The Filter Wizard
issue 32: Direct Waveform Synthesis Using Filter-like Techniques
Kendall Castor-Perry

One of my earliest exposures to the joys of filtering was constructing the VCF (Voltage-Controlled Filter) for an analogue music synthesizer. This was a heady brew of transistors, diodes, op-amps resistors and capacitors that actually did something exciting! I've been returning to the wonderful world of music synths recently, having been seized by the idea that the new Cypress PSoC 3 and PSoC 5 devices will make fantastic substrates for both old-school and bang-up-to-date synth circuit approaches.

Notes played from musical instruments have a pitch that's essentially set by their fundamental frequency. But they also contain a wealth of harmonics (also sometimes some non-harmonically related components) that provide the characteristic 'timbre' that enables us to identify and distinguish between them. Sound synthesis is an attempt to create complex sounds – generally, musically useful ones – from simpler elements. It's not a new idea, or limited to the realm of electronics. Pipe organ designers have been combining harmonic mixes from acoustic structures for centuries. More recently, rotating toothed wheels, sensed electromagnetically, helped to create the unmistakable bluesy sound of the Hammond Organ.

There are two common approaches to the electronic synthesis of musical sounds with specific harmonic structures: subtractive and additive. Subtractive is the classical way used from the start by Bob Moog, arguably the father of electronically controlled sound synthesis. You generate a harmonic-rich waveform at the desired note frequency, then shape its timbre with a filter whose response usually varies over time. The filter can't add any harmonics, only take them away, though it can have a peaky response that emphasizes some of the harmonics. This is the origin of the classic synth 'filter' sound. It's not a great way of mimicking the behaviour of real musical instruments. But the unreal electronic sounds that issued forth from those telephone exchange-sized patch panels of the early Moog modular synths were crucial to progressive rock music in the 1970s (pauses to play some Yes and ELP tracks...).

The additive approach, as you might surmise, involves assembling the timbral structure of a musical note one harmonic at a time. This is quite a 'handful' in the analogue domain; oscillators that neatly produce separate sinewave outputs at the fundamental frequency and a whole collection of harmonics are not easy to design. Hence the bias towards subtractive synthesis in analogue synths. With the advent of progressively more powerful digital signal processing techniques, additive synthesis has become more feasible, though it has never been as popular as some other techniques described in Martin Russ's excellent book "Sound Synthesis and Sampling".

Ever on the lookout for cool things to do with filters and filter-oriented hardware, I began trawling around for techniques that would enable me to replicate the great timbral characteristics of retro analogue synths in the digital domain within a Cypress PSoC 3 device. The problem falls into two distinct parts:

*	cleanly generating a suitable harmonic-rich waveform at a precise note frequency that's not simply related to the digital audio sample rate;
*	filtering that waveform to produce a characteristic, dynamically varying harmonic spectrum.

The first part is not as easy as you might think, and that's because we're working in the sampled domain. As a thought experiment, consider how you would make a clean squarewave with an exact 1:1 mark:space ratio, in a sampled system. A little of that thought stuff should suffice to reveal that such a squarewave can only be produced if the high and low periods of the squarewave are the same, **integer** multiple of the sample period. This means that any "perfect squarewave" you can produce digitally can only have a frequency equal to half the sample rate divided by an integer. Now, for this to give frequencies that are musically accurate enough over the full range of say a piano requires a sample rate that's inconveniently high for audio applications. Let's say we want to create 4186 Hz (a very high C) to an accuracy of 0.3 percent (about the limit of pitch difference perception; ideally we'd aim for better than this). This would require a sample rate of about 1.2 MHz. Realistically, we'd much rather work at standard 48 kHz or 96 kHz audio sample rates. If you just allow the signal to transition between states at the nearest sample point, you get a considerable amount of phase modulation on the waveform, which can make it sound unacceptably 'dirty' or 'rough'. This might be OK for a toy but it's not appropriate for a proper musical instrument.

