XMOS

AN02004: Biquad Quantization Noise in Fixed Point DSP

Publication Date: 2024/10/16
Document Number: XM-015057-AN v1.0.0

IN THIS DOCUMENT

Efficient execution of Digital Signal Processing algorithms requires them to use *fixed-point* arithmetic. Unlike the Real Numbers used in maths, fixed-point numbers have a limited number of bits (decimal places) used to store coefficients, signal values, and intermediate results.

The difference between the ideal mathematical value and the actual value as processed in fixed-point arithmetic is known as *quantization noise*. There are many factors that can influence the level of the quantization noise, some of which that affect biquad filters are discussed below. We focus on Biquad filters as they have a feedback component in them that can amplify the noise.

1 Biquad Topology

A biquad is a second order digital filter that can be used to generate many common equalisation filters. The basic flow-graph of a direct form 1 biquad is shown in Fig. 1. This is commonly used for fixed point biquad implementations.

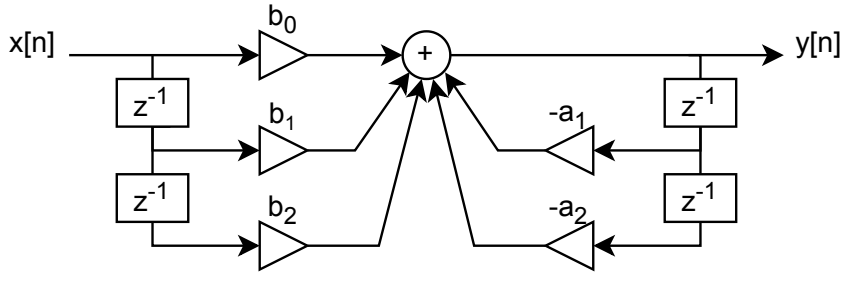

Fig. 1: Direct form 1 biquad

Multiplying an int32 input or delayed output sample by an int32 coefficient results in an int64 number, which at some point must be rounded or truncated before shifting back to an int32. The rounding or truncation causes quantization noise, and the implementation details of the filter influence the level of this noise.

2 Quantization noise vs filter frequency

We use three simple peaking biquad filters as our test case, implemented with int32 arithmetic. We use a 997 Hz sine wave as a test signal, and assume a sampling frequency of 48 KHz. We have picked 997 Hz, as it is a non-integer multiple of the sampling frequency. Quantization noise can be seen at low frequencies. As the peak of the filter frequency moves towards zero, an increase in noise is seen in Fig. 2.

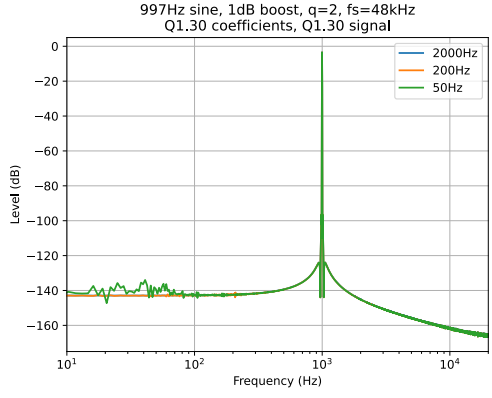

Fig. 2: Quantization noise vs filter frequency for Q1.30 coefficients and Q1.30 signal

The quantization noise originates from the bit shift required after multiplying two int32 numbers. As the filter frequency decreases, the magnitude of the filter poles increases This causes an increase in the amplification of the quantization noise. The magnitude of the poles shown in Fig. 2 are:

Frequency	Pole magnitude
2000	0.94067142
200	0.99384081
50	0.99845649

3 Fixed-point number formats

We denote fixed-point number formats as *QN.M* where *N* is the number of bits before the decimal point, and *M* is the number of bits after the decimal point. *M* and *N* will always sum to 31 for an int32, leaving one bit available for the sign bit,

We assume that the input signal is always in Q0.31; that is, it has a magnitude of at most 0.999999999534. To avoid overflow, the signal must have 1 bit of headroom for every 6 dB of boost in the filter. If we have a 1dB boost, Q1.30 format can represent the output signal without overflow, although we will have to clip our signal if outputting audio to Q0.31 format. Alternatively, we can attenuate the signal by -1dB before converting it back to Q0.31.

4 Quantization noise from number formats

Number formats have a significant influence on quantization noise. Most biquad coefficients are between -2 and +2, so we can represent them in Q1.30 format.

