# Hard Sync Without Aliasing

Eli Brandt
School of Computer Science, Carnegie Mellon University
email: eli@cs.cmu.edu

**Abstract**

*"Hard sync", a form of oscillator synchronization, is a technique which synthesizes a characteristic rich family of sounds. We describe how to perform it by integrating a bandlimited impulse pattern, avoiding the unpleasant aliasing heard in a naive digital rendering. The synthesis is refined by using a minimum-phase bandlimited step function, which eliminates lookahead and integration. This idea also gives simple bandlimited syntheses of other discontinuous waveforms.*

## 1 What is hard sync?

Oscillator synchronization involves two oscillators, a master and a slave, with frequencies $f_0$ and $f_1$. We will consider sawtooth oscillators. In *hard sync*, whenever the master cycles around, it resets the phase of the slave oscillator. The fundamental frequency of the slave's output, is thus equal to $f_0$. Figure 1 shows this for a master wavelength of 10 and a slave wavelength of 3. We treat the signal as unipolar, 0 to 1, for simplicity.

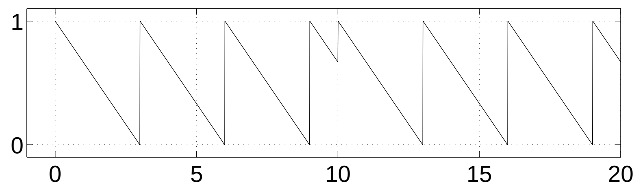

Figure 1: two cycles of hard-sync signal $y(t)$

The interesting regime is where $f_1$ is higher than $f_0$, but not (for very long) a multiple of it. Generally, the spectrum has formants at multiples of $f_1$—the technique is related to VOSIM (Kaegi and Tempelaars 1978) and FOF (Rodet 1984)—but the sound's richness comes from the complex evolution of harmonic amplitudes as $f_1$ is swept.

Varieties other than hard sync, and oscillators other than sawtooth, will be mentioned below.

## 2 Digital aliasing

The naive way to synthesize hard sync digitally is to run a simple digital oscillator at $f_1$, and reset its phase according to $f_0$. This is equivalent to sampling from the ideal hard-sync signal $y(t)$. That signal, however, is not bandlimited, and sampling it directly means aliasing. Just as with sawtooth or other simple oscillators, this is heard as roughness, sub-fundamental tones, and inharmonicity. Figure 2 shows the aliased spectrum that results, compared with the correct spectrum (from a waveform calculated as in Section 4.1), for $f_0 = 3/128$ and $f_1 = 8/128$.

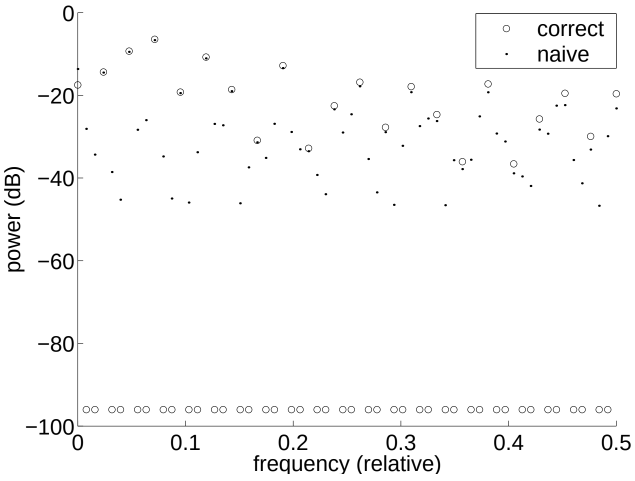

Figure 2: Spectra of correct and of naive hard sync

## 3 Hard sync in terms of impulse trains

We construct our hard-sync signal from impulse trains (whose bandlimited synthesis is a solved problem), following Stilson and Smith (1996). Later, in Section 6.1, we will construct hard sync from step functions.

The derivative of $y(t)$ in Figure 3 is the impulse pattern $y'(t)$ in Figure 3, with an appropriate DC offset. We treat $y(t)$ as unipolar, 0 to 1, for simplicity.

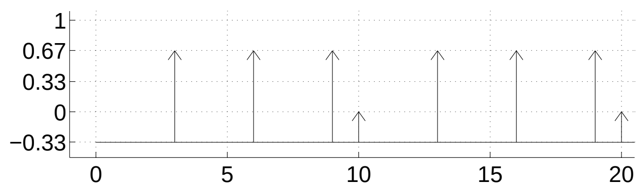

Figure 3: two cycles of hard-sync impulse pattern $y'(t)$

In general, for frequencies $f_0$ and $f_1$ (normalized 0–1 from DC to $f_s$),

$$y'(t) = \left(\sum_{i=1}^{n_{01}} p(t - iT_1)\right) + r_{01}p(t) - d$$

$$\begin{aligned} \text{where} \quad T_0 &= 1/f_0 \\ T_1 &= 1/f_1 \\ n_{01} &= \lfloor T_0/T_1 \rfloor \\ r_{01} &= T_0/T_1 - n_{01} \\ p(t) &= \sum_{i=-\infty}^{\infty} \delta(t - i/f_0) \\ &\delta(t) \text{ is the Dirac delta function} \\ \text{and} \quad d &= f_1 \text{ is a DC offset correction.} \end{aligned}$$

