Principles of Echolocation
in Bats and Machines

Note 3

From One Bit at 4.5 MHz to 450 kHz

PDM microphones, noise shaping and the choice of the decimation filter

Jan Steckel
Cosys-Lab, University of Antwerp, and Flanders Make, Belgium jan.steckel@uantwerpen.be

Version 2026-10-31 16:15

1 Why One Bit?

An eRTIS sensor listens with 32 MEMS microphones. Each of them delivers its signal not as a stream of 16- or 24-bit samples, but as a single bit, clocked at 4.5 MHz: a pulse-density modulated (PDM) stream. The advantages for an array are large. The analog-to-digital converter sits inside each microphone, every microphone needs only a clock and a data line, and two microphones can share one line (one on each clock edge). Before we can match filter and beamform, however, each of these 1-bit streams has to be turned back into ordinary samples at a sensible rate, here 450 kHz. This step, the decimation, is the subject of this note. As we will see, there is more than one good way to do it, and each has its consequences.

2 Sigma-Delta Modulation and Noise Shaping

Inside the microphone, a sigma-delta modulator compares the analog signal x with the bits it has already sent and chooses the next bit, $y[n] = \pm\frac{1}{2}$, so that their running average follows the signal. The density of +1s then tracks the waveform (Fig. 1a). A single bit is, of course, a very coarse quantizer, and the quantization error e is large. The trick is that the modulator does not leave this error white. With a loop filter, it shapes it:

$$Y(z) = X(z) + \mathrm{NTF}(z)\,E(z), \tag{1}$$

where the noise transfer function NTF is a high-pass with its zeros at (or near) zero frequency. The signal passes unchanged, while the quantization noise is pushed out of the band where the signal lives, to the high frequencies that the 4.5 MHz clock offers in abundance. An L-th order modulator gives a noise that rises with $6L$ dB per octave above the signal band.

Figure 1b shows this in the raw bit streams of the 32 microphones of an eRTIS (Knowles SPH0641LU4H-1), recorded for 2 s with the emitter off. In the call band, the noise floor of the microphones lies at about -141 dB re 1 V^2/Hz around 30 kHz and -131 dB at 50 kHz, and the 32 channels agree to within 1.5 dB. From about 80 kHz, the shaped noise takes over. It rises steeply, by about 24 dB per octave, which points to a fourth-order modulator, and levels off at about -68 dB re 1 V^2/Hz above 1 MHz, more than 70 dB above the floor in the call band. Almost all the power of the bit stream is this shaped noise. For comparison, the magenta curve is the same kind of stream while the eRTIS emits its call. The narrow lines at 30 and 35–37 kHz are ultrasonic tones in the lab, and a strong idle tone sits at exactly 2.25 MHz, a typical artifact

of sigma-delta modulators. It is harmless here: 2.25 MHz is five times 450 kHz, so it folds onto 0 Hz.

