region consisting of the passband, transition band and lower region of the stopband.

Figure 12 shows the frequency response of a filter designed using this method. The Parks-McClellan algorithm was used to synthesize a filter for 128x OSR with 80dB stopband-damping. This is the same 5677-order filter shown in figure 2. Then, the filter was run through a fifth-order delta-sigma modulator optimized for 128x OSR and quantized to three levels. The red trace shows the frequency-response of the original filter while the blue trace shows the frequency response of the filter where all coefficients are either -1, 0 or 1. As can be seen, the quantized filter has very low attenuation at high frequencies, but in the passband, transition band and lower region of the stopband, it is very efficient.

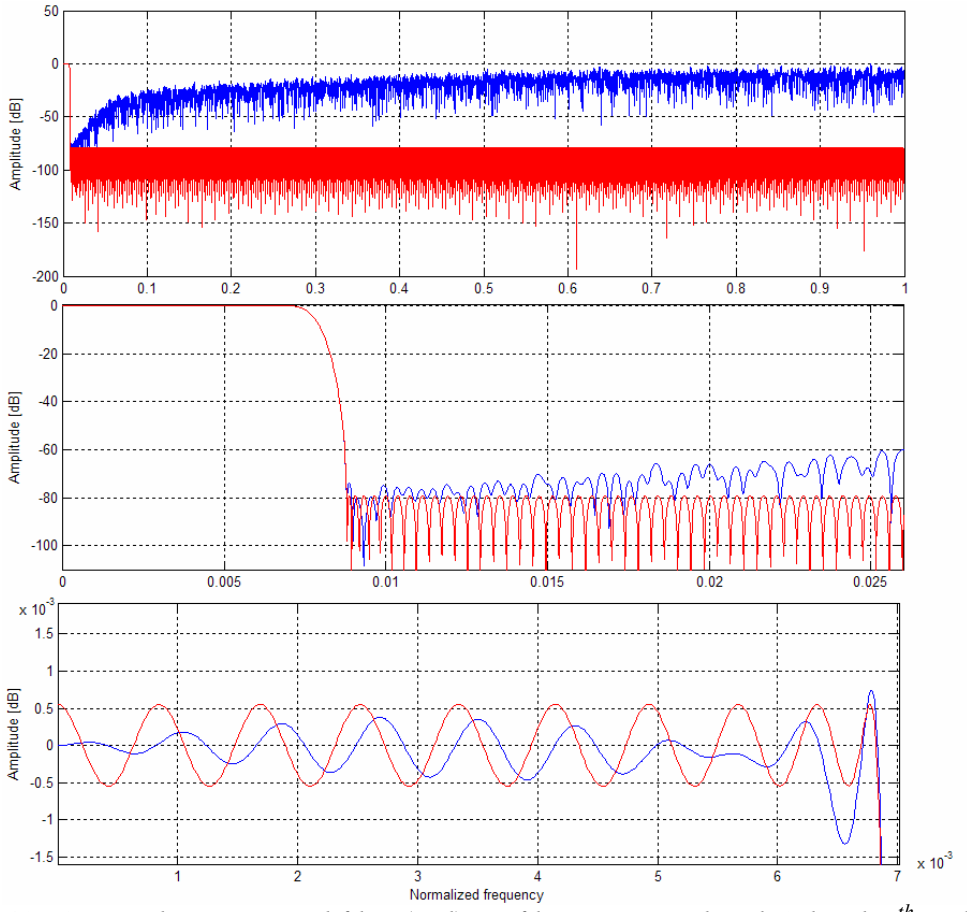

Figure 12: Original unquantized filter(red) vs filter quantized with 3-level, 5th order DSM

Up to about twice the passband range, the attenuation is 70dB or better. The passband-ripple is very low, increasing by only $0.8 \cdot 10^{-3}$dB in the worst case.

This technique is very useful in audio digital-to-analog converters, as the spectral image leakage will follow the same envelope as the output of the DAC modulator itself. Hence, the spectral energy-distribution of the final DAC output will not be significantly altered. This is a very viable alternative to the multistage, half-band method. However, it should be noted that since one-stage filters still have to be very, very long, the circuit will include many delay-elements. If these use less area and power than the multiplications in a many-bit implementation though, it should definitely be considered. It should also be noted that this method is not usable for ADC-applications or applications where there is subsequent sampling, as the large HF-images would then fold down into the baseband. This technique is tailor-made for highly interpolated delta-sigma DACs.