The way round this conundrum is to start out with a digitally-generated **sinewave** from some form of DDS (Direct Digital Synthesis) or modeled oscillator process, creating a sequence of samples that represents a sinewave with a frequency that can be arbitrarily precisely defined. Of the two methods, I've a strong preference for modeling a sinewave oscillator digitally, since it needs no look-up table for the sinewave generation. As part of an earlier filter project, a colleague experimented with a digital sinewave generator built around an oscillating filter configuration and designed to run on PSoC 3's Digital Filter Block, so I know this approach will be feasible in practice.

Then, of course, you have to create the harmonics. Add enough of these together with the appropriate amplitude and phase, and you can build up any waveshape you like, subject to two caveats. The first one is that the frequency of any harmonic you add mustn't be so high that it aliases round and comes out at the wrong frequency. Disaster, for a musical instrument! In other words, stop adding harmonics when their frequency is above Nyquist. Or, more practically, when their frequency is above the cutoff frequency of the decimation filter in the DAC that's going to reproduce the signal. Otherwise it will probably beat with its own image, since most audio DACs use "half band" interpolation filters that let some image frequencies through near Nyquist. I touched on that issue right back in "Alias, Damned Alias and Statistics". In practice, memory limitations may limit the number of harmonics you can use, before you get to this point.

The second caveat is that phase is important – and some synthesis methods might actually not be able to generate harmonics with the phase you need. It happens that the classic old-school synth waveforms – sawtooth, triangle, and variable-ratio rectangular – can only be made by adding together weighted amounts of sin(nx), not by any combination of cos(nx) terms! We'll have a lot more to say on these matters in part 2.

Back to our sums of sinewaves. Figure 1 shows the result of adding together a 1 kHz sinewave and sin(nx) harmonics up to the 12th, in ratio inversely proportional to the harmonic number. In other words, to the fundamental add half as much 2nd harmonic, one-third as much 3rd harmonic, and so on. This same sum is shown in the Martin Russ book. The result is a useful approximation to a sawtooth waveform. In a practical synth, we'll use a higher number of terms, to get the harmonics to extend up into the 'presence' band, which confers much of the characteristic brightness of such sounds. If waveforms can be iconic, then this is the iconic waveform of the classical synthesizer.

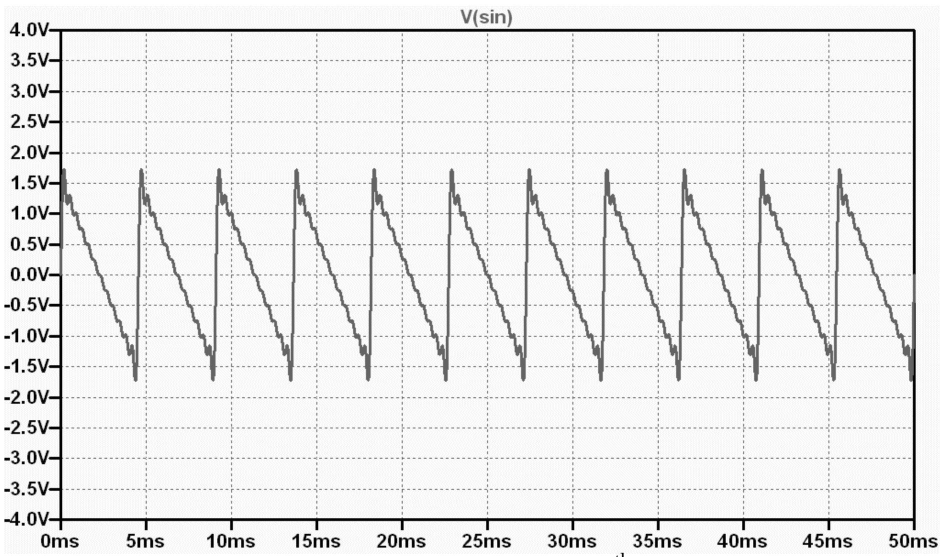

Figure 1: A 440 Hz sinewave and harmonics up to 12th added up with 1/n weighting.