When more bits of headroom are required in the signal, the quantization noise increases by roughly 6dB per bit of precision lost. In Fig. 3, Q4.27 is used for the signal. This means

the full scale sine wave has a peak value of just less than 2^27, and the quantization noise increases for the 50Hz filter by around 18 dB. Additional noise can also be seen on the 200Hz filter, while the 2000Hz filter still has low quantization noise due to the lower pole gain.

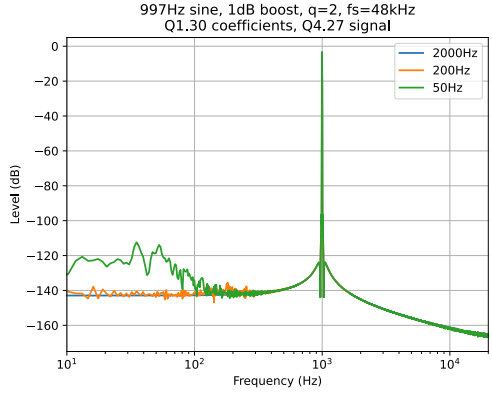

Fig. 3: Quantization noise vs filter frequency for Q1.30 coefficients and Q4.27 signal

Depending on the gain of the filter, the magnitude of its coefficients may become greater than 2, and thus not able to be represented in Q1.30 format. This can be a particular problem for shelf filters and low Q factor peaking filters, where a large portion of the frequency spectrum is amplified. This may result in additional quantization noise.

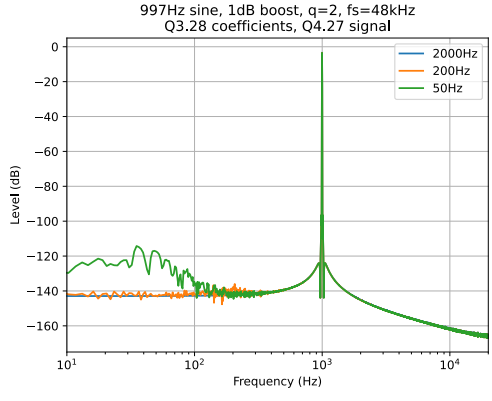

Fig. 4: Quantization noise vs filter frequency for Q3.28 coefficients and Q4.27 signal

In Fig. 4 we have allocated two extra bits of headroom for the coefficients using a Q3.28 representation rather than a Q1.30 representation. Very little extra quantisation noise is visible compared to Fig. 3. The coefficients have been rounded slightly different (an error of 2^-28), meaning that the filter has a slightly different characteristic; the peak or gain may have moved by 2^-28, but very little extra noise is amplified into the signal.

5 VPU Quantization

In a CPU-based fixed point biquad implementation, the int32 coefficients and samples are multiplied and accumulated into an int64, which is then truncated back to an int32. This is shown in Fig. 5.

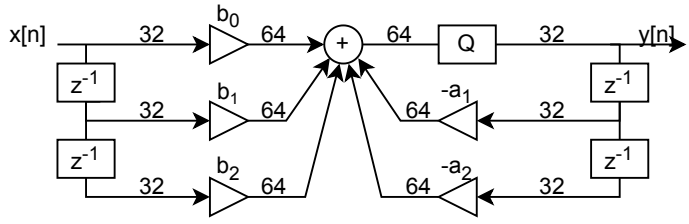

Fig. 5: Quantization on the CPU

In this CPU implementation a 64-bit accumulator is used that keeps full precision answers without rounding. That is, two Q0.31 values are multiplied into a Q1.62 value, and added to a Q1.62 accumulator. Any headroom has to be accounted for by adding headroom to either the signal or the coefficients. An implementation of this for XMOS devices is available in the lib_dsp library.

When using vector instructions on the VPU to implement a biquad, the products of the multiplication are truncated to 34 bits before being summed into a 40 bit accumulator as is shown in Fig. 5. An implementation of this for XMOS devices is available in the lib_xcore_math library.