2.4 Coefficient sharing

The linear-phase FIR filter is symmetric. Hence, the number of stored coefficients can be halved by exploiting the fact that the coefficients on each side of the centre tap are equal. This is illustrated in figure 13.

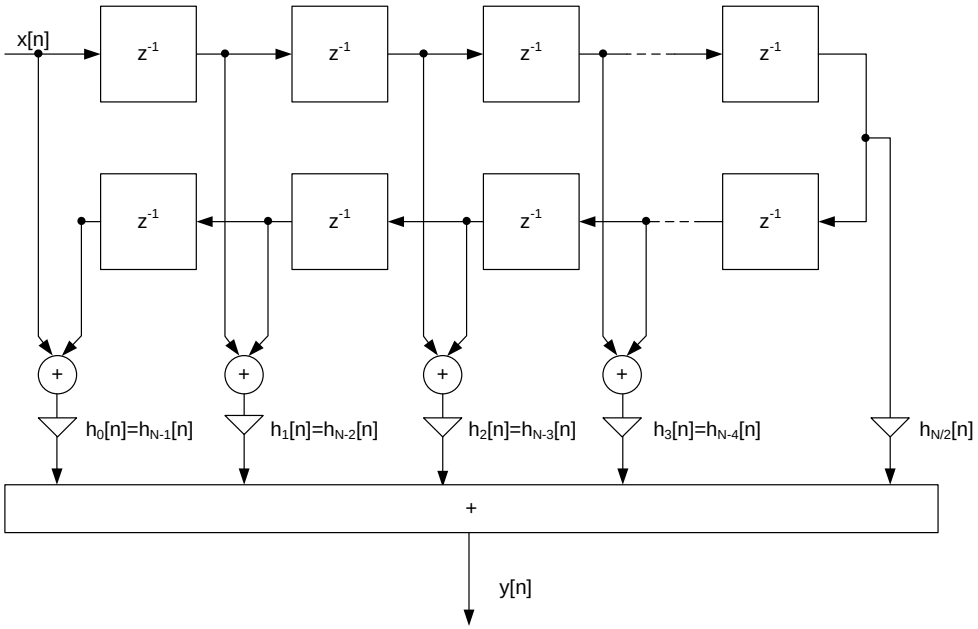

Figure 13: Symmetrical FIR filter with coefficient sharing

It should be noted that although the number of multiplicators are halved, the numbers of additions and delay-elements remain the same.

2.5 Filter implementation strategies

The filter implementation itself is non-trivial and the chosen strategy depends on the target hardware; ASIC, FPGA or DSP. We will look at a few much used implementation strategies with pros and cons.

2.5.1 Direct implementation

The most direct method of realization is to directly implement the filter in direct form. This is shown in figure 14. The advantage of the direct implementation is that it can run very fast, the disadvantage is that the large amount of delay-elements, multiplications and additions leads to a high resource usage.

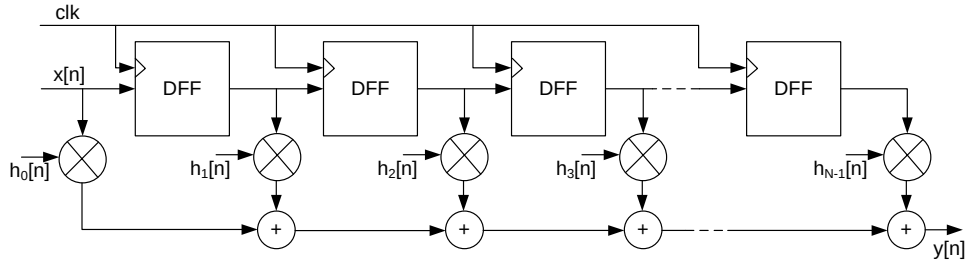

Figure 14: Direct implementation of filter transfer function

However, this strategy can be efficient in a multiplier-free realization, since it is multipliers that use most resources, especially in a hardware implementation. When quantized, the

coefficients can be realized as shift-and-add operations. For instance an 8-bit quantized coefficient must be an integer multiple of $1/2^8$. The value 71/256 might for instance be realized as:

$$\frac{71}{256} = \frac{1}{4} + \frac{1}{32} - \frac{1}{256}$$

To reduce the number of shifts, the expression may be nested:

$$\frac{71}{256} = \frac{1}{4}\left(1 + \frac{1}{8}\left(1 - \frac{1}{8}\right)\right)$$

While the first operation requires 15 right shifts, the second requires only 8.

