Chapter 2

How are bits played back from an audio CD?

T. Dutoit (°), R. Schreier (*)

(°) Faculté Polytechnique de Mons, Belgium
(*) Analog Devices, Inc., Toronto, Canada

An audio digital-to-analog converter adds noise to the signal, by requantizing 16-bit samples to one-bit. It does it...on purpose.

Loading a CD player with one's favorite CD has become an ordinary action. It is taken for granted that the stream of 16-bit digital information it contains can easily be made available to our ears, i.e., in the analog world in which we live. The essential tool for this is the digital-to-analog converter (DAC).

In this chapter we will see that, contrary to what might be expected, many audio DACs (including those used in CD and MP3 players, for instance, or in cell phones) first requantize the 16-bit stream into a *1-bit* stream[1] with very high sampling frequency, using a signal processing concept known as delta–sigma[2] ($\Delta\Sigma$) modulation, and then convert the resulting bipolar signal back to an audio waveform.

The same technique is used in ADCs for digitizing analog waveforms. It is also the heart of the direct stream digital (DSD) encoding system implemented in super audio CDs (SACDs).

---

[1] In practice, 1-bit $\Delta\Sigma$ DACs have been superseded by multiple-bit DACs, but the principle remains the same.

[2] Sometimes also referred to as *sigma–delta.*

T. Dutoit, F. Marqués (eds.), *Applied Signal Processing*,
DOI 10.1007/978-0-387-74535-0_2,

2.1 Background – Delta–sigma modulation

An N-bit DAC converts a stream of discrete-time linear PCM[3] samples of N bits at sample rate F_s to a continuous-time voltage. This can be achieved in many ways. Conventional DACs (Section 2.1.2) directly produce an analog waveform from the input PCM samples. Oversampling DACs (Section 2.1.3) start by increasing the sampling frequency using digital filters and then make use of a conventional DAC with reduced design constraints. Adding noise shaping makes it possible to gain resolution (Section 2.1.4). Delta–sigma DACs (Section 2.1.5) oversample the input PCM samples and then requantize them to a 1-bit data stream, whose low-frequency content is the expected audio signal.

Before examining these DACs, we start with a review of uniform quantization (Section 2.1.1), as it will be used throughout the chapter.

2.1.1 Uniform quantization: Bits vs. SNR

Quantization lies at the heart of digital signal processing. An N-bit uniform quantizer maps each sample $x(n)$ of a signal to one out of 2^N equally spaced values $X(n)$ in the interval $(-A, +A)$, separated by the quantization step $q=2A/2^N$. This operation (Fig. 2.1) introduces an error $e(n)$:

$$e(n) = X(n) - x(n) \qquad -q/2 \le e(n) \le q/2 \tag{2.1}$$

If the number of bits is high enough and the input signal is complex, the quantization error is equivalent to uniform white noise in the range[$(-q/2, +q/2)$]. It is easy to show that its variance is then given by the following equation:

$$\sigma^2_{ee} = \frac{q^2}{12} \tag{2.2}$$

The main result of uniform quantization theory, which can be found in most signal processing textbooks, is the standard "1 bit = 6 dB" law, which gives the expression of the signal-to-quantization-noise ratio:

$$SNR(dB) = 10 * \log_{10}\left(\frac{\sigma_{xx}^2}{\sigma_{ee}^2}\right) \tag{2.3}$$

as a function of N and A, in the absence of saturation:

[3] PCM stands for pulse code modulated.

$$SNR(dB) = 6.02N + 4.77 + 20\log_{10}(\Gamma) \qquad (\Gamma = A/\sigma_{xx}) \qquad (2.4)$$

where Γ is the load factor defined as the saturation value of the quantizer normalized by the standard deviation of the input signal.

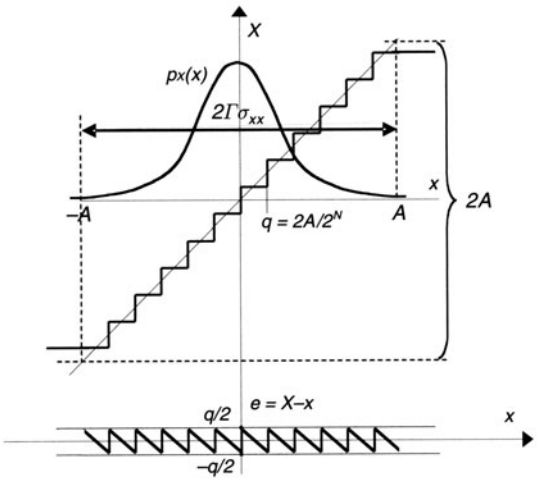

Fig. 2.1 Uniform quantization[4]

When the amplitude of the input signal approaches the quantization step, the quantization error becomes correlated with the signal. If the signal itself is not random, the quantization error can then be heard as nonlinear audio distortion (rather than as additive noise).

This can be avoided by *dithering*, which consists of adding real noise to the signal before quantizing it. It has been shown that a triangular white noise (i.e., white noise with a triangular probability density function) in the range $[-q,+q]$ is the best dither: it decorrelates the quantization noise with the signal (it makes the mean and variance of the quantization noise independent of the input signal; see Wannamaker 1997) while adding the least possible noise to the signal. Such a noise is easy to obtain by summing two independent, uniform white noise signals in the range $[-q/2,+q/2]$, since the probability density function of the sum of two independent random variables is the convolution of their respective pdfs.

[4] We actually show a *mid-rise* quantizer here, which quantizes a real number x into (floor(x/q)+0.5)*q. Mid-thread quantizers, which compute floor(x/q+0.5)*q, are also sometimes used (see Section 3.2.5, for instance).

As Wannamaker puts it: "Appropriate dithering prior to (re)quantization is as fitting as appropriate anti-aliasing prior to sampling – both serve to eliminate classes of signal-dependent errors."

2.1.2 Conventional DACs

A conventional DAC uses analog circuitry (R/2R ladders, *thermometer* configuration, and others; see, for instance, Kester and Bryant 2003) to transform a stream of PCM codes $x(n)$ (at the sampling frequency F_s) into a staircase analog voltage $x^*(t)$, in which each stair lasts $T_s=1/F_s$ seconds and is a linear image of the underlying PCM code sequence. Staircase signal $x^*(t)$ is then smoothed with an analog low-pass filter $S(f)$ (Fig. 2.2), which suppresses the spectral images.

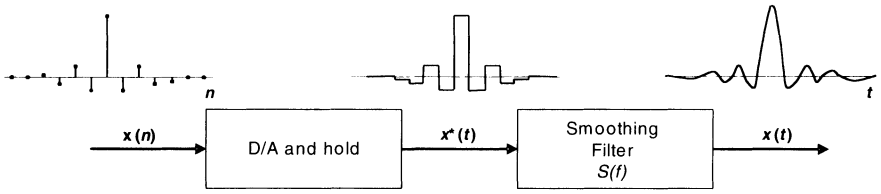

Fig. 2.2 Conventional DAC

The first operation can be seen as convolving a sequence of Dirac pulses $x^+(t)$ obtained from PCM code with a rectangular wave of length T_s, hence filtering the Dirac pulses with the corresponding low-pass filter (Fig. 2.3). The second operation completes the smoothing job.

Conventional DACs have several drawbacks. First, they require high-precision analog components and are very vulnerable to noise and interference. In a 16-bit DAC with 3 V reference voltage, for instance, one half of the *least significant bit* corresponds to 2^{-17} 3 V=23 μV. What is more, they impose hard constraints on the design of the analog smoothing filter, whose transition band must fit within $[F_m, F_s-F_m]$ (where F_m is the maximum frequency of the signal; Fig. 2.3), so as to efficiently cancel spectral images.

2.1.3 Oversampling DACs

An oversampling DAC first performs digital K times oversampling of the input signal $x(n)$ by first inserting $K-1$ zeros between each sample of $x(n)$ and then applying digital low-pass filtering with passband equal to

$[0,F_s/K]$ (Fig. 2.4). The resulting signal, $x(n/K)$, is then possibly requantized on N' bits (with $N'<N$), by simply keeping the N' most significant bits from each interpolated sample, and the requantized signal $x'(n/K)$ is sent to a conventional DAC clocked at $K*F_s$ Hz with N' bits of resolution.

