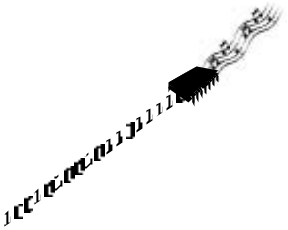
# Jitter and digital audio

## Contents



- ΣΔ introduction
- ΣΔ coding and properties
- Analog reconstruction
- Influence of jitter in time- and freq domain
- Means of reducing jitter sensitivity
- Dealing with the remaining jitter
- Conclusions

*Remco Stoutjesdijk*

*UltrAnalog / University of Twente / Philips Electronics N.V.*



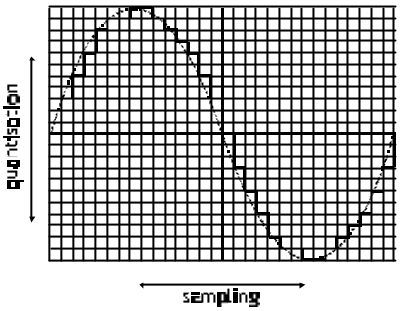
# Notes slide 1

Well, this speaks for itself.

# ΣΔ coding and properties I

## RedBook PCM

- 16 bits linear code, 44.1 kHz sampling frequency
- constraints: 2 kHz filtering bandwidth
- tight constraints on analog IC components (current sources, switches etc)
- Current output very sensitive to jitter






# Notes slide 2

Let's start out with a general overview of digital audio formats.

The first commercial digital audio standard was PCM (Pulse Code Modulation). PCM is a linear code: every sample is described by a codeword, every bit of which weighs half as much as the previous bit. There are 16 bits, making up for 65536 possible discrete sample values. Per second, 44100 samples are taken. The Nyquist criterion dictates now the maximum digitizeable frequency is 22010 Hz. After that, mirror images of the original baseband signal will appear and it is impossible to distinguish these from the original signal. Hence, digitizing and reconstruction requires a very steep filter blocking out everything beyond fs/2. In later developments, oversampling was used to relax these constraints.

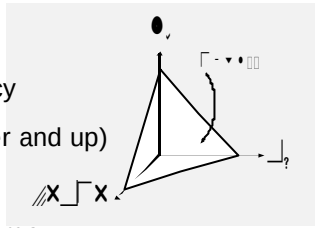
Reconstruction of a PCM signal into an analog signal requires distributing the binary weights of the bits in the code into analog values. Usually these are currents, so a PCM DAC consists of 16 current sources, each twice as small as the previous one. The maximum error allowed in the current value is 1/65535th of the total current (-96 dB) over a large range of currents. This is a darn hard job.

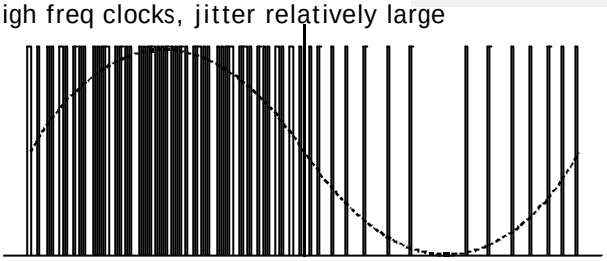
Since the current sources vary so much in size, timing is everything in a PCM DAC. Suppose a 4 bit dac with a code of 0111 that goes to 1000 in the next sample. This is a change of only 1 Least Significant Bit. However, it incorporates a switch in all of the 4 current sources. If they don't all switch exactly simultaneously, the code may go from 0111 to 1111 to 1000, for instance. That is, translated into analog, a huge spike in the signal, and an extremely fast one at that, hard to filter. This is one of the main reasons jitter has received so much attention in the PCM world.

# ΣΔ coding and properties II

ΣΔ coding

- 1 bit, 2.822 MHz sampling frequency
- constraints: noise shaping (3rd order and up)
- high PSU susceptibility
- high freq clocks, jitter relatively large







# Notes slide 3

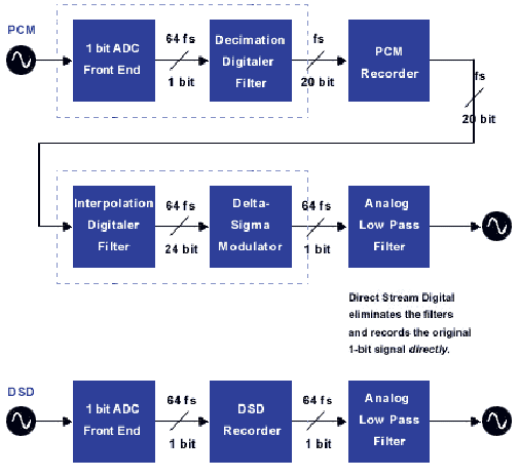
A completely different way of digitally describing audio is sigma/delta modulation. This is also called delta/sigma, an ever ongoing dispute that makes no sense. The sigma and delta denominate two elements in the modulator which work in a loop, so which one you name first depends just on where you board the loop. That's all.