Still, even when using shift operations to multiply, the large number of registers and adders will use a lot of area in a hardware implementation.

2.5.2 MAC implementation

A very popular way of realizing high-order FIR-filters is the multiplier-accumulator (MAC) implementation. Using this method, you rotate the coefficients in the ROM and multiply them with consecutive input samples. The output is added to an accumulator. When all sample·coefficient products are accumulated, a convolution sum is ready and is sent to the output. The input data must also be stored in a rotational register where new samples are shifted in for each complete convolution. The principle is shown in figure 15.

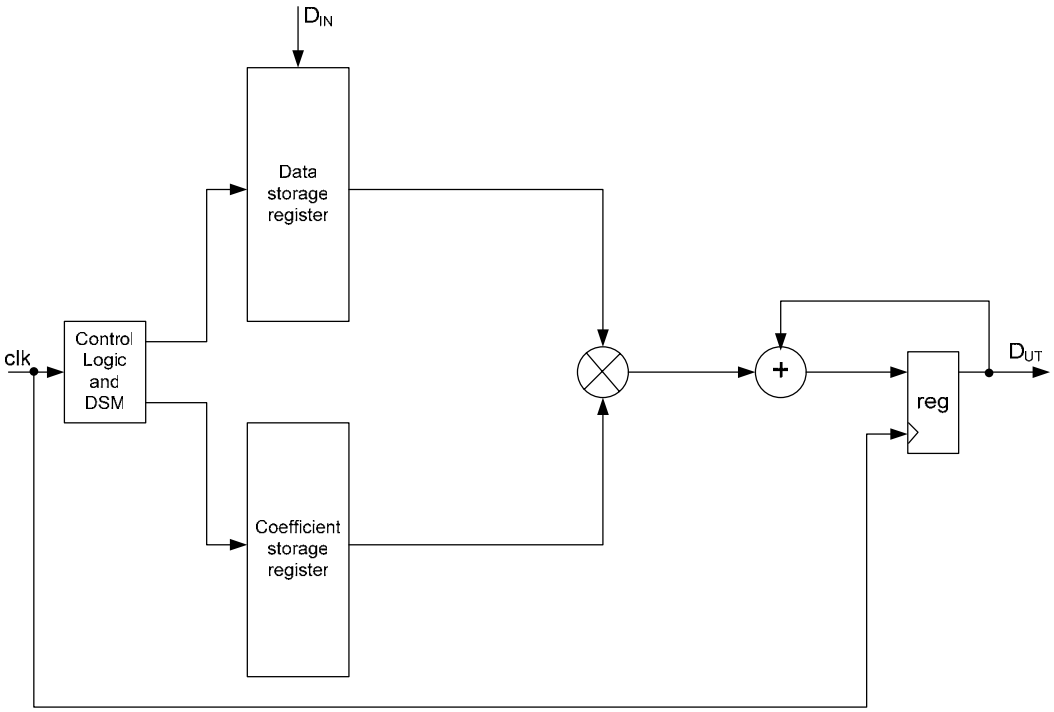

Figure 15: MAC FIR-filter implementation

The MAC-implementation requires two storage registers of length N for an N^{th}-order filter. However, as there is only one adder and one multiplier, the complexity of the circuit is significantly reduced. The drawback is that the registers, multiplier and accumulator must operate at N times the sample rate, since a full convolution must be performed for each input sample. This means the MAC-implementation is not very suited for high-speed operation, but for low sample rate applications, it represents are very area-efficient solution. In audio interpolator and filtering applications, it is the dominant method of implementation.