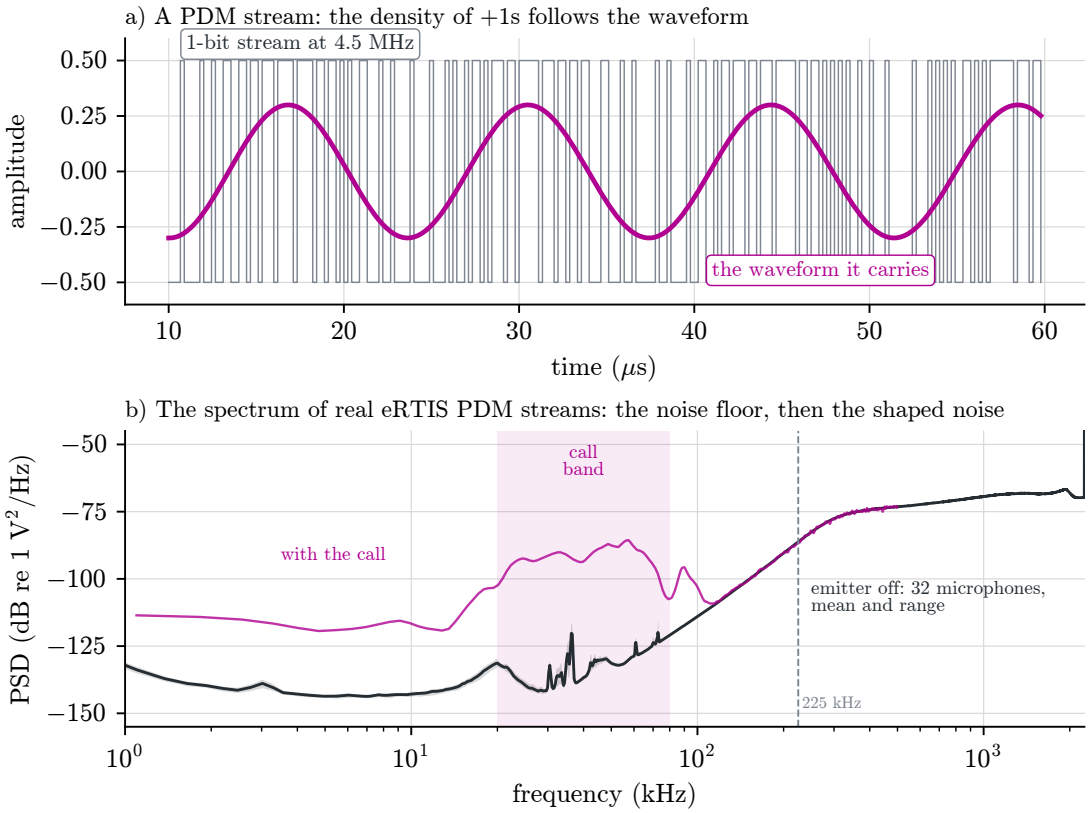


Figure 1: a) A 1-bit stream from a textbook fourth-order modulator (gray) and the waveform it encodes (magenta), a short piece of the eRTIS call. b) The power spectral density of the raw 1-bit streams of the 32 eRTIS microphones, emitter off (black: mean, gray: range over the channels, 2 s at 4.5 MHz), and of a stream recorded during the call (magenta, up to 0.5 MHz). The real microphones shape their noise more steeply than the textbook modulator in a).

3 The Real Problem: Down-Sampling

As long as we stay at 4.5 MHz, the shaped noise does no harm. It lives far above the call, and nothing we do to the call band touches it. The problem starts when we go to 450 kHz by keeping one sample in ten. Every frequency above the new Nyquist frequency of 225 kHz then folds back into the range from 0 to 225 kHz: a component at frequency f lands on $|f - k \cdot 450\ \text{kHz}|$ for the integer k that brings it into that range. Without a filter, the whole 0.25 V^2 of the bit stream ends up between 0 and 225 kHz, as a flat floor at -60 dB re 1 V^2/Hz, some 70 dB above the noise floor of the microphones (Fig. 3a, measured). Once folded, noise and signal share the same frequencies, and no later filter can separate them again. Everything therefore depends on the low-pass filter applied *before* the down-sampling.

Take 50 kHz, in the middle of the call band. After down-sampling, it also receives whatever was at 400 and 500 kHz (450 ∓ 50), at 850 and 950 kHz (900 ∓ 50), at 1300 and 1400 kHz, and so on. The whole call band is thus stacked with the bands $k \cdot 450\ \text{kHz} \pm (20 \ldots 80)$ kHz (shaded in Fig. 2a), and in those bands, the shaped noise is many tens of dB stronger than the noise of the modulator in the call band itself. A low-pass filter before the down-sampling has two jobs:

1. suppress the bands that fold onto the call band, so that the folded noise ends up well below the signal,

2. keep the call band itself flat, in amplitude and preferably in delay.

A third job is optional: suppress the band from 100 to 225 kHz. It does not fold, and stays in the output as it is. The eRTIS software uses a cutoff of 100 kHz. We compare three families of filters with this cutoff (Fig. 2), and above all what is left of the spectrum after filtering *and* down-sampling (Fig. 3b).