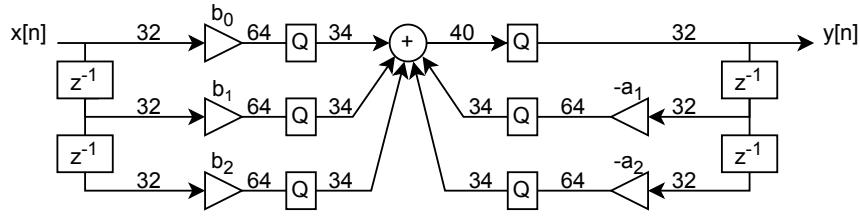

Fig. 6: Quantization on the VPU

As a consequence of the 34-bit quantisation on the VPU, compared to the 64-bit quantisation on the CPU, extra noise is added to the output of a biquad. This can be seen in Fig. 7 where one simple peaking biquad (a 50 Hz boost) has been executed using three forms of arithmetic: double-precision float, a CPU based int64 accumulator with int32 signal, and a VPU based int40 accumulator. The double precision float is to all intends and purposes equal to a mathematical real value; one can see the noise floor rise to -125 dB for the CPU model and -120 dB for the VPU model.

The quantization noise can reach significant levels if both the coefficients and signal have headroom bits in them as shown in Fig. 8. In this case, we have allocated two extra bits of headroom in the coefficients, and two extra bits of headroom for the signal. This has resulted in the noise floor for the VPU implementation creeping up to just above -100dB. As such, additional care should be taken when using the VPU optimised biquad functions.

Note that for higher frequency filters with lower pole magnitudes, the quantization noise is still very low, this is shown in Fig. 9 where we show the three different forms of arithmetic for a peaking filter with a 2000 Hz boost, using the same headroom as in Fig. 8. Note that all three lines are superimposed on each other.

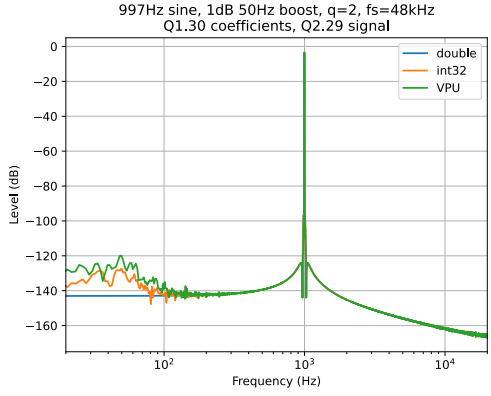

Fig. 7: Quantization noise for a 50 Hz boost with Q1.30 coefficients and Q2.29 signal

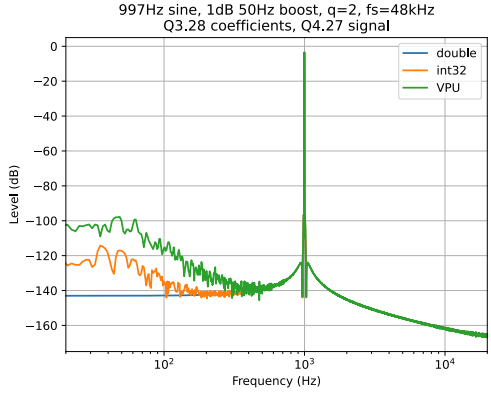

Fig. 8: Quantization noise for a 50 Hz boost with Q3.28 coefficients and Q4.27 signal

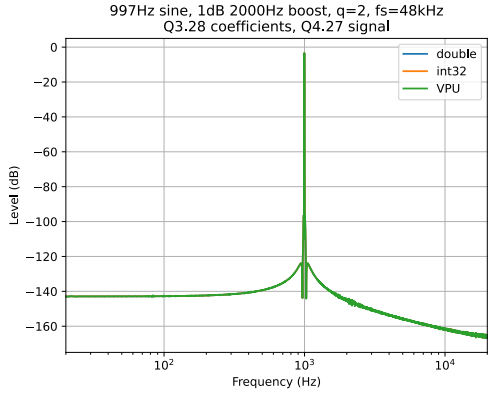

Fig. 9: Quantization noise for a 2000 Hz boost with Q3.28 coefficients and Q4.27 signal

6 Conclusion

Care should be taken when using the VPU optimised biquad functions. The headroom of the signal should be analysed, and headroom bits in the signal should only be added if strictly necessary. Any unnecessary headroom in the signal may become visible as a raised noise-floor, especially in filters with a high pole magnitude.

If poles with high magnitudes are required, it is worth considering to run those biquads on the CPU, and all the other biquads on the VPU. The CPU based biquads require more instruction cycles per biquad, but that may be easily managed if only a very small fraction of the biquads is ran on the CPU.

7 Further reading

- Robin John Clark, "Investigation into Digital Audio Equaliser Systems and the Effects of Arithmetic and Transform Error on Performance", available at: https://spmc.plymouth.ac.uk/rclark.html
- Rhonda Wilson, "Filter Topologies", available at: https://www.aes.org/e-lib/browse.cfm?elib=6169