Sigma/delta was developed as a way of masquerading noise in PCM DACs when it was yet impossible to achieve full 16 bit linearity, but started to lead a life of its own when it matured. In its most extreme and (nowadays) common form, the coder uses just 1 bit. 1 bit equals 2 possible levels: above half, or below half. This is a rather crude approximation of the delicate analog signal and yields a large quantisation error. However, the trick in a sigma/delta modulator consists of feeding that error back and deducting it from the next sample. If you do that fast enough, no one will notice. And that's what sigma/delta modulators do: they run extremely fast. 64*44100 Hz, or 2.8 MHz, is a minimum these days. In the time PCM DACs take to process one sample, sigma/delta dacs do at least 16 samples and in doing so, they can average out the error.

Jitter, which is a deviation in the clock timing, can possibly do a lot of harm in sigma/delta DACs, because of the simple fact that they run much faster. Any deviation seems larger if the clock cycle is smaller.

# Analog reconstruction

- ΣΔ bears very close resemblance to analog signal
- Very large distance between passband and stopband
- Consequence: filtering & buffering






# Notes slide 4

Why have sigma/delta DACs become so popular, so fast? There are two reasons. First, having only 2 levels, they require less analog precision which is advantageous (cheap) to chip manufacturers. The trade-off, a very high switching speed, is of no importance anymore in the GHz era. Secondly, the analog reconstruction is very easy to implement. Since the DAC has such a very high speed, the bandwidth of the analog signal is much larger and hence, it requires less and easier filtering.

One downside of the two levels is a high PSU susceptibility, but on the other hand, a differential output is easily obtained by inverting the bitstream, and the differential output has a good PSRR (see the CD enhancer II for details).

Sigma/delta describes the signal by high/low pulses. Because of the oversampling and feedback nature of the DAC, you can regard a high level as an attempt to raise the output signal and a low level as an attempt to lower the output. So the DAC describes the signal as “UP, UP, DOWN, UP, DOWN, DOWN, UP, DOWN, UP, UP, DOWN, …” etc. It is easy to envision that zero, or in fact any DC signal, would be described by an alternating UP and DOWN, averaging to no change. So, to retrieve the output signal, we have to average the pulse stream. Mathematically, that is exactly what low-pass filtering does. The amount of pulses thus equals the analog amplitude. This is called Pulse Density Modulation.

In this respect, the digital data stream is already half analog: the area that is in a pulse equals the amount of energy that the analog signal should contain. Compare this to the PCM data, which is just a stream of numbers with no analog information. This is why all the sigma/delta manufacturers (for instance the SACD consortium) will tell you 1 bit is more analog than PCM, and they’re right.

# Jitter in ΣΔ: time domain

- ΣΔ = pulse density modulation
- Average energy content = analog information
- Influence of jitter twofold:
  - energy content alteration -> DNL degradation
  - intersymbol interference -> data dependency
- When data dependent:
  - properties of coder and filter determine result
  - little known -> WCS / BCS

<t>



# Notes slide 5

However, if we encounter jitter, and this jitter widens or narrows one of the pulses, the analog energy content is directly affected. Not so good. It can be even worse: jitter never comes alone. If one pulse comes too late due to jitter, there is a good chance the next pulse will come too soon since the mean of the jitter is zero. If the first pulse was a '1' and the second was a '0', there is a) not enough 'UP' info since the pulse was too short and b) too much 'DOWN' info since the second pulse was too long. The effect of the jitter has been doubled! However, in a similar fashion, you can imagine that a stream of 2 subsequent '1' pulses will tend to cancel out the jitter.

So, what do we have here? Either the jitter effect is doubled or it is canceled out, depending on the actual data. This is a mindbreaker for the mathematicians. Because of the noise shaping feedback in the sigma/delta modulator there is a large amount of chaos in the bitstream. In fact, the higher the chaos, the better the modulator is regarded since predictable data streams can cause unwanted idle tones (called limit cycles). However, writing out an equation relating the bitstream form and measurable quantities such as harmonic distortion is nearly impossible. All one can do is write down the worst and best case scenarios I described above.