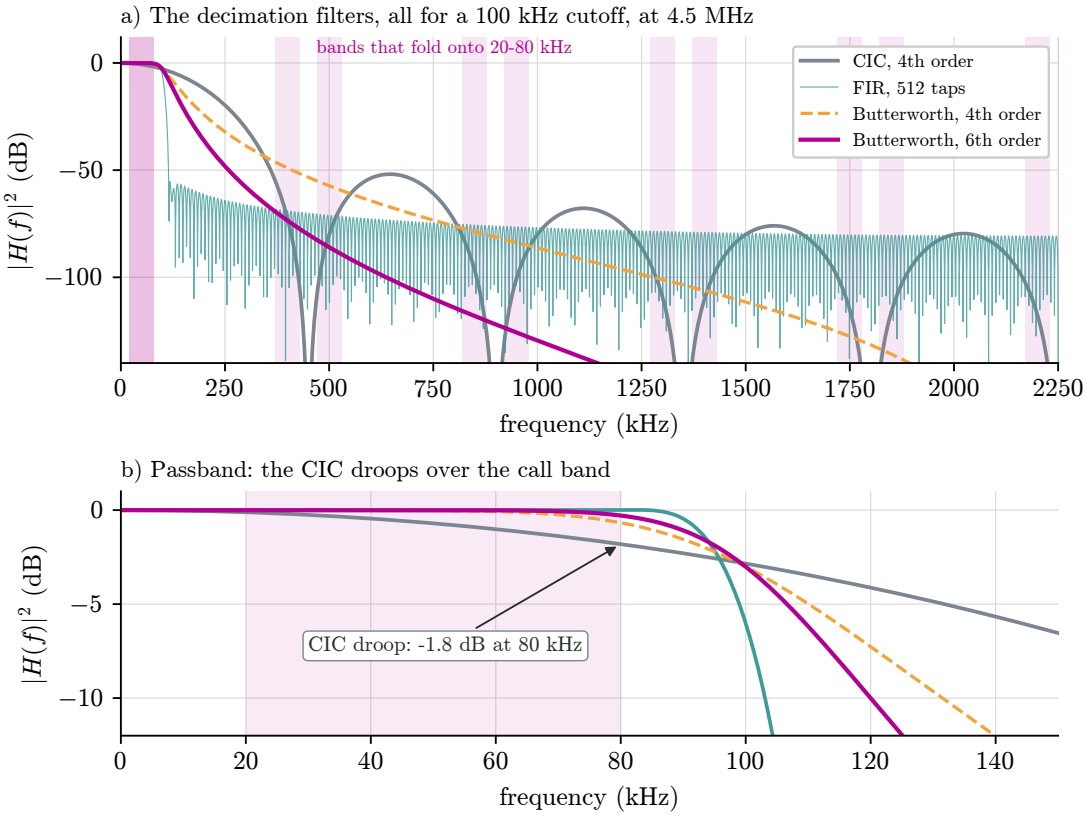


Figure 2: The decimation filters, all designed for a 100 kHz cutoff at 4.5 MHz. a) Full response, with the bands that fold onto the call band (shaded). b) Passband.

The CIC filter. A cascaded integrator-comb filter of order N consists of N integrators (running sums) at 4.5 MHz, the down-sampling, and N differences at 450 kHz. Its response is

$$H_{\mathrm{CIC}}(f) = \left(\frac{\sin(\pi f R/f_s)}{R \sin(\pi f/f_s)} \right)^N, \qquad R = 10, \tag{2}$$

which has its zeros exactly at the centers of the folding bands, $k \cdot 450$ kHz. It needs no multiplications at all, only $NR+N$ additions per output sample, which is why it is the classic choice in hardware. Its weaknesses follow from the same formula. The response falls over the call band (a droop of 1.8 dB at 80 kHz for $N = 4$), the zeros are narrow, so that the edges of the folding bands are attenuated by only 55 dB at 370 kHz, and above all, the CIC hardly attenuates the band from 100 to 225 kHz. That shaped noise stays in the output, which therefore rises again above 120 kHz (Table 1, output noise from 100 to 225 kHz). How much a CIC suppresses the folding bands depends on its order: the zeros must be deep and wide enough to hold down a noise that rises as steeply as the shaped noise does. The classic rule is that the order of the CIC should exceed the order L of the modulator, $N \geq L+1$ (Candy and Temes, 1992). For our microphone, which looks like a fourth-order modulator, that is a fifth-order CIC. A low-order CIC is indeed a poor decimator: a running average, which is a CIC with $N = 1$, raises the noise in the call band by more than 40 dB (Table 1).