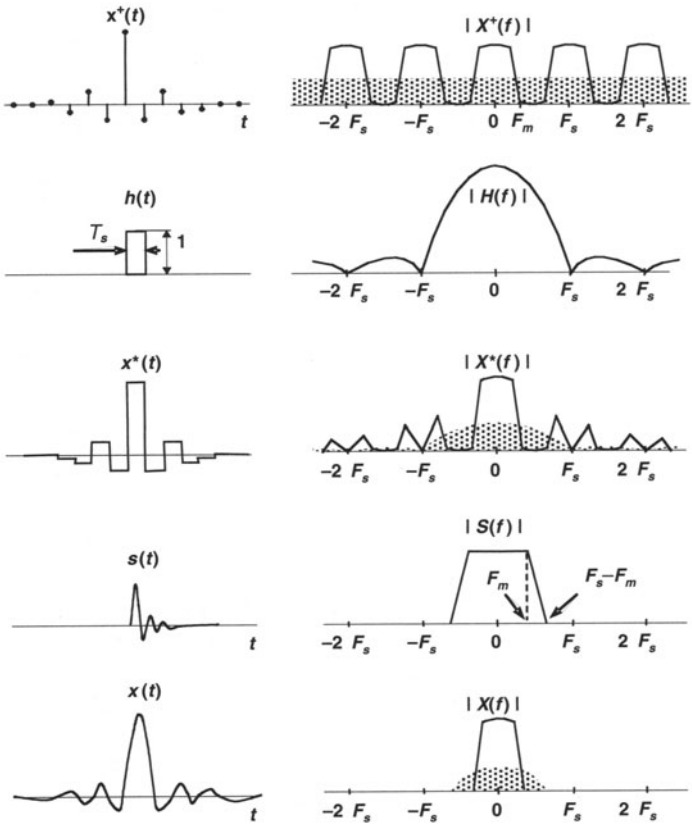

Fig. 2.3 Digital-to-analog conversion seen as double filtering: $x^+(t)$ is convolved with $h(t)$ to produce $x^*(t)$, which is smoothed by $S(f)$. Quantization noise is shown as superimposed texture

In principle, requantizing to less than the initial N bits lowers the SNR by 6 dB for each bit lost. However, although the variance of quantization noise $e'(n)$ generated by N'-bit requantization is higher than that of the initial N-bit quantization noise $e(n)$, it is now spread over a (K times) larger frequency range. Its power spectral density (PSD, in V^2/Hz) is thus given by the following equation:

$$S_{e'e'}(f) = \frac{\sigma_{e'e'}^2}{K F_s} \tag{2.5}$$

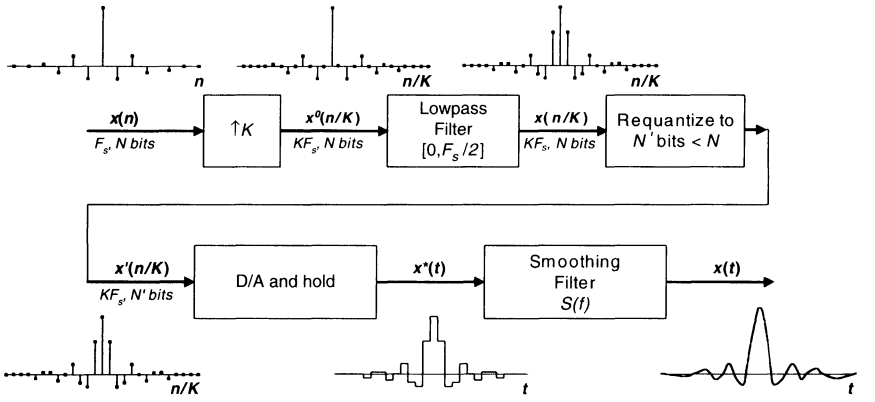

Fig. 2.4 Oversampling DAC (shown here with oversampling ratio set to 2, for convenience)

Since only a fraction of this PSD (typically, $1/K$) will eventually appear in the analog output, thanks to the action of the smoothing filter, the effective SNR is higher than its theoretical value. In practice, each time a signal is oversampled by a factor 4, its least significant bit (LSB) can be ignored. As a matter of fact, the 6 dB increase in SNR is compensated by a 6 dB decrease due to the fact that only one-fourth of the variance of the new quantization noise is in the range $[0, F_s/2]$.[5]

Physically speaking, this is perfectly understandable; successive requantization noise samples $e'(n)$ produced at KF_s are independent random variables with variance equal to $q'^2/12 > q^2/12$. The low-pass smoothing filter performs a weighted average of neighboring samples, thereby reducing their variance.

This effect is very important in practice, as it allows a bad resolution DAC (N' bits$<N$) to produce high-resolution signals.

As a result of oversampling, the smoothing filter is also allowed a larger transition bandwidth $[0, KF_s - F_m]$ (Fig. 2.5). What is more, the same low-pass filter can now be used for a large range or values for F_s (as required for the sound card of a computer, for instance).

[5] Strictly speaking, in $[-F_s/2, F_s/2]$. In this discussion, we consider only positive frequencies, for convenience.

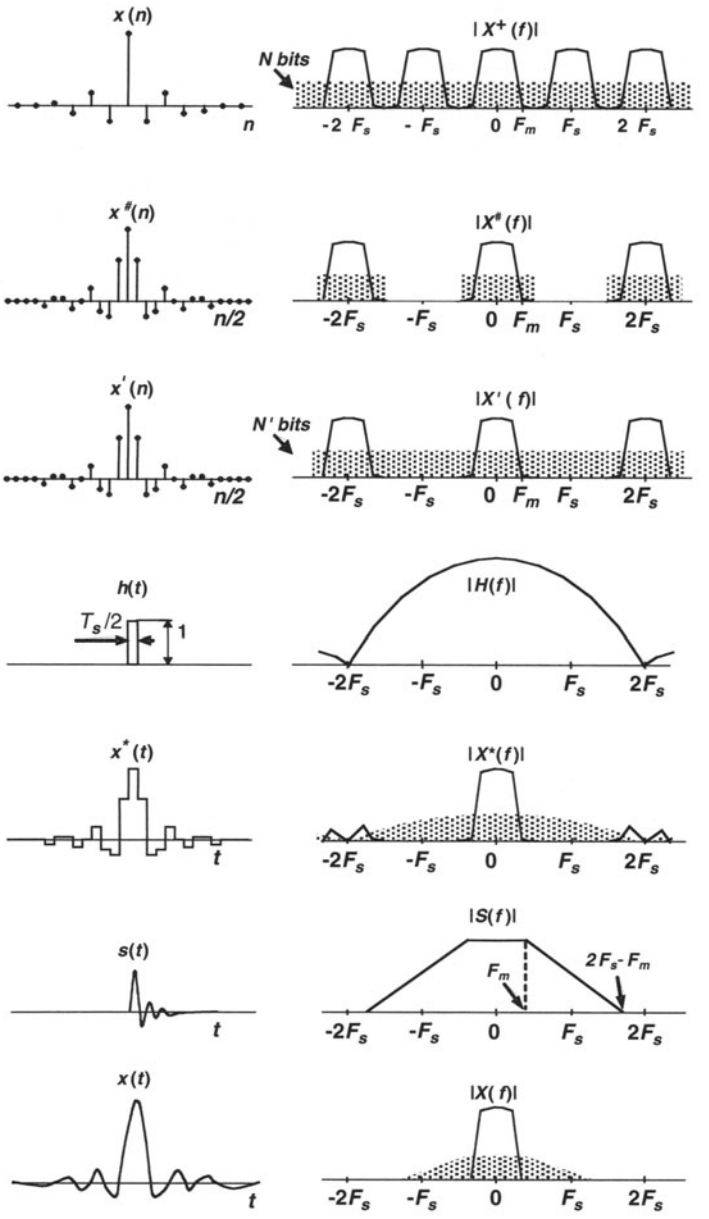

Fig. 2.5 Oversampling (shown here with an oversampling factor of 2) prior to digital-to-analog conversion. Quantization and requantization noise are shown as superimposed texture

2.1.4 Oversampling DACs – Noise shaping

Oversampling alone is not an efficient way to obtain many extra bits of resolution: gaining B bits requires an oversampling ratio of 4^B, which quickly becomes impractical. An important improvement consists of performing *noise shaping* during requantization. Instead of keeping N' bits from each interpolated sample $x(n)$, a noise-shaping quantizer implements a negative feedback loop between $x(n)$ and $x'(n)$ (Fig. 2.6), whose effect is to push the PSD of the requantization noise toward frequencies far above $F_s/2$ (up to K*$F_s/2$) while keeping the PSD of the signal untouched. As a result, the effective SNR is further increased (Hicks 1995).

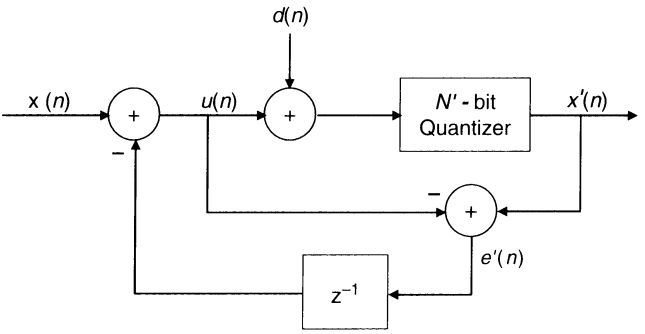

Fig. 2.6 First-order noise-shaping (re)quantizer

As a matter of fact, we have

$$\begin{aligned} U(z) &= X(z) - z^{-1}(X'(z) - U(z)) \\ &= \frac{X(z) - z^{-1}X'(z)}{1 - z^{-1}} \end{aligned} \tag{2.6}$$

and since the combined effect of dithering and quantization is to add some white quantization noise $e'(n)$ to $u(n)$

$$\begin{aligned} X'(z) &= U(z) + E'(z) \\ &= \frac{X(z) - z^{-1}X'(z)}{1 - z^{-1}} + E'(z) \\ &= X(z) + (1 - z^{-1})E'(z) \end{aligned} \tag{2.7}$$

which shows that the output of the noise-shaping requantizer is the input signal plus the first derivative of the white quantization noise $e'(n)$

produced by requantization. This effectively results in colored quantization noise $c(n)=e'(n)-e'(n-1)$, with most of its PSD in the band $[F_s/2, KF_s/2]$ (Fig. 2.7, to be compared to Fig. 2.5), where it will be filtered out by the smoothing filter. As the noise-shaping function $(1-z^{-1})$ is of first order, this configuration is termed as a first-order noise-shaping cell.