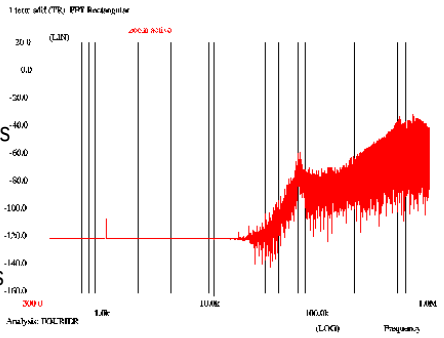
The other way jitter can influence the sigma/delta converter is by degrading InterSymbol Interference (ISI).

The rising and falling edges of a pulse are almost never equal. For instance, the rising is performed by PMOST devices and the falling is performed by NMOST devices, which are never completely complementary due to the difference in electron and hole mobilities. Since the slopes of rising and falling are only infinitely steep in the textbooks, this creates small amplitude errors when the timing is a bit off (i.e. Jitter). Now, envision datastream 1: 1-0-1-0 and datastream 2: 1-1-0-0. Both should have the same energy. Datastream 1 encounters the error 2 times, but datastream 2 encounters it only once. The difference is data dependent again: two subsequent 1's are different than two independent 1's.

# Jitter in ΣΔ: freq. domain

## Pulse Density Modulation:

- average frequency = n*fs (n >= 64)
- high order modulators: broad band, no limit cycles
- effects of jitter comparable to those of dither
- marginally stable modulators may destabilise





# Notes slide 6

Not knowing exactly how the DAC is affected does not mean we have no means of countering jitter influence, however. This will be described in the subsequent section.

But first: a quick look at the frequency behavior of the sigma/delta DAC.

The sampling frequency is very high, in this case it is 2.8 MHz (64 times oversampling). This implies that at 1.4 MHz (fs/2) the entire band starts repeating itself. The noise above 20 kHz is the noise that is created by the crude 1 bit quantisation. The sigma/delta modulator has shifted that noise to supersonic frequencies where they are filterable (noise shaping).

Now the main question: how can jitter harm us here? Well, in the frequency domain you won't see much of jitter. It cannot create harmonic distortion since the datastream is so random. It may cause the noise floor to rise since jitter adds to the total noise. It won't create HF spikes, au contraire: if a badly designed sigma/delta modulator were exhibiting a predicable data pattern (for instance, 10101010101010), the jitter would disrupt it, causing the spike to disappear much in the same way as the usual triangular probability density function added noise does (aka dither).

The only possible harm jitter could do, is to cause a sigma/delta modulator to destabilize, but then it would already have to be at the limits of stability, which means it is a badly designed one, which deserves it.

# Reducing jitter influence I

Switched Capacitor data signaling:

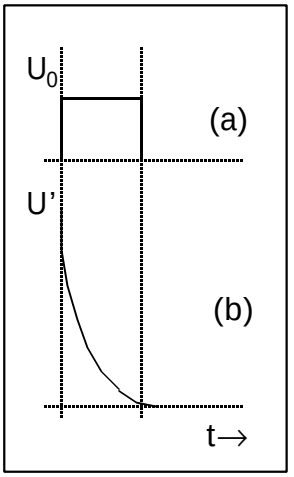
- eliminate one flank
- eliminate ISI
- high amount of HF energy
- implemented in DAC7

Square pulse:

$$\Delta A = U_o \Delta t$$

Switched capacitor pulse:

$$\Delta A = U_0 \frac{t_0}{\left(e^{+at_0} - 1\right)}\left(e^{-a\Delta t} - 1\right)$$





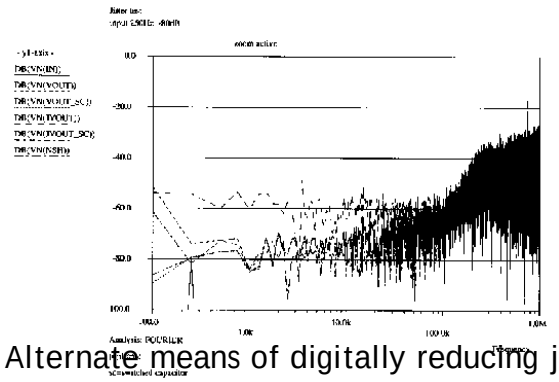

# Notes slide 7

Now we have investigated jitter in SD modulators, we are not very reassured by the findings. It may work, it may not… that's not something you can afford being a manufacturer.