2.5.3 Distributed arithmetic implementation

The distributed arithmetic realization is quite different from the MAC or direct realization as it relies only on look-up-tables (LUTs), shift registers and a scaling accumulator, it does not employ any multiplicators at all. DA is principally a bit-serial operation that performs the dot-product of a pair of vectors in a single step [10]. We want to calculate a sum of products:

$$y = \sum_{n=1}^{N} A_n \cdot x_n$$

We can then define the n^{th} bit of the input sample x_n as b_{nk} and rewrite the expression on the form:

$$y = \sum_{k=1}^{K-1} \left[\sum_{n=1}^{N} A_n \cdot b_{nk} \right] + \sum_{n=1}^{N} A_n \cdot \left(-b_{n0} \right)$$

Considering the bracketed expression:

$$\sum_{n=1}^{N} A_n \cdot b_{nk}$$

Since b_{nk} is a single bit and can only take the values 0 or 1, the expression can only have 2^N possible values. Rather than computing these values on-line, they can be precomputed and stored in a LUT. The input-signal can then be used to address this LUT and the partial products be outputted to an accumulator. This is shown in figure 16.

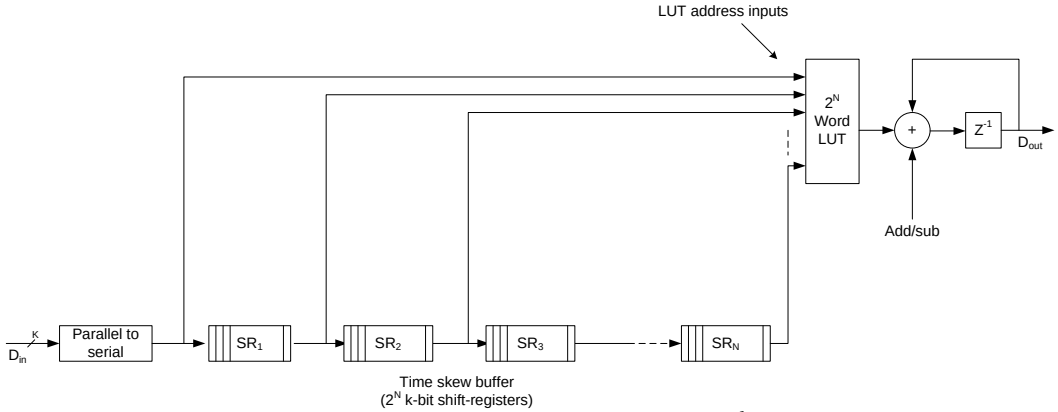

Figure 16: Direct Arithmetic implementation of N^{th} order FIR-filter

The advantage of the DA-filter is that the sample throughput is not dependent on the filter length. In a MAC-implementation, the number of operations per input sample is proportional to the filter length N, in a DA-implementation an output can be formed for each K^{th} clock cycle if the input precision is K bits, regardless of the filter length. This makes the DA approach much more suitable for high-speed implementation. The disadvantage is the necessity for a 2^N word LUT, which takes up quite a lot of area. For higher order filters, the MAC implementation is usually more area efficient, but as mentioned not very well suited for high sample-rates.

3 Transient response: The hidden distortion?

A linear phase FIR filter has an impulse response that is symmetric around the main impulse as shown in figure 17. This means that the ringing at the output starts before the main impulse, also called pre-ringing. This can also be seen from the step-response where we have ringing before the main step. The ringing is related to the discontinuity in the frequency response (known by the Gibbs phenomenon) and is consequently found to be at the filter cut-off frequency.