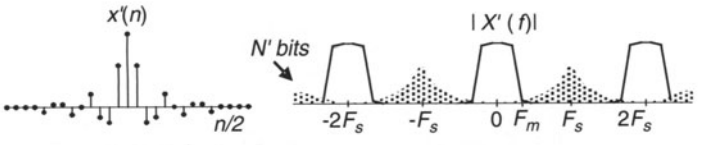

Fig. 2.7 The effect of noise shaping combined with oversampling (by a factor 2) on signal $x'(n)$ at the output of the quantizer

Noise shaping does increase the power of quantization noise, as the variance of colored noise $c(n)$ is given by the following equation:

$$\begin{aligned}\sigma_{cc}{}^2 &= \frac{\sigma_{e'e'}{}^2}{K F_s}\int_{-KF_s/2}^{-KF_s/2} |1-e^{-j2\pi f}|^2\, df \\ &= \frac{\sigma_{e'e'}{}^2}{K F_s}\int_{-KF_s/2}^{-KF_s/2} (1+|e^{-j\theta}|^2)\, df \\ &= 2\sigma_{e'e'}{}^2\end{aligned} \tag{2.8}$$

But again, since this variance is mostly pushed in the $[F_s/2, KF_s/2]$ band, the effective SNR can be lowered (Fig. 2.8).

This technique makes it possible to gain 1 bit every time the signal is oversampled by a factor 2. It was used in early CD players when only 14-bit hardware D/A converters were available at low cost. By combining oversampling and noise shaping (in the digital domain), a 14-bit D/A converter was made comparable to a 16-bit D/A converter.

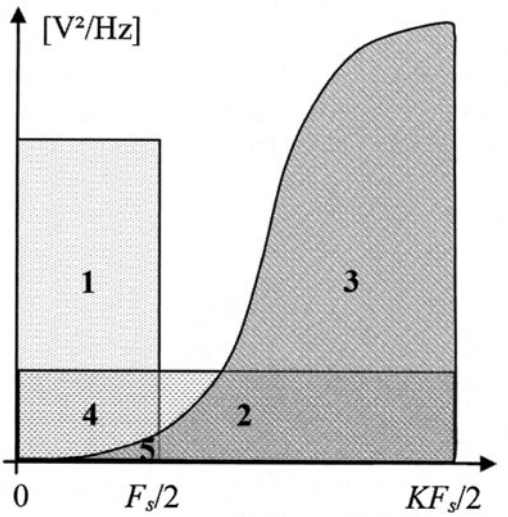

Fig. 2.8 A comparison of quantization noise power density functions for the same number of bits. (1) With a conventional DAC; effective noise variance = area 1; (2), With an oversampling DAC, effective noise variance = area 4; (3) With an oversampling DAC using noise shaping, effective noise variance = area 5.

2.1.5 Delta–sigma DACs

The delta–sigma architecture is the ultimate extension of the oversampling DAC and is used in most voiceband and audio signal processing applications requiring a D/A conversion. It makes use of a very high oversampling ratio, which makes it possible to requantize the digital signal to 1 bit only. This 1-bit signal is then converted to a purely bipolar analog signal by the DAC, whose output switches between equal positive and negative reference voltages (Fig. 2.9).

The bipolar signal is sometimes referred to as "pulse-density modulated" (PDM), as the density of its binary transitions is a function of the amplitude of the original signal.

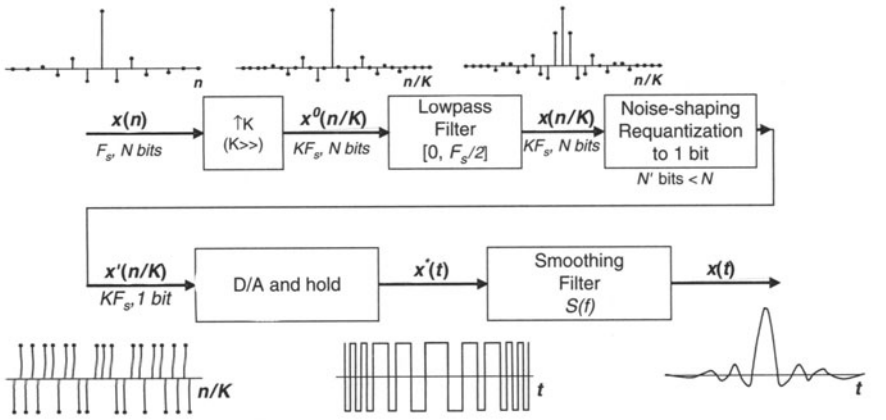

Fig. 2.9 Delta–Sigma DAC (shown here with oversampling ratio set to 2, for convenience; in practice, much higher ratios are used)

In CD and MP3 players, this implies a gain of 15 bits of resolution. Improved noise shaping is therefore required, such as second-order noise shaping cells (whose noise-shaping function is $(1-z^{-1})^2$) or cascades of first-order noise-shaping cells (termed as MASH: multi-stage noise shaping; Matsuya et al. 1987). Deriving a general noise-shaping quantizer with noise-shaping function $H(z)$ from that of Fig. 2.6 is easy: one simply needs to replace the delay by $1-H(z)$ (Fig. 2.10).

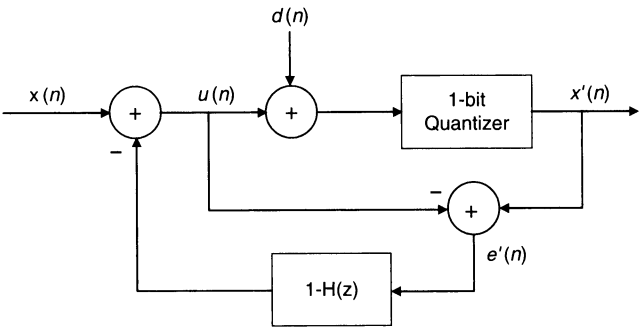

Fig. 2.10 General digital delta–sigma modulator

The influence of the oversampling ratio and of the order of the noise-shaping filter on the noise power in the signal bandwidth is given in Fig. 2.11.

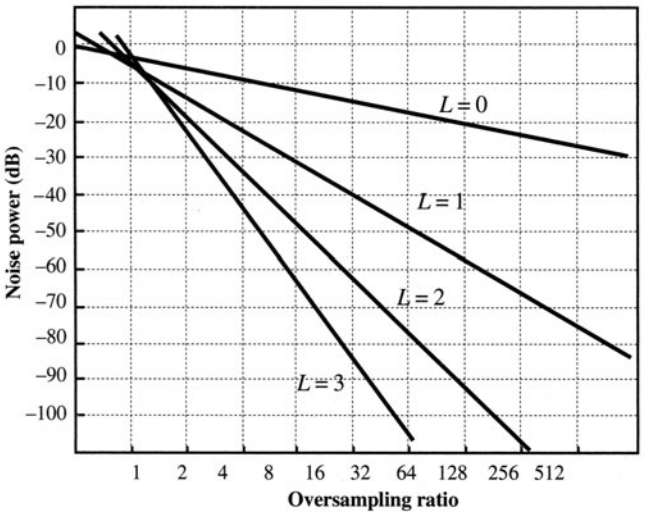

Fig. 2.11 Power of the noise in the signal bandwidth as a function of the oversampling ratio for various noise-shaping orders L. 0 dB of noise level corresponds to that of PCM sampled at the Nyquist rate. All quantizers use the same level spacing (after Candy (1997))

The very limited bandwidth allowed for the interpolation filter ([0, $F_s/2$] at a sampling frequency of KF_s) theoretically implies a very high order for its digital implementation (especially if it is synthesized as a linear-phase FIR filter), i.e., many additions and multiplications per sample. Oversampling is therefore systematically performed in two steps: a first step that does most of the oversampling and uses a *comb filter* (or cascades thereof), for the interpolation, followed by a second step competing with lower oversampling ratio and a more efficient interpolation filter. A comb filter of order N is an FIR filter with all coefficients set to 1:

$$H_N(z) = \sum_{n=0}^{N-1} z^{-n} = \frac{1-z^{-N}}{1-z^{-1}} \tag{2.9}$$

This filter has linear phase, while being straightforwardly implemented as

$$y(n) = x(n) - x(n-N) + y(n-1) \tag{2.10}$$

which implies no multiplication. Comb filters, however, cannot provide very efficient frequency responses (for an example, see Fig. 2.16-left). Their transition band is not sharp. This is counterbalanced by the second interpolation stage, which produces a sharp cut-off and additionally compensates for the frequency droop of the comb filter around $F_s/2$.

In summary, delta–sigma DACs are a wonderful example of mixed A–D signal processing constraints. Most of their load is in the digital domain, while the analog output smoothing filter usually reduces to a simple RC lowpass. These converters can thus be fabricated on a wide range of IC processes, which implies low cost and robustness to time and temperature drifts. This is achieved at the expense of speed: since the hardware has to operate at the high oversampling rate, sigma–delta converters are usually limited to sampling rates below 1 MHz.