The FIR filter. The eRTIS processing software uses a windowed-sinc FIR low-pass of 512 taps, a general low-pass, not a filter optimized for decimation (which only needs attenuation in the folding bands). Its passband is flat, its phase is linear, its stopband suppresses everything above 125 kHz by at least 56 dB, and because the input consists of single bits, every multiplication reduces to adding or subtracting a coefficient. Moreover, only every tenth output has to be computed. The price is length: 512 additions per output sample, and a delay of 255.5 input samples, or 57 μs (25.6 output samples, a range offset of about 1 cm that has to be calibrated away).

The IIR (Butterworth) filter. A recursive filter reaches a steep roll-off with very few coefficients. We often use a fourth- or sixth-order Butterworth low-pass, implemented as two or three second-order sections. Two things change compared with the FIR. Firstly, a recursive filter needs all its past outputs, so it must run at the full 4.5 MHz, and after the first section its signals are no longer single bits: five multiplications per section per input sample, or at most 100 (fourth order) to 150 (sixth order) per output sample (the first section sees single bits, so its numerator needs no multiplications). Secondly, its phase is not linear: the delay depends on frequency, from 6.2 to 9.1 μs over the call band for the sixth order (Fig. 4b). Because the cutoff is only 2% of the sampling rate, the poles lie very close to $z = 1$, and the filter must be implemented as second-order sections in sufficient precision.

4 Consequences

Compared with the call, the folded noise is small for every filter. But the call is the loud direct path from the emitter next to the microphones, while real echoes are much weaker, and what decides whether a weak echo can be detected is the noise floor in the passband. We therefore passed the real bit streams of Fig. 1b, recorded with the emitter off, through each decimator, and measured the noise floor after down-sampling (Fig. 3b, Table 1). The dotted curve is the floor without any folding: the noise of the microphones themselves.

- The fourth- and fifth-order CIC, the compensated CIC, the 512-tap FIR and the sixth-order Butterworth stay on that floor: in the call band, folding adds less than 1 dB. (The 512-tap FIR leaks a little through its finite stop band: about 4 dB at 30 kHz, where the microphones are quietest.)

- The fourth-order Butterworth folds in a flat floor at about -122 dB re 1 V^2/Hz: its 24 dB per octave only just balances the rise of the shaped noise. At 30 kHz, where the microphones are quietest, that is 18 dB above their own noise, and the noise in the call band rises by 6.5 dB in total. The third-order CIC does almost the same (7.3 dB), mostly in the upper half of the band.

- The second- and first-order CIC (a running average) are unusable: they raise the noise in the call band by 22.9 and 42.7 dB.

This is the rule $N \geq L + 1$ in practice: the microphones behave like a fourth-order modulator, and only CICs from the fourth order on, or other filters with an equally steep roll-off, keep the passband clean. The order of the filter matters more than its family. At the same order four, the CIC even does better than the Butterworth (0.6 against 6.5 dB), because its zeros sit exactly on the folding bands. To be fully clean, the CIC needs the fifth and the Butterworth the sixth order, and then the two are equally good, except that the CIC gets there with adders only.

CIC: cheapest by far, but the call band is not flat, and the output keeps the shaped noise between 100 and 225 kHz. The droop can be equalized later, and the matched filter rejects most of the out-of-band noise, but any processing that looks at the raw signals (an energy detector, a noise estimate) sees a noise floor that rises with frequency. In hardware, the CIC is therefore usually followed by a short compensating FIR at the low rate. A fifth-order CIC with a 31-tap