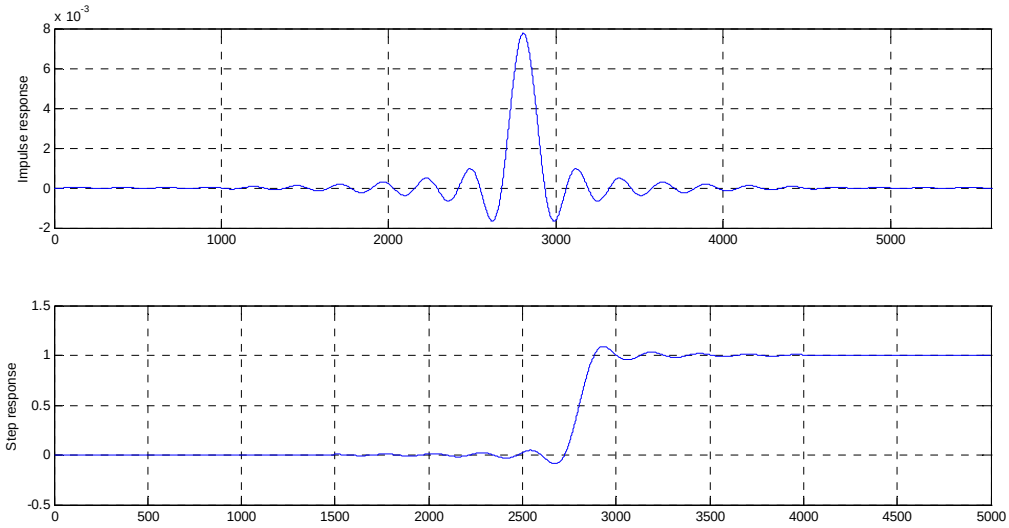

Figure 17: 5677-tap FIR-filter impulse and step response

The pre-ringing is half the impulse length or 2839 samples long. Since the filter is used at $128 \cdot f_s$ this means the pre-ringing starts 0.5ms before the step itself. It has been suggested that this negatively effects stereo imaging [4, 6], since the ear use temporal cues to localise sound sources. The time resolution of temporal localisation cues has, for certain spatial angles, been indicated to be as low as a few microseconds [12]. Since the filter pre-ringing blurs the onset of transient sounds, it is suggested that they will blur stereo imaging. Several producers of audiophile DAC units, like Audionote and Sakura Systems [13, 14], have consequently begun shipping products based on non-oversampling converters. It should however be noted that although a few listening tests have been published that do indicate an audible difference, pre-ringing has not been proved and quantified as an isolated source of quality degradation in itself as of yet.

But even if pre-ringing does lead to audible distortion, the purchase of a non-oversampling DAC is unlikely to alienate the problem for the consumer, as most recordings are done using

high oversampling ADCs with steep digital decimation filters at their outputs. Also, non-oversampling converters have significantly lower performance than their oversampling counterparts, and are very cost-inefficient. To attack this problem, filters with little or no pre-ringing should be developed.

One solution is to use asymmetrical minimum-phase FIR filters instead of symmetrical linear-phase ones. However, the minimum-phase filter has a non-linear phase response and consequently a frequency-variant group delay, which may again blur the spatial information in the audio signal. Also, minimum-phase filters usually have to be significantly longer for the same stopband and transition-band requirements and the computational burden is therefore significantly increased.

Another solution is to use shorter IIR-filters. However, as we know, these can also in most cases not be designed with linear phase. Also, IIR-filters are often expensive to implement in hardware, because of the transfer function consists of numerator and a denominator and thus requires a large division operation.

A suggested method that has been researched somewhat recently is the inclusion of an apodization filters [15]. Apodisation is a term much used in radio transmission and astronomy, where it means to apply a tapering function to surpress the sidelobes in an interferogram, or basically the same as windowing. A window function will round off the sharp edge of a brickwall (or close to brickwall) filter and thus reduce the ripple in the impulse response. An example of an apodizer can be a $cos^2(\pi f/2)$ filter function, better known as the hann-window. The impulse-response of the brickwall-filter is a sinc-function with a decay that is proportional to 1/t. When it is apodized with a hann-window, the decay is proportional to $1/t^3$. There are better apodizing functions however, [15] suggests an equalized cos^{10}-function that, unlike the cos^2, has a flat passband before smoothly turning down towards zero. With 192kHz input sample-rate, any ringing is then 60dB down 50µs from the main impulse and at -200dB after 100µs.

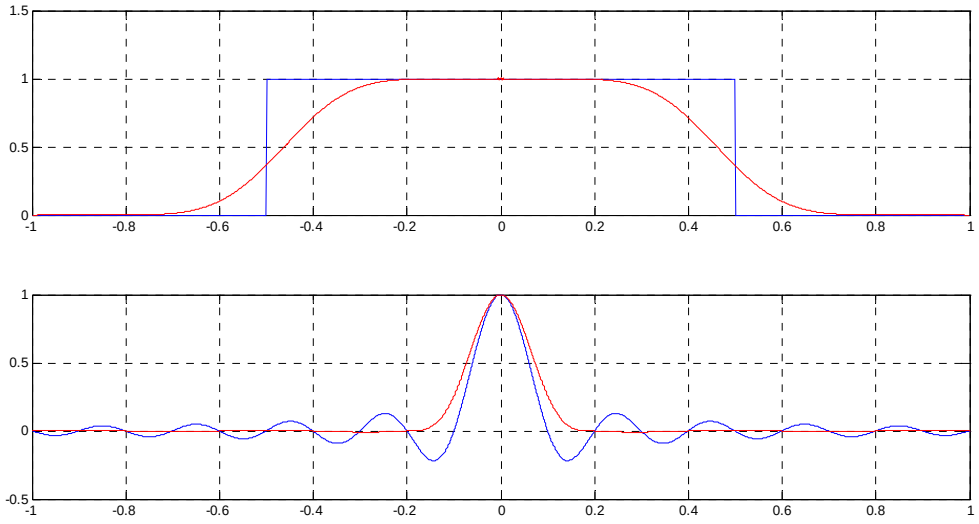

Figure 18: Top: Brickwall-filter and equalized cos^{10} apodization-filter. Bottom: Impulse response of brickwall and apodized filter.