From 1990 to 2000, the consumer audio industry has produced a large number of 1-bit sigma–delta-based converters, mainly because this technique leads to cheaper manufacturing. It was argued, however, that 1-bit sigma–delta converters were not suitable for high-quality audio applications (Lipschitz and Vanderkooy 2001), as they allow only partial dithering to be performed.[6] Most audio DACs made since 2000 implement *multi-bit* sigma–delta converters, which take the best of both the 1-bit and the 20+-bit worlds (go to the Analog Devices web page, for instance, and search for "audio DAC").

2.2 MATLAB proof of concept: ASP_audio_cd.m

In this Section, we will first revise the basics of uniform quantization (Section 2.2.1), including the important part played by dithering (Section 2.2.2). We will then compare the internals of a conventional DAC (Section 2.2.3) to those of more advanced DACs using oversampling (Section 2.2.4), noise-shaping (Section 2.2.5), and delta–sigma modulation (Section 2.2.6).

2.2.1 Uniform quantization

In order to perform D/A conversion, we first need to create a PCM signal. We will first check the main results of uniform quantization theory, which we will use later.

[6] This led to an interesting controversy in several papers published in conventions of the Audio Engineering Society.

Let us first generate 8000 samples of uniform white noise with zero mean, which we will assume to have been sampled at F_s=8,000 Hz. For convenience, we set its variance to 1 (0 dB) by imposing its range to [–*peak*, +*peak*], with $peak = \sqrt{12}/2 \approx 1.73$. This is confirmed on the power spectral density of the signal (Fig. 2.12).

```
signal_std=1;
peak=sqrt(12*signal_std)/2;
signal=(rand(1,8000)-0.5)*2*peak;

pwelch(signal,[],[],[],2); [7]
```

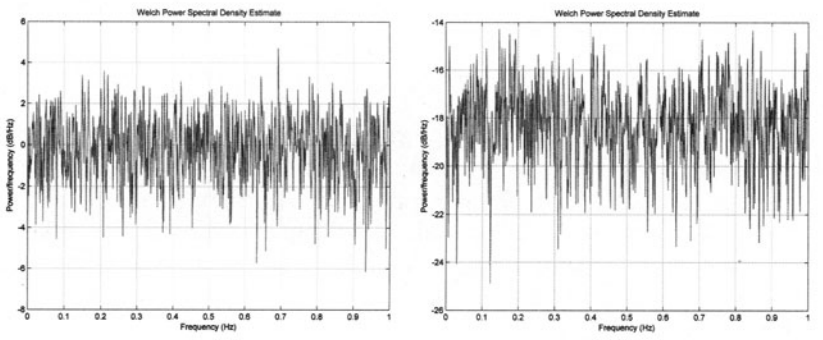

Fig. 2.12 Power spectral density of the input signal (*left*) and of the 6-bit quantized error (*right*)

Quantizing this signal uniformly on *N* bits is easy. Let us choose *N*=3 for convenience. We use mid-rise quantization (hence the +q/2), which is best suited for an even number of quantization steps.

```
N=3;
quantizer_saturation=peak;
q=(2*quantizer_saturation)/2^N;
signal_quantized=floor(signal/q)*q+q/2; % mid-rise
error=signal-signal_quantized;
```

The quantization noise is also white, as revealed by power spectral density estimation. The variance of the noise can thus be read (in dB) on the PSD plot (Fig. 2.12). From the load factor (i.e., the saturation value of the quantizer normalized by the standard deviation of the signal), we can

[7] MATLAB's `pwelch` function shows the PSD in the range [0, F_s/2]. The PSD values it shows are such that their integral over [0, F_s/2] equals the variance of the signal. Claiming F_s=2 as an input argument therefore shows the variance as the average PSD over [0, 1].

compute the theoretical SNR due to quantization. It matches to the SNR computed from the samples and corresponds to the variance of the noise (with opposite sign), since the variance of the signal is 0 dB.

MATLAB function involved:

- `snr = snr(signal, signal_plus_noise, max_shift, showplot)` returns the signal-to-noise ratio computed from the input signals. `Max_shift` gives the maximum time shift (in samples) between `signal` and `signal_plus_noise`. The actual time shift (obtained from the maximum of the cross-correlation between both signals) is taken into account to estimate the noise. If `showplot` is specified, then the `signal`, `signal_plus_noise`, and `error` are plotted, and the SNR is printed on the plot.

```
pwelch(error,[],[],[],2);
load_factor=quantizer_saturation/signal_std;
snr_theory=6.02*N + 4.77 - 20*log10(load_factor)
snr_measured=snr(signal,signal_quantized,0)
```

```
snr_theory =  18.0588

snr_measured =  18.1119
```

2.2.2 Dithering

Let us apply this to a sine wave with fundamental frequency f_0=200 Hz, sampled at F_s=8,000 Hz, and set its variance to 1 (0 dB) by imposing its peak to $\sqrt{2}$.

```
signal_std=1;
peak=sqrt(signal_std*2);
signal=peak*sin(2*pi*200*(0:1/8000:2));
var_signal_dB=10*log10(var(signal))
```

```
var_signal_dB = -2.8930e-015
```

When we quantize it uniformly to 3 bits in the range [–2,+2], we note that the quantization error does not look like real noise (Fig. 2.13).

```
N=3;
quantizer_saturation=2;
q=2*quantizer_saturation/2^N;
signal_quantized=floor(signal/q)*q+q/2; % quantization
error=signal-signal_quantized;

plot(signal(1:50),'-.'); hold on;
plot(error(1:50));hold off
```

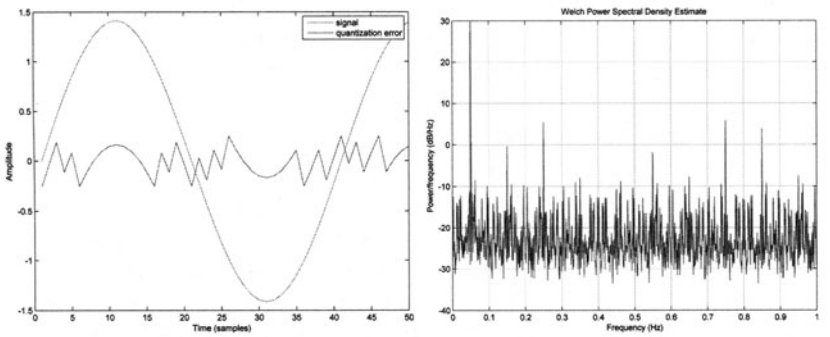

Fig. 2.13 Quantization error (*left*) and power spectral density of the quantized signal (*right*) without dithering. Frequencies are normalized with respect to $F_s/2$

Harmonics of the original frequency appear in the spectrum of the quantized signal (Fig. 2.13) and sound very unpleasant. The SNR is not directly readable on the PSD plot because of these harmonics.

```
snr_quantized=snr(signal,signal_quantized,0)
pwelch(signal_quantized,[],[],[],2);
```

```
snr_quantized = 16.9212
```

In such a case, it may be interesting to whiten the quantization error by adding real noise to it. This operation is termed as dithering.

Let us add a dithering noise with triangular pdf in the range $[-q, +q]$, i.e., two times the quantization step. Such a noise is easily obtained by adding two uniform white noises in the range $[-q/2, q/2]$.

```
dither=(rand(size(signal))-0.5)*q+(rand(size(signal))-0.5)*q;
signal_dithered=signal+dither;
```

The resulting quantization error looks indeed more noise-like (Fig. 2.14)[8].

```
signal_dithered_quantized=floor(signal_dithered/q)*q+q/2;
error_dithered_quantized=signal-signal_dithered_quantized;

plot(signal(1:50),'-.'); hold on;
plot(error_dithered_quantized(1:50));hold off
```

[8] Note that we do not account for quantizer saturation, which may occur when dither is added to a signal that approaches the saturation levels of the quantizer. This is not the case in our example.

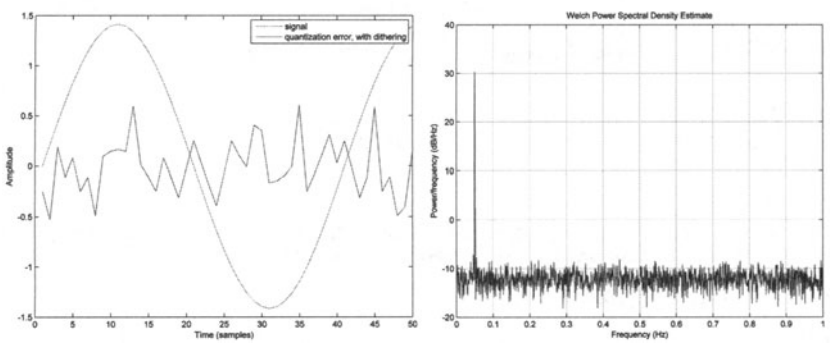

Fig. 2.14 Quantization error (*left*) and power spectral density of the quantized signal (*right*) with dithering

The power spectral density of the quantized signal now appears as white noise added to the initial sine wave.

```
pwelch(signal_dithered_quantized,[],[],[],2);
```

Dithering clearly degrades the SNR (by about 4.8 dB) but results in perceptually more acceptable quantization error.

```
load_factor=quantizer_saturation/signal_std;
snr_theory=6.02*N + 4.77 - 20*log10(load_factor)
snr_dithered_quantized=snr(signal,signal_dithered_quantized,0)
```

```
snr_theory =  16.8094

snr_dithered_quantized =  12.0562
```

2.2.3 Conventional DAC

A conventional DAC first creates a staircase signal from the sequence of samples by converting each sample into an analog voltage and holding this voltage for $1/F_s$ seconds. This is called zero-order (analog) interpolation. This operation is critical: the higher the number of bits in the PCM code to convert, the higher the precision required for creating the staircase signal!