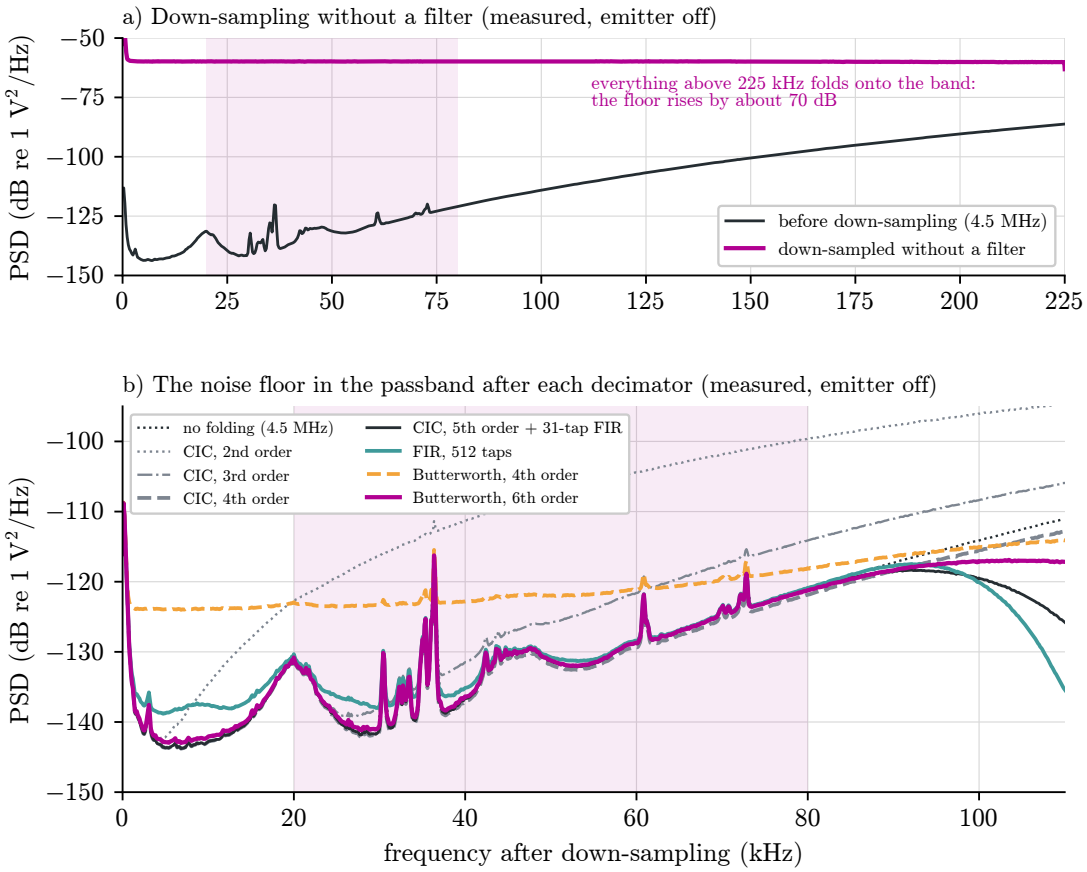


Figure 3: The real bit streams of the 32 eRTIS microphones (emitter off) after down-sampling to 450 kHz. a) Without a filter, all the shaped noise folds onto 0–225 kHz. b) In the passband, after each decimator (mean over the 32 channels). Dotted: the noise of the microphones before down-sampling, without folding. The peaks at 35–37 kHz are ultrasonic tones in the lab.

	droop at 80 kHz (dB)	attenuation at 370 kHz (dB)	noise floor[a] at 30 kHz	noise floor[a] at 50 kHz	noise rise by folding[b] (dB)	output noise 100–225 kHz[c]	delay in the call band (μs)	operations per output sample
no folding (microphones)	–	–	-141	-131	–	–	–	–
CIC, $N = 1$	0.5	14	-90	-86	42.7	-27	1.0	11 add.
CIC, $N = 2$	0.9	27	-116	-108	22.9	-39	2.0	22 add.
CIC, $N = 3$	1.4	41	-138	-126	7.3	-46	3.0	33 add.
CIC, $N = 4$	1.8	55	-141	-132	0.6	-51	4.0	44 add.
CIC, $N = 5$	2.3	68	-141	-132	0.0	-55	5.0	55 add.
CIC, $N = 5$, + 31-tap FIR	0.1	68	-141	-131	0.0	-81	38.3	55 add. + 16 mult.
FIR, 512 taps	0.0	76	-138	-131	0.3	-83	56.8	512 add.
Butterworth, 4th	0.7	46	-123	-122	6.5	-61	4.2–6.1	$\leq$ 100 mult.
Butterworth, 6th	0.3	69	-140	-131	0.1	-70	6.2–9.1	$\leq$ 150 mult.