We'll talk about how to actually create these harmonics next time, but let's first think what we would normally do with this waveform once it's created. In an old-school synth, this harmonic-rich waveform goes straight to a filter, often voltage-controlled, whose cutoff frequency behaviour usually tracks the desired note frequency (so that the timbre is independent of the pitch). As a simple illustration, figure 2 shows what figure 1's waveform looks like when we feed it through a 2nd order lowpass filter with a Q of 5 and a cutoff frequency of six times the fundamental note frequency.

Depending on the Q and the frequency of the filter, various harmonics are brought to prominence. Our auditory system is highly sensitive to this, because it's how we distinguish between different vowel sounds and even different speakers (of the human kind). So it's not surprising that messing around with harmonic structures like this can have such a noticeable subjective effect.

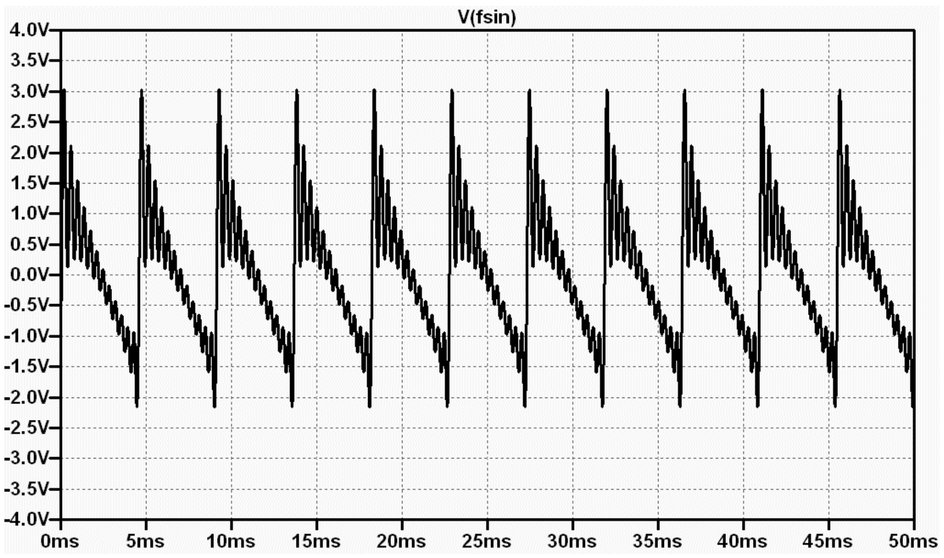

Figure 2: The sawtooth from figure 1 put through a very peaky filter.

To make things more interesting, the filter parameters, and hence the harmonic balance, might also vary with elapsed time after the onset of the note. This behaviour often takes the form of an additional trajectory of filter cutoff frequency and/or Q during the evolution of the note sound. As the balance between harmonics in the note changes, its character evolves over time, giving much more sonic interest (and a better match to real oscillating structures) than just a constant waveshape and timbre.

Now, before we go rushing off to design a super digital filter to do this process – and believe me, we certainly can, using the DFB on PSoC 3 or PSoC 5 – let's reflect on something. We just created a waveform with a particular harmonic balance, and now plan to filter it with a time-varying filter in order to shift the output's harmonic balance with time. But, why not just generate the waveform with the time-varying dynamic harmonic balance that the filter would have given in the first place, by changing the harmonic levels in the synthesis over time? Then we don't need a filter at all! You can imagine how much it pains a Filter Wizard to write that.

To make a time-varying filter, we have to dynamically update coefficients in the filter hardware anyway. So instead, let's just dynamically update the parameters that determine how much of each harmonic is present in the output waveform. If we can get the amplitude and the phase of each of the harmonics correct, we should be in good shape. Hah! That's a good pun, because it's the shape that changes if we get these wrong, as we will see. So, can we do it? Well, you'll just have to wait for part 2 to find out. Meanwhile, why not think about whether you have applications where an oscillator and a variable filter could be replaced by a programmable sum-of-harmonics generator. Additive might just become addictive! best / Kendall