We will demonstrate this on a sine wave with N=6 bits.

MATLAB function involved:

- `signal_quantized = uquantize(signal,N,saturation)` quantizes `signal` uniformly to `N` bits in the range [–saturation, +saturation], using a

mid-rise quantizer and triangular dither with a range of twice the quantization step. Clipping is performed when saturation is reached.

```
signal_std=1;
peak=sqrt(signal_std*2);
signal=peak*sin(2*pi*200*(0:1/8000:2));

N=3;
quantizer_saturation=2;
signal_quantized=uquantize(signal,N,quantizer_saturation);
snr_quantized=snr(signal,signal_quantized,0)
```

```
snr_quantized = 12.0809
```

We can *simulate* the analog zero-order interpolation in the digital domain by working with a much higher sampling frequency than F_s (say, F_s'=10 F_s, i.e., 80 kHz). Zero-order interpolation is then equivalent to inserting nine zeros in between each sample and convolving the resulting signal with a sequence of 10 unity samples (Fig. 2.15).[9]

```
signal_pulses=zeros(1,10*length(signal_quantized));
signal_pulses(1:10:10*length(signal))=…
                                    signal_quantized;
hold_impresp=ones(1,10);
signal_staircase=conv(signal_pulses,hold_impresp);

plot(1:10:391, signal_quantized(1:40),'o'); hold on;
plot(signal_staircase(1:400)); hold off;
```

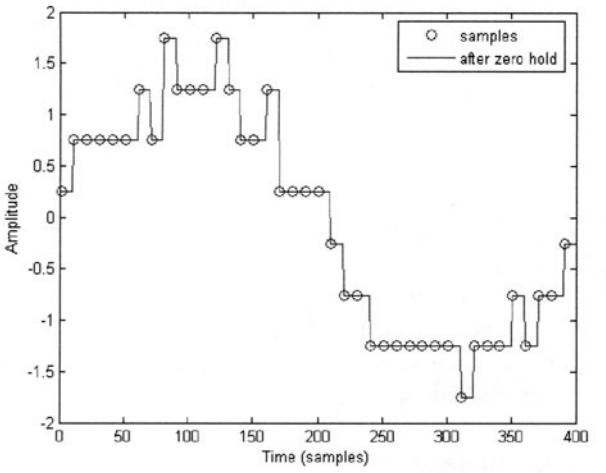

Fig. 2.15 Analog staircase signal after zero-order interpolation

[9] This is actually a very rudimentary form of numerical interpolation.

Note that zero-order interpolation acts as a low-pass filter performing a first attenuation of the spectral images of the signal at integer multiples of F_s. This directly affects the spectrum of the resulting "analog" signal. One can clearly see spectral images weighted by the effect of the interpolation (Fig. 2.16).

```
freqz(hold_impresp,1, 256, 80000);
pwelch(signal_staircase,[],[],[],20);
```

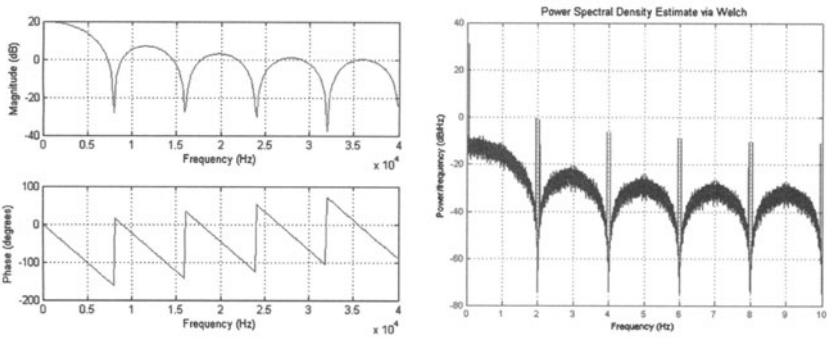

Fig. 2.16 Frequency response of the zero-order analog interpolation (*left*) and power spectral density of the resulting staircase signal (*right*)

The DAC then feeds the resulting staircase signal to an analog low-pass smoothing filter, which removes the spectral images due to sampling. The passband of this filter is limited by the maximum frequency of the signal, F_m, and its stopband must not be greater than F_s–F_m. Let us assume that the PCM signal we have is a telephone signal, with F_m=3,400 Hz and F_s=8,000 Hz, and perform the approximation of the filter with 0.1 dB of ripple in the passband and –60 dB in the stopband. We use Chebyshev approximation so as to keep the order of the filter low.

```
[order,Wn]=cheb1ord(2*pi*3400,2*pi*4600,0.1,60,'s');
[Num_LP,Den_LP]=cheby1(order,0.1,Wn,'s');
zplane(Num_LP,Den_LP);
```

The frequency response of this 12th-order filter meets our requirements (Fig. 2.17).[10]

```
freqs(Num_LP,Den_LP);
```

[10] Note that its phase is nonlinear, but for audio applications this is not a problem.

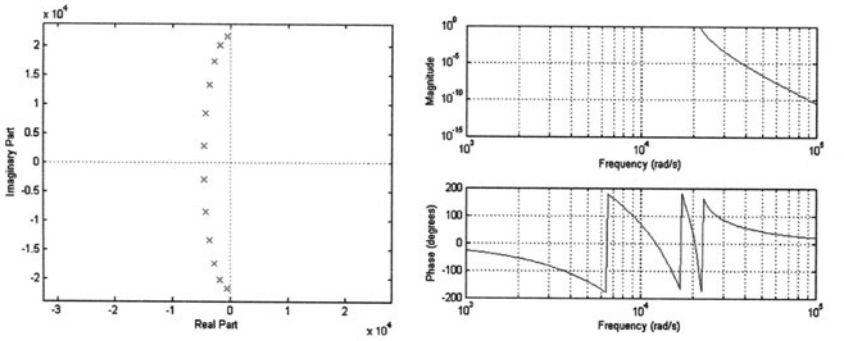

Fig. 2.17 Pole-zero plot and frequency response of the analog smoothing filter

We can now simulate analog filtering by convolving our highly oversampled staircase signal with a sampled version of the impulse response of the LP filter (Fig. 2.18).

MATLAB function involved:

- `y = filters(N,D,x,Fs)` simulates analog filtering of the data in vector `x` with the filter N(s)/D(s) described by vectors `N` and `D` to create the filtered data `y`. `Fs` is the sampling frequency of the input (and output). Filtering is performed by convolving the input with an estimate of the impulse response of the filter obtained by partial fraction expansion.

```
analog_output=filters(Num_LP,Den_LP,signal_staircase,80000);
plot(analog_output(1000:2000));
```

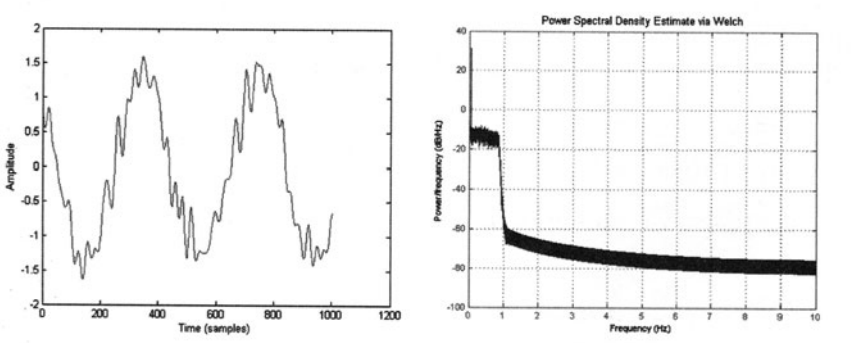

Fig. 2.18 Analog signal produced by the traditional DAC (*left*) and corresponding power spectral density (*right*)

The noise level has been reduced by the analog LP filter, as revealed by spectral analysis (Fig. 2.18). The final SNR is a bit higher than what we had before D/A conversion, but this is due to the fact that the zero-order interpolation attenuates the noise at the upper edge of its passband.

Note, though, that the samples sent to the MATLAB's `soundsc` function are made available to your ears by ... another (real) DAC. One should therefore consider the final quality of this audio sample with care.

```
pwelch(analog_output,[],[],[],20);
signal_sampled_at_10Fs=peak*sin(2*pi*200*(0:1/80000:2));
snr_analog =snr(signal_sampled_at_10Fs,analog_output,400)
                                        % 400=max possible delay
soundsc(analog_output,80000);
```

```
snr_analog =  13.5954
```

2.2.4 Oversampling DAC

By oversampling the digital signal prior to sending it to the analog part of the DAC (which is responsible for creating a staircase analog signal before final analog low-pass filtering), we can broaden the transition band of the analog smoothing filter, thereby strongly decreasing its order (and hardware complexity).