Table 1: The decimators (4.5 MHz to 450 kHz, call band 20–80 kHz), measured on the real bit streams of the 32 eRTIS microphones with the emitter off. [a] dB re 1 V^2/Hz, after filtering and down-sampling. [b] In-band noise (20–80 kHz) after down-sampling, relative to what the same filter passes in the band without folding. [c] Total noise power between 100 and 225 kHz in the 450 kHz output, dB re 1 V^2. The compensated CIC uses a symmetric 31-tap FIR at 450 kHz that flattens the droop up to 80 kHz and suppresses everything above 120 kHz. Operations are per microphone. For 32 microphones at 450 kHz, multiply by $1.44 \cdot 10^7$ per second.

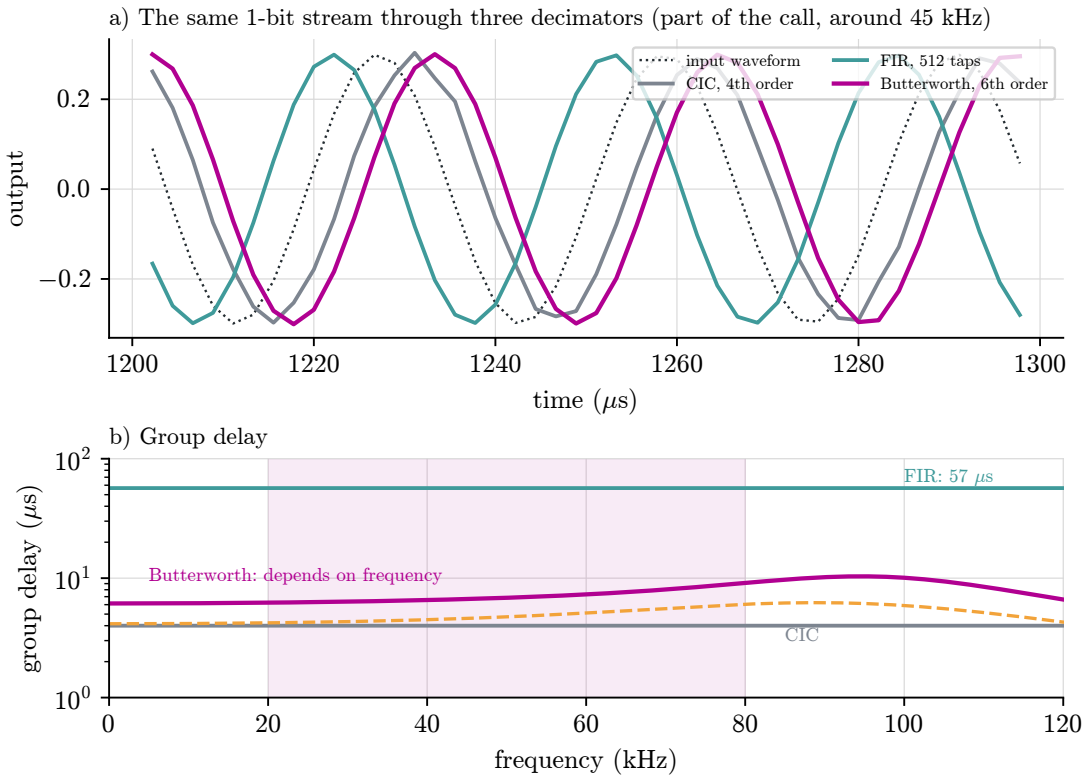


Figure 4: a) The same 1-bit stream (a piece of the call around 45 kHz) after three decimators: each filter delays the signal differently, and the CIC also reduces its amplitude (droop). b) The group delay of the filters.

compensator is flat to within 0.1 dB and as clean above 120 kHz as the 512-tap FIR, for 55 additions and 16 multiplications per output sample (Table 1).