The limitation to this approach lies in the fact that the apodization-filter must roll off in the passband of the filter that is to be apodized, this to smooth out the discontinuity in the brickwall response. Hence, the approach is only useable when the passband of the filter with the lowest cut-off frequency is significantly higher than the band of interest. In audio systems with sampling-rate of 96kHz or 192kHz, the nyquist frequency is well above the audible range, and a droop in the antialias filter passband is not of any concern. But even the equalized $\cos^{10}$-apodizer has 9dB droop at the passband edge, and can thus not be used with e.g. 44.1kHz CD, as you would then get a significant and audible treble attenuation. Also, if a chain has two apodizing-filters in cascade, the effect of the apodization will be nullified. This means that if apodization-filtering has been used during AD-conversion, the DAC should not contain any and vice versa. The use of apodization should thus be standardised, to be included in either the input or playback chain, but not both.

We have in this chapter taken a brief look at the FIR transient response and how it might possibly lead to transient distortion. It should however be stressed that it is not rigorously proved that this distortion is audible. We have seen at possible solutions, including minimum-phase filters and apodization and given a short assessment of their pros and cons. This is an area where little documentation has been published, and in the author's opinion, the researchers in psychoacoustics should probably provide more rigorous proof to its relevance before electrical engineers allocate significant resources to resolving it.

4 References

[1] Fujimori, Nogi, Sugimoto: "*A Multibit Delta-Sigma Audio DAC with 120dB Dynamic Range*", IEEE JSSCC, August 2000
[2]: Nakano, Terasawa: "*A 117dB Current-Mode Multi-Bit Audio DAC for PCM and DSD Audio Playback*", AES Convention Paper 5190, September 2000
[3]: Adams, Ngyuen, Sweetland: "*A 112dB Oversampling DAC with Segmented Noise-Shaped Scrambling*", AES Convention Paper 4774, August 1998
[4]: Harris, Kelly, McLeod, Story: "*Effects in High Sample-Rate Material*", DCS Technical Paper, Presented at the 20th Tonmeistertagung, Kalsruhe, Germany 1998
http://www.dcsltd.co.uk/technical_papers/effects.pdf
[5]: R. Lagadec and T. Stockham: "Dispersive Models for A-to-D and D-to-A Conversion Systems", JAES, vol.32, p.469, June 1984
[6] Black, R: "*Antialias-filters: The Invisible Distortion Mechanism in Digital Audio*", AES Convention Paper 4966, May 1999.
[7]: Mitra, S: "*Digital Signal Processing: A Computer-Based Approach*", McGraw-Hill International Press, Third Edition, 2005
[8]: D. Wise, "*Block Floating-Point FIR-filters Using Fixed-Point Multipliers*", AES Convention Paper 5175, September 2000.
[9]: P.W.Song and R.M.Gray: "*FIR Filters with Sigma-Delta Modulation Encoding*", IEEE Transactions on Acoustics, Speech and Signal Processing, vol.38, p. 979-990, June 1990.
[10]: J.W. Scott: "*Multiplier-free Interpolation for Oversampled Digital-to-Analog Conversion*", AES Convention Paper 3317, March 1992.
[11]: S.A. White: "*Applications of Distributed Arithmetic to Digital Signal Processing*", IEEE ASSP Magazine, vol.6, p.4-19, July 1989.
[12]: M.D. Wilde: "*Temporal Localisation Cues and Their Role in Auditory Perception*", AES Convention Paper 3708, October 1993.
[13]: www.audionote.co.uk
[14]: http://www.sakurasystems.com
[15]: Craven, P: "*Antialias Filters and System Transient Response at High Sample Rates*", JAES, vol 52, no. 3, pp 216-242, March 2004