What is more, if we requantize the signal on N–1 bits (1 bit less, i.e., 6 more decibels of total noise power) after multiplying F_s by 4, only one quarter of the resulting power spectral density will contribute to quantization noise in the [0, F_s/2] band; the audible part of the requantization noise power will thus be 10 $\log_{10}$(1/4)dB (i.e., 6 dB) lower than its total power. Hence, this N–1 requantization step will be felt as a new N-bit quantization.

In other words, dropping one bit from four times oversampled digital signals (or k bits for after oversampling by a ratio of k*4) does not do much harm to the signal, while it decreases the required precision of the hardware DAC

Let us check this on our sinusoidal test signal quantized to 6 bits.

```
signal_std=1;
peak=sqrt(signal_std*2);
signal=peak*sin(2*pi*200*(0:1/8000:2));

N=6;
quantizer_saturation=2;
signal_quantized=uquantize(signal,N,quantizer_saturation);

pwelch(signal_quantized,[],[],[],2);
snr_quantized=snr(signal,signal_quantized,0)
```

```
snr_quantized =  30.1654
```

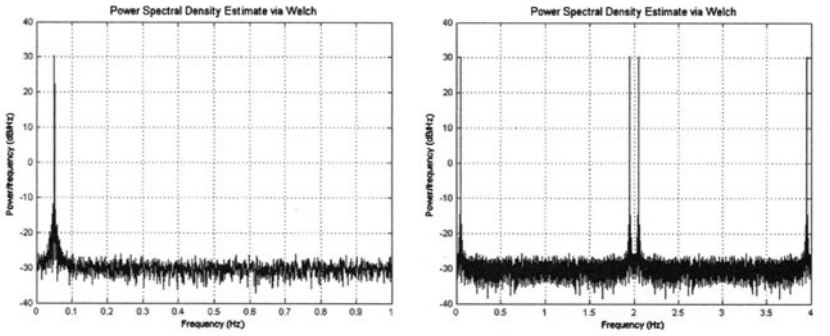

Fig. 2.19 Power spectral density of a sine wave quantized on 6 bits before (*left*) and after four times upsampling (*right*)

We now upsample this quantized signal by a factor of 4 by adding 3 zeros in between its samples. The resulting signal has a sampling frequency F_s'=32,000 Hz (Fig. 2.19). The amplification of samples by 4 is required for the final sine wave to have the same peak-to-peak level as the original.

```
signal_pulses=zeros(1,4*length(signal_quantized));
signal_pulses(1:4:4*length(signal_quantized))=4*signal_quantiz
ed;

pwelch(signal_pulses,[],[],[],8);
```

Interpolation is performed by filtering the impulses with a quarter-band digital LP filter (Fig. 2.20). Note that we use here a very sharp filter, which requires a length of 300 coefficients. In oversampling DACs, LP filtering is performed much more crudely for meeting low computational load constraints.

```
lp_fir=firpm(300,[0 .24 .26 1],[1 1 0 0]);
                % Linear phase quarter-band FIR filter
signal_interpolated=filter(lp_fir,1,signal_pulses);

plot(signal_interpolated(1000:1400));
```

The variances of the signal and of the quantization noise (hence the SNR) have not changed.

```
pwelch(signal_interpolated,[],[],[],8);
signal_sampled_at_4Fs=peak*sin(2*pi*200*(0:1/32000:2));
snr_interpolated=snr(signal_sampled_at_4Fs,…
                     signal_interpolated,300)
```

snr_interpolated = 30.1699

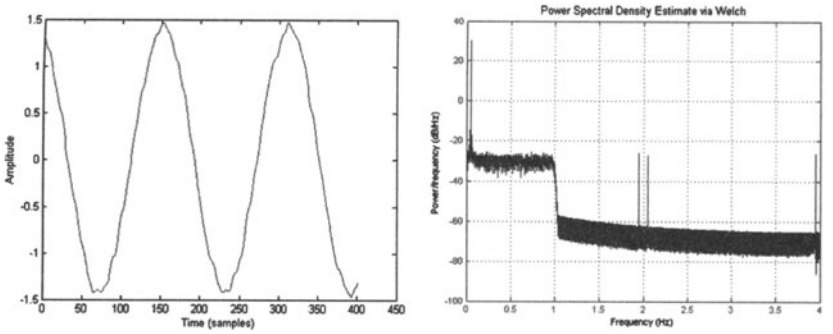

Fig. 2.20 Sine wave after upsampling and interpolation: time domain (*left*) and power spectral density (*right*).

Here comes the first positive effect of oversampling on SNR; (re)quantizing the interpolated signal with N–1 bits (i.e., omitting the least significant bit of the underlying PCM codes) does not much affect the SNR.[11]

Let us first perform 4-bit requantization of the interpolated signal (Fig. 2.21).

```
signal_requantized=uquantize(signal_interpolated,4,…
                             quantizer_saturation);
plot(signal_sampled_at_4Fs(2000:2400)); hold on;
plot(signal_requantized(2151:2550)); hold off; [12]
```

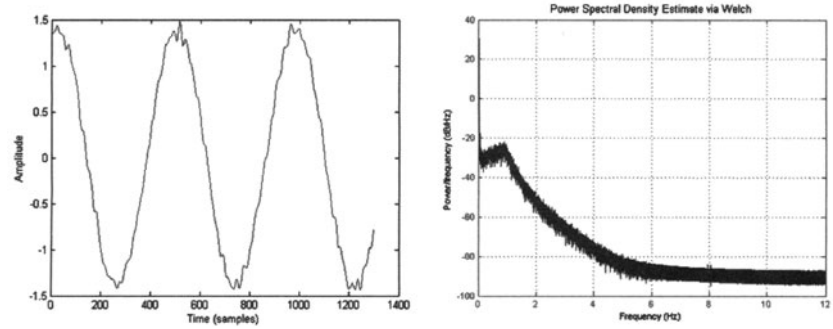

Fig. 2.21 Sine wave after requantization on 4 bits at $4F_s$: time domain (*left*) and power spectral density (*right*)

[11] At worst it could degrade the SNR by 3 dB.

[12] Note that we have to shift the oversampled signal when comparing it to the requantized signal; this is due to the delay of the interpolation filter.

The PSD of the requantized signal shows noise with a variance of about 12 dB higher than its initial value, but only one-fourth of that noise is in the band [0, $F_s/2$]. Therefore, the measured SNR does not reflect the actual SNR. We will see below that the actual SNR has decreased only by about 6 dB.

```
pwelch(signal_requantized,[],[],[],8);
snr_requantized=snr(signal_sampled_at_4Fs,…
                    signal_requantized,300)
```

```
snr_requantized =  17.8152
```

Now we can again *simulate* the analog part of D/A conversion (as in Section 2.2.3) using an "analog sampling frequency" of 3*32=96 kHz (Fig. 2.22). Oversampling has a second positive effect on this part: it allows for a much wider transition band for the low-pass smoothing filter: [3,400 Hz, 28,600 Hz]. This results in a simpler filter, of order 4.[13]

```
[order,Wn]=cheb1ord(2*pi*3400,2*pi*28600,0.1,60,'s');
[Num_LP,Den_LP]=cheby1(order,0.1,Wn,'s'); % relaxed filter
signal_pulses=zeros(1,3*length(signal_requantized));
signal_pulses(1:3:3*length(signal_requantized))=…
                                        signal_requantized;
hold_impresp=ones(1,3);
signal_staircase=conv(signal_pulses,hold_impresp);
analog_output=filters(Num_LP,Den_LP,signal_staircase,96000);

plot(analog_output(2000:3300));
pwelch(analog_output,[],[],[],24);
```

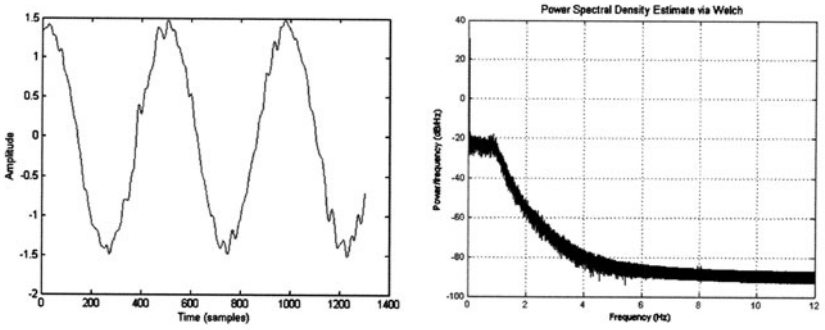

Fig. 2.22 Analog signal produced by the oversampling DAC: time domain (*left*) and power spectral density (*right*)

[13] We use Chebyshev approximation here. For some applications, a Bessel filter could be preferred, as it does not imply important phase nonlinearity. For our sine-based proof of concept, it makes no difference.

As a result of the analog filtering, the apparent SNR is now about 7 dB lower than that of our initial 6-bit quantization, i.e., only 1 dB less than 5-bit quantization at 8 kHz. The lost decibel comes from dithering, and from the fact that the LP filter specifications are loose.

```
signal_sampled_at_12Fs=peak*sin(2*pi*200*(0:1/96000:2));
snr_analog=snr(signal_sampled_at_12Fs,analog_output,4000)
```

```
snr_analog = 22.9805
```

In summary, a 6-bit DAC operating at F_s leads to the same SNR as a 5-bit DAC operating at the 4× oversampling rate $4*F_s$.