FIR: the cleanest result (flat, linear phase, everything above 120 kHz gone), at the highest cost and with the largest delay. On a GPU, which is where the eRTIS software runs it, the cost is no problem.

IIR: a steep roll-off with a handful of coefficients, and almost no delay. Its phase is not linear, but for an array this hardly matters: all 32 channels pass through the same filter, so the beamformer sees no difference between them, the variation of the delay over the call band (2–3 μs from 20 to 80 kHz) corresponds to less than a millimeter of range, and it can be included in the reference of the matched filter. The sixth order removes the shaped noise almost as well as the FIR, while the fourth order folds a flat noise floor into the passband (Fig. 3b), because its 24 dB per octave just balances the rise of the shaped noise.

5 FPGA, GPU or CPU?

Which filter is best also depends on where it runs. For the 32 microphones of an eRTIS, the decimator delivers $32 \times 450{,}000 = 1.44 \cdot 10^7$ output samples per second. Table 2 turns the operation counts of Table 1 into rates, and the three platforms favor different filters.

FPGA. The CIC is made for FPGAs: integrators and differences need only adders and registers, no multipliers. Thanks to modular (wrap-around) arithmetic, the integrators may overflow freely, as long as the registers hold $1 + N \log_2 R \approx 15$ bits for $N = 4$. Its weaknesses, the droop and the noise above 100 kHz, are then fixed by a short FIR at 450 kHz, which needs only a few multipliers. This CIC-plus-compensator chain is the standard in hardware. A long FIR on 1-bit input needs no multipliers either (every product is $\pm$ a coefficient), but 512

filter	operations per second, 32 microphones	rate	suits
CIC, $N = 4$	$0.63 \cdot 10^9$ additions	4.5 MHz (integrators)	FPGA
CIC, $N = 5$, + 31-tap FIR	$0.79 \cdot 10^9$ add. + $0.23 \cdot 10^9$ mult.	4.5 MHz and 450 kHz	FPGA
FIR, 512 taps	$7.4 \cdot 10^9$ additions	450 kHz	GPU
FIR, 512 taps, 8-bit tables	$0.92 \cdot 10^9$ table lookups	450 kHz	CPU
Butterworth, 4th order	$1.4 \cdot 10^9$ multiplications	4.5 MHz, sequential	CPU, FPGA
Butterworth, 6th order	$2.2 \cdot 10^9$ multiplications	4.5 MHz, sequential	CPU, FPGA

Table 2: The decimators for an eRTIS (32 microphones, 4.5 MHz to 450 kHz), and the platform they suit best.

additions per output and per channel cost many adders, or many clock cycles. An IIR filter is feasible as well: at 4.5 MHz, a design clocked at 200 MHz has 44 cycles per input sample to close its feedback loop. The difficulty lies in the arithmetic. With a cutoff at 2% of the sampling rate, the poles lie close to $z = 1$, so the filter must be built from second-order sections with wide fixed-point words (typically 24 to 32 bits), or it drifts and produces limit cycles.

GPU. A GPU wants thousands of independent tasks. The FIR provides them: every output sample is an independent dot product, and only every tenth one has to be computed. This is why the eRTIS processing software, which runs on a GPU, uses the 512-tap FIR. An IIR filter is the opposite: every sample depends on the previous one, so the work only parallelizes over the 32 channels, which leaves most of a GPU idle (unless the recursion is rewritten as a parallel prefix scan). A CIC fits in between: its integrators are prefix sums, which GPUs compute well, but only in integer arithmetic: the N cascaded running sums over the 163,840 samples of one eRTIS recording grow like n^N, far beyond the precision of single-precision floats.