One possible way of reducing jitter is by changing the form of the pulses. Instead of square pulses, we could make them exponentially shaped. The advantage lies in the fact that the pulse now only has one flank. If a timing error were to occur, it would maybe displace the first flank somewhat, but that's not so important at these speeds. The second flank, which causes a linear offset in a square form, now causes only an exponentially lowered offset, thus largely nullifiying the effect.

How can we create an exponentially shaped pulse? Well, for instance, by the discharge of a capacitor through a resistor. This technique works excellent, and has been used in the famous TDA1547 (DAC7) filter for instance. It does have its downside though: the pulse now looks more like a spike. In effect, it creates a lot of HF pollution which is hard to get rid off on chip. This is the main reason this technique has never become widely accepted.

# Reducing jitter influence II



Alternate means of digitally reducing jitter and ISI:

- RZ coding
- Manchester coding



# Notes slide 8

Here is a very crude experiment I took with a 1 bit class-D amplifier. I purposely injected massive amounts of jitter, so much that the noise floor came up to -60 dB. Then the signaling was altered to switched-cap and it directly went back to -80 dB, canceling 20 dB of jitter induced noise.

So, switched-cap signalling works on the pulse shape in an analog fashion, but has its drawbacks. There are however also means of re-writing the digital code so it's unaffected by jitter:

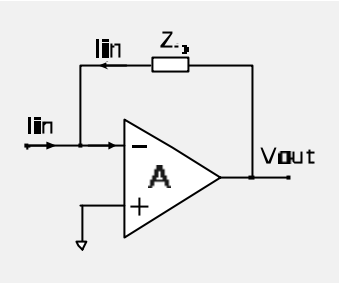
Return-to-Zero Coding: a '1' is described by a 75% high, 25% low code and a '0' is described as a 25% high, 75% low code. Now, every bit, whether '0' or '1', goes up and down once per cycle. Jitter cancels out within 1 bittime and long strings of '1' bits are no longer different from single '1' bits.

Manchester coding, a form of biphase coding, ensures that long strings of '1's alternate phase, thus reducing jitter influence by making the flank changes no longer directly data dependent.

# Dealing with the jitter

- Wideband output stages
  - deal with square shaped signals while maintaining linearity
- No opamps!
  - slew induced distortion

    = even order HD
- Passive filtering
  - linearity
  - signal level independent






# Notes slide 9

Even with the methods of jitter canceling described, there will always remain some jitter. Dealing with that is at least as important as trying to reduce jitter itself.

One of the important aspects is dealing with digital streams correctly. Square shaped signals coming out of the dac are hard to process for analog circuits. Usually, nullor configured opamps are used as in the figure above. However, during the flank, their feedback signal has not yet arrived and the opamp is running on its input stage alone. That one usually can't handle the fast transition and is slewing while charging its Miller capacitance. During slewing, it creates a triangularly shaped error, which boils down to even order harmonic distortion, not of the audio signal, but of the digital signal. That creates high frequency intermodulation, usually with higher order mirrors, which fold back into the audio band at unpredictable places and amplitudes. To be avoided!

It is wiser to use an intrinsic high bandwidth stage here, such as the common-gate configuration, which doesn't suffer from the Miller effect. For much the same reason, the usual opamp feedback based active filtering should be avoided. Active filters suffer from signal level dependent errors: if the signal is just experiencing a step input, the filtering is off. Again the result is audible degradation that is hard to predict and measure.

# Conclusions

Jitter in ΣΔ signals:

- influence larger due to higher clock freq
- non-symmetric flanks lead to inter symbol interference
- DNL and data dependency degradation

Means of dealing with jitter:

- duh- good clock
- signaling with switched cap or RZ protocol
- radical redesign of analog output stages



# Notes slide 10

So- what have we seen:

Jitter is possibly bad in SD converters, under exactly the wrong circumstances. Since we cannot exactly predict where and when this will happen, it is better to avoid the situation altogether. Either by a good low-jitter clock (which should be the starting point) or by changing the signaling to cancel out jitter. Nowadays most SD DACs use a digital jitter-canceling signaling protocol. The DAC7 is a classic example of a jitter canceling solution that does more harm than good.

Having reduced jitter as far as we can, we still have to design the output stage to deal with the remainder. This can be done by designing it for intrinsic high bandwidth and inorporating passive filtering.