2.2.5 Oversampling and noise-shaping DAC

We will now show that noise shaping makes it possible to produce 4-bit quantization at $4*F_s$ with the same resolution as 6-bit quantization at F_s.

We start by quantizing to 6 bits and oversampling by 4.

```
signal_std=1;
peak=sqrt(signal_std*2);
signal=peak*sin(2*pi*200*(0:1/8000:2));
%quantization
signal_quantized=uquantize(signal,6,2);
% Oversampling by 4
signal_pulses=zeros(1,4*length(signal_quantized));
signal_pulses(1:4:4*length(signal_quantized))=…
                                        signal_quantized;

lp_fir=firpm(300,[0 .24 .26 1],[1 1 0 0]);
signal_interpolated=filter(lp_fir,1,4*signal_pulses);
```

Let us now perform 4-bit requantization of the interpolated signal with noise shaping as shown in Fig. 2.6.

```
N=4;
quantizer_saturation=2;
q=2*quantizer_saturation/2^N;

delay_memory=0;
signal_requantized=zeros(size(signal_interpolated));
for i=1:length(signal_interpolated)
    u=signal_interpolated(i)+delay_memory;
    % quantization, including dithering
    signal_requantized(i)=uquantize(u,N,quantizer_saturation);
    delay_memory=u-signal_requantized(i);
end;

signal_sampled_at_4Fs=peak*sin(2*pi*200*(0:1/32000:2));
plot(signal_sampled_at_4Fs(2000:2400)); hold on;
plot(signal_requantized(2151:2550)); hold off;
```

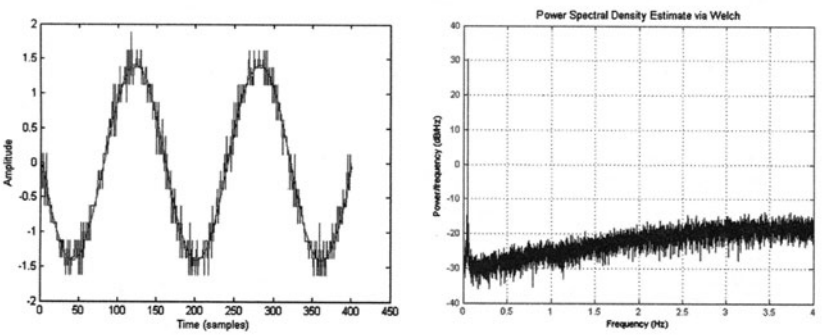

Fig. 2.23 Sine wave after requantization to 4 bits at $4F_s$ using noise shaping: time domain (*left*) and power spectral density (*right*)

The PSD of the requantized signal shows colored quantization noise with most of its energy above $F_s/2$ as a result of noise shaping (Fig. 2.23). Therefore, the measured SNR does not reflect the actual SNR.

```
pwelch(signal_requantized,[],[],[],8);
snr_requantized=snr(signal_sampled_at_4Fs,…
                    signal_requantized,300)
```

```
snr_requantized =  14.8699
```

Again, we can *simulate* the analog part of D/A conversion (Fig. 2.24).

```
signal_pulses=zeros(1,3*length(signal_requantized));
signal_pulses(1:3:3*length(signal_requantized))=…
                                        signal_requantized;
hold_impresp=ones(1,3);
signal_staircase=conv(signal_pulses,hold_impresp);

[order,Wn]=cheb1ord(2*pi*3400,2*pi*28600,0.1,60,'s');
[Num_LP,Den_LP]=cheby1(order,0.1,Wn,'s'); % relaxed filter

analog_output=filters(Num_LP,Den_LP,signal_staircase,96000);

plot(analog_output(2000:3300));
```

As a result of analog filtering, the apparent SNR is now only 3 dB lower than that of our initial 6-bit quantization, i.e., only 3 dB less than 4-bit quantization at 8 kHz. Again, the lost decibels come from dithering and from the fact that the LP filter specifications are loose.

```
pwelch(analog_output,[],[],[],24);
signal_sampled_at_12Fs=peak*sin(2*pi*200*(0:1/96000:2));
snr_analog=snr(signal_sampled_at_12Fs,analog_output,4000)
soundsc(analog_output,96000);
```

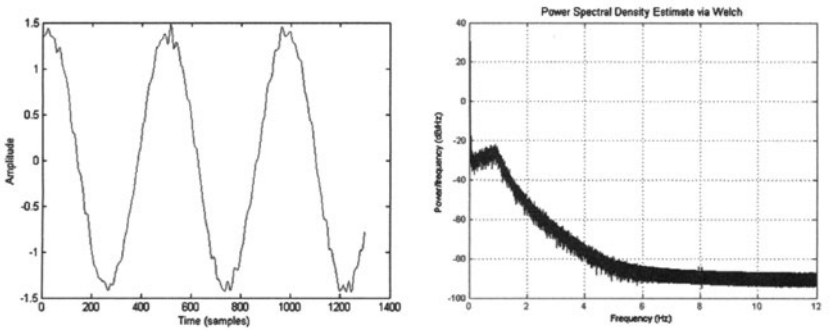

Fig. 2.24 Analog signal produced by the oversampling DAC with noise shaping: time domain (*left*) and power spectral density (*right*)

```
snr_analog =      27.0433
```

2.2.6 Delta–sigma DAC

The principle of delta–sigma modulation is to apply noise shaping and oversampling in such a way that the signal ends up being quantized on 1 bit. This considerably alleviates the task of the hardware D/A converter (which reduces to a switch) and puts most of the load in the digital domain.

In this proof of concept, we will perform 6-to-1 bit requantization using an oversampling ratio of 32 (2^5) and a first-order delta–sigma modulator. Real delta–sigma DACs use higher-order modulators or cascades of first-order modulators and thus do not have to implement oversampling of ratio 2^15!

We start by quantizing to 6 bits and oversampling by 32.[14]

```
signal_std=1;
peak=sqrt(signal_std*2);
signal=peak*sin(2*pi*200*(0:1/8000:1/8));

%quantization
N=6;
quantizer_saturation=3;
signal_quantized=uquantize(signal,N,quantizer_saturation);
snr_quantized=snr(signal,signal_quantized,0)

% Oversampling by 32
signal_pulses=zeros(1,32*length(signal_quantized));
```

[14] We do it here with a single interpolation filter, for convenience. In practice, it would be more computationally efficient to implement oversampling as a cascade of two intermediate oversampling blocks, as mentioned in Section 2.1.5.

```
signal_pulses(1:32:32*length(signal_quantized))=...
   signal_quantized*32;
lp_fir=firpm(300,[0 1/32-1/100 1/32+1/100 1],[1 1 0 0]);
signal_interpolated=filter(lp_fir,1,signal_pulses);
```

```
snr_quantized = 26.6606
```

Let us now perform 1-bit requantization with noise shaping. The resulting quantized signal is purely binary (Fig. 2.25).

```
N=1;
quantizer_saturation=3;
q=2*quantizer_saturation/2^N;

dither=(rand(size(signal_interpolated))-0.5)*q...
      +(rand(size(signal_interpolated))-0.5)*q;

signal_requantized=zeros(size(signal_interpolated));
delay_memory=0;
for i=1:length(signal_interpolated)
    u=signal_interpolated(i)-delay_memory;
    % quantization, including dithering
    signal_requantized(i)=uquantize(u,N,quantizer_saturation);
    % saving the internal variable
    delay_memory=signal_requantized(i)-u;
end;

signal_sampled_at_32Fs=peak*sin(2*pi*200*(0:1/256000:1/8));
plot(signal_sampled_at_32Fs(2000:2600)); hold on;
plot(signal_requantized(2151:2750)); hold off;
xlabel('Time (samples)'); ylabel('Amplitude');
pwelch(signal_requantized,[],[],[],64);
```

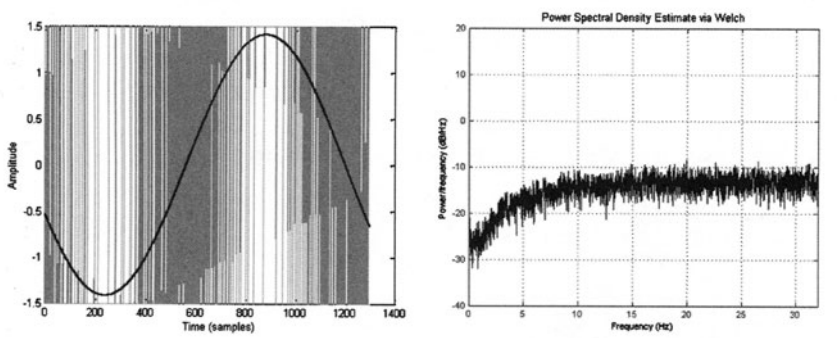

Fig. 2.25 Sine wave after requantization on 1 bit at $32F_s$ using noise shaping: time domain (*left*, superimposed with the original sine wave) and power spectral density (*right*)