CPU. The IIR filter is a good match for a CPU. The recursion is sequential in time, but the 32 channels run side by side in the vector (SIMD) units, and $2 \cdot 10^9$ multiply-adds per second fit on a single modern core. A FIR filter on 1-bit input becomes cheap with a classic trick: take the input bits eight at a time, and replace the eight coefficient additions by one lookup in a precomputed table of 256 partial sums. The 512 taps then need 64 lookups per output sample, about $0.9 \cdot 10^9$ per second for the whole array. In practice, this takes 64 tables of 256 entries (one per block of eight taps, 64 kB in single precision). Because ten is not a multiple of eight, the bit window moves by two bits relative to the byte boundaries from one output to the next, so the bits are re-aligned with shifts or the tables are prepared for each alignment. When two microphones share a data line, their interleaved bits must first be separated. A CIC is the cheapest option on a CPU too, but its noisy output then still needs a second filter.

What We Would Use, and Why

A general-purpose PC (CPU). A fifth-order CIC with a short compensating FIR, in integer arithmetic. It is as clean as the long FIR in the passband, its phase is linear, its running sums vectorize well, and the channels are independent, so they spread over the cores. If simplicity matters more than linear phase, a sixth-order Butterworth in second-order sections (one library call, double precision) is equally clean. Avoid the fourth-order Butterworth and CICs below the fourth order, because they raise the noise floor.

An embedded microcontroller (STM32, ESP32). Do not decimate 32 channels in software. Even the cheapest clean decimator needs on the order of 10^9 operations per second for the whole array, several times what a microcontroller can execute. The microcontroller should do what the eRTIS does: capture the raw bits of all microphones with DMA (one 32-bit word per clock) and send them to a host. If a few channels must be decimated on the chip, use hardware: STM32 parts with a DFSDM peripheral contain sinc filters, which are CICs up to the fifth order, and a short compensating FIR in software then completes the recommended chain. The PDM receiver of the ESP32 is designed for audio microphones, so for an ultrasonic

array, capture the raw bits instead.

An FPGA. Again a fifth-order CIC with a compensating FIR. The CIC needs only adders and narrow wrap-around registers, and costs almost nothing even for 32 channels. A single compensating FIR at 450 kHz, shared by all channels, needs only a few multipliers. Avoid an IIR filter here, because of the wide fixed-point words and the limit cycles, and a long FIR, because of the many adders.

A GPU. A long FIR, computed as a strided convolution. Every output sample is independent, which is exactly what a GPU needs, and the length of the filter costs almost nothing. The FIR can therefore also be made longer, or designed specifically for decimation, to remove the small leak of the current 512-tap filter at 30 kHz. An IIR filter is a poor match, because it is sequential in time, and a CIC brings no advantage on a GPU.

Key idea

A PDM stream is mostly noise, pushed out of the signal band. The decimation filter has to remove it before the down-sampling folds it back onto the quiet call band. On the real eRTIS microphones, which behave like a fourth-order modulator, a CIC of order four or five (ideally with a short compensating FIR), the 512-tap FIR and a sixth-order Butterworth keep the passband at the noise of the microphones themselves, while a fourth-order Butterworth and low-order CICs raise it by 7 to more than 40 dB. Among the good ones, the choice is about flatness, delay, phase and cost.

Exercise

1. Why must a recursive (IIR) decimation filter run at the full input rate, while a FIR filter only computes every tenth output? What does that mean for the cost of the two?

2. The CIC has zeros at every multiple of 450 kHz. Why does that suppress the noise folding onto 0 Hz perfectly, but the noise folding onto 80 kHz much less?

3. The eRTIS software also has an audible mode, with a PDM low-pass at 24 kHz. Which filter family would you choose there, and why?

4. Design the cheapest filter chain (for example a CIC followed by a short FIR) that keeps the droop below 0.5 dB at 80 kHz and the folded noise 20 dB below the modulator's own noise in the call band.

Further Reading

- R. Schreier and G. C. Temes, *Understanding Delta-Sigma Data Converters*, Wiley-IEEE Press, 2005.
- J. C. Candy and G. C. Temes (eds.), *Oversampling Delta-Sigma Data Converters: Theory, Design, and Simulation*, IEEE Press, 1992.
- E. B. Hogenauer, "An economical class of digital filters for decimation and interpolation," IEEE Trans. Acoust., Speech, Signal Process., 29(2), 155–162, 1981.
- R. Kerstens, D. Laurijssen and J. Steckel, "eRTIS: A fully embedded real time 3D imaging sonar sensor for robotic applications," IEEE ICRA, 2019.