Again, we can *simulate* the analog part of D/A conversion (Fig. 2.26). We use an "analog sampling frequency" equal to the one we have reached after 32 times interpolation.

```
[order,Wn]=cheb1ord(2*pi*3400,2*pi*28600,0.1,60,'s');
[Num_LP,Den_LP]=cheby1(order,0.1,Wn,'s'); % relaxed filter
zplane(Num_LP,Den_LP);

analog_output=filters(Num_LP,Den_LP,signal_requantized,…
                      256000);

plot(analog_output(2000:3300));
```

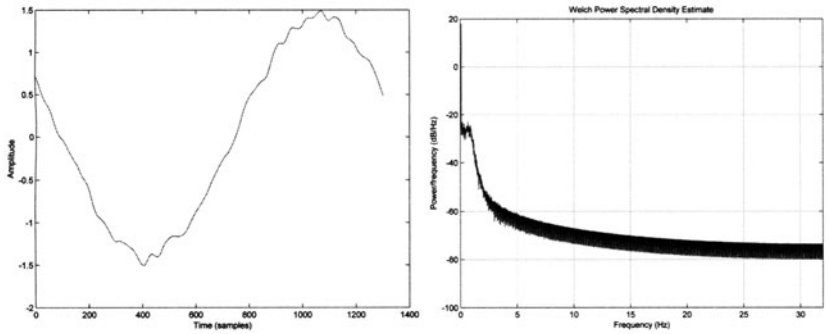

Fig. 2.26 Analog signal produced by the delta–sigma DAC: time domain (*left*) and power spectral density (*right*)

As a result of analog filtering, the apparent SNR is now very close to that of our initial 6-bit quantization! This is confirmed by listening.

```
pwelch(analog_output,[],[],[],64);
snr_analog=snr(signal_sampled_at_32Fs,analog_output,4000)
soundsc(analog_output,256000);
```

```
snr_analog =  25.4677
```

This technique is used in most CD players today[15], for requantizing 16-bits samples to 1 bit, hence with much higher oversampling ratios. Interpolation is therefore performed in several steps for keeping the interpolation filters as simple as possible. Delta–sigma is also the heart of the DSD (direct stream digital) coding used in super audio CDs (SACDs), in which a 1-bit stream is created by the ADC, stored on the CD, and directly converted to sound by the DAC.

[15] Note that all the filters used in this proof of concept use floating-point arithmetics. Using fixed-point arithmetics is more complex.

2.3 Going further

A very simple interactive tutorial on delta–sigma analog-to-digital conversion is available from Analog Devices (2007).

The unescapable reference in delta–sigma modulators is Schreier and Temes (2005). This is a companion book to R. Schreier's MATLAB Delta–Sigma toolbox (Schreier 2003).

2.4 Conclusion

As mentioned in the introduction of this chapter, efficient DACs add noise to the PCM signal in three ways. First, they use dithering to avoid having quantization errors correlated with the input signal. Second, they requantize the signal at a higher sampling rate, which can be seen as adding a second quantization noise. Third, they use noise shaping, which increases the overall quantization noise power but pushes most of it outside the useful band. As a result, and rather unexpectedly, 1-bit DACs are synonymous with high quality.

Besides, 1-bit sigma–delta conversion is the basis of the direct stream digital, a technology used to store audio signals in a digital format, which is used in super audio CD (SACD), the high-resolution CD audio format trademarked by Philips and Sony (Reefman and Janssen 2004). In this technique (Fig. 2.27), the audio waveform for each channel is fed to an analog delta–sigma modulator, which directly samples the signal with a sampling frequency 64 times higher than required and quantizes it to 1 bit. This 82 Mbps-bit stream (to be compared to the 705-kbps bit stream of the classical CD) is recorded on the SACD (it is not low-pass filtered). The SACD reader therefore reduces to a 1-bit DAC producing a bipolar signal, followed by the smoothing filter of an oversampling DAC (i.e., with relaxed constraints).

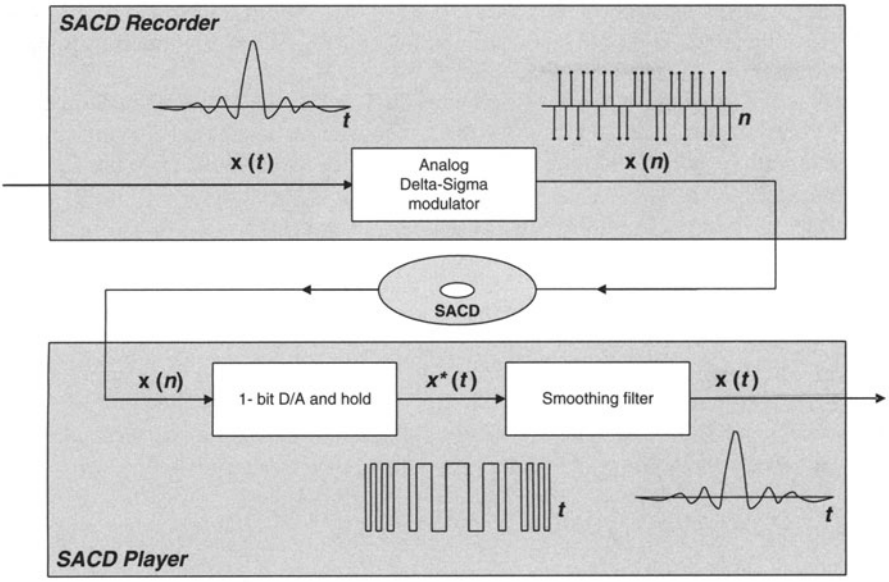

Fig. 2.27 The super audio CD concept

Last but not least, the pulse density modulated (PDM) available after the 1-bit DAC in a DSD player ($x^*(t)$ in Fig. 2.27) can directly be amplified by transistors switching from full ON to full OFF, thereby implementing a fully digital *Class D* power amplifier (Gaalaas 2006).

References

Analog Devices (2007) Interactive Design Tools: Sigma-Delta Analog-to-Digital Converters: Sigma–Delta ADC Tutorial [online] Available: http://www.analog.com/ [19/3/2007]

Candy JC (1997) An overview of basic concepts. In: Norsworthy, Schreier, and Temes (eds) Delta-Sigma Data Converters. IEEE Press: Piscataway, NJ, pp 1–43

Gaalaas E (2006) Class D Audio Amplifiers: What, Why and How. Analog Dialogue 40-06 pp 1–7 [online] Available: http://www.analog.com/library/analogDialogue/archives/40-06/class_d.pdf [15/3/2007]

Hicks C (1995) The application of Dither and Noise-Shaping to Nyquist-Rate Digital Audio: An Introduction [online] Available: www.digitalsignallabs.com/noiseb.ps [15/3/2007]

Kester W, Bryant J (2003) DACs for DSP applications. In: Kester W (ed) Mixed Signal and DSP Design Techniques. Newnes: Amsterdam, pp 99–115.

Lipschitz SP, Vanderkooy J (2001) Why 1-Bit Sigma-Delta Conversion is Unsuitable for High-Quality Applications. 110th Audio Engineering Society Convention Paper 5395, Amsterdam.

Matsuya Y, Uchimura K, Iwata A, Kobayashi T, Ishikawa M, and Yoshitome TA (1987) 16-bit oversampling A-to-D conversion technology using triple-integration noise shaping. IEEE Journal of Solid-State Circuits 22-12: 921–929

Reefman D, Janssen E (2004) One-bit audio: An overview. Journal of the Audio Engineering Society 52-3:166–189

Schreier R (2003) The Delta-Sigma Toolbox [online] Available: http://www.mathworks.com/matlabcentral/ (search for "delta-sigma") [19/3/2007]

Schreier R, Temes GC (2005) Understanding Delta-Sigma Data Converters. IEEE Press: Piscataway, NJ.

Wannamaker RA (1997) The Theory of Dithered Quantization. PhD Thesis, University of Waterloo, Canada

Chapter 3

How is sound processed in an MP3 player?

T. Dutoit (°), N. Moreau (*)

(°) Faculté Polytechnique de Mons, Belgium
(*) Ecole Nationale Supérieure des Télécommunications, Paris

Mr. Audio was a bit loose in his PCM tuxedo. Tayloring the suit helped it lose bits and become a lighter MP3 dinner jacket.

In his 1929 painting "La trahison des images," Belgian painter René Magritte highlighted the power of illusions by painting a pipe and commenting it with "Ceci n'est pas une pipe" ("This is not a pipe") as Magritte himself explained: "Try to stuff the painting with tobacco... ."

Perceptual illusions have been used by engineers for designing many consumer products. As an example, one of the most extraordinary imperfections of our visual process, known today as the phi effect (and mistaken for a long time with the persistence of vision), has induced the choice for 24 images per second in the movie pictures industry. In this chapter, we will see how the limitations of the human *auditory* process have been taken into account for the design of *lossy but transparent* perceptual audio coders. In particular, we will focus on the principles underlying the MPEG-1 Layer-I audio coding norm.

3.1 Background – Sub-band and transform coding

Sub-band signal processing is very useful in audio (as well as in video) coding applications. The sub-band encoder (Fig. 3.1) decomposes an input

T. Dutoit, F. Marqués (eds.), *Applied Signal Processing*,
DOI 10.1007/978-0-387-74535-0_3,