To get a hard-sync signal $y_b(t)$ that is bandlimited to $f_b$ ($f_b \leq f_{\text{Nyq}} = 1/2$), we integrate the bandlimited derivative $y'_b(t)$.

$$y'_b(t) = \left(\sum_{i=1}^{n_{01}} p_b(t - iT_1)\right) + r_{01}p_b(t) - d$$

$$\begin{aligned} \text{where} \quad p_b(t) &= \sum_{i=-\infty}^{\infty} \operatorname{sinc}(t - i/f_0) \\ \operatorname{sinc} x &= (\sin \pi x)/\pi x \\ \text{and} \quad d &= f_1 \text{ is a DC offset correction.} \end{aligned}$$

So we have reduced the problem to the synthesis of $p_b(t)$, a bandlimited impulse train.

## 4 Synthesizing impulse trains

This section summarizes the work of Stilson and Smith (1996), and applies it to the problem of hard-sync synthesis.

### 4.1 Exact syntheses

The signal $p_b(t)$ can be calculated exactly. Additive synthesis from cosines is straighforward, and will be efficient for sufficiently high $f_0$. For lower $f_0$, a better exact synthesis is

$$\begin{aligned} p_b(t) &= \frac{M}{P} \operatorname{sinc}_M \frac{M}{P} t \\ \text{where} \quad P &= 1/f_1 \\ M &= \lfloor bP \rfloor \\ \text{and} \quad \operatorname{sinc}_M t &= \frac{\sin \pi x}{M \sin(\pi x/M)} \end{aligned}$$

Some care must be taken here with the $0/0$ case.

### 4.2 Windowed-sinc approximation

A bandlimited impulse is a sinc function, time-scaled so as to have one zero-crossing per sample period. (Bandlimiting to below $f_{\text{Nyq}}$ means dilation, and concomitant scaling-down.) Stilson and Smith truncate and apply a window function to this signal, store it in a table, and use it to generate bandlimited impulse trains (BLITs): subsample the table whenever an impulse is desired. We reapply the technique, bypassing the BLIT $p_b(t)$ and generating our impulse pattern $y'(t)$ directly.

The windowed-sinc table is oversampled by a factor of $\Omega$. Generate each impulse by subsampling the table—by a factor of $\Omega$, if the bandlimit is $f_{\text{Nyq}}$; that is, stepping through it $\Omega$ places at a time. Linear interpolation is adequate if $\Omega$ is large enough. If a pulse is to be centered between samples, shifted past the last sample by $\alpha$, then start stepping from place $\Omega(1 - \alpha)$.

Master and slave phases are updated as in the straightforward (aliased) version. The difference is in the output, which is generated by placing an impulse center wherever the slave phase resets. The placement must be predicted, looking ahead half of an impulse length, which is troublesome when frequencies are varying. One can either make control inputs take effect with a delay, or assume they don't vary too quickly and predict as if they were constant over the span of the prediction.

There are several parameters in building the windowed-sinc table: the oversampling factor $\Omega$, the number of zero crossings $N_z$, and the window function. $\Omega$ should be large enough that the linear interpolation doesn't cause noise. The window function determines the level of the stopband—the level of aliasing—and also the shape of the transition from passband to stopband. The value of $N_z$ then determines the width of the transition, and $f_b$ its position, so that enough high frequencies are passed, but aliases are blocked.

Figure 4 reprises Figure 2, but uses the windowed-sinc technique, with $f_b = 1$, $\Omega = 64$, $N_z = 16$, and a Blackman window.

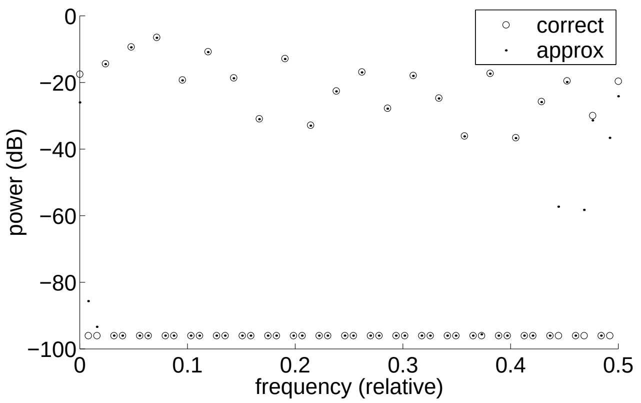

Figure 4: Spectra of correct and of windowed-sinc hard sync

## 5 Performing the integration

This would be a good place to explain the DC offset correction $d$. The $y'$ signals need their DC component to be zero. Each impulse has a sum of 1, so the appropriate downward shift is the number of full-height impulses plus the height of the single scaled-down one. This turns out, after some cancellation, to be $f_1$:

$$\begin{aligned} f_0(n_{01} + r_{01}) &= f_0(n_{01} + T_0/T_1 - n_{01}) \\ &= f_0(T_0/T_1) \\ &= f_0(f_1/f_0) \\ &= f_1 \end{aligned}$$

In fact, the windowed-sinc approximation generates an impulse whose sum is only approximately equal to 1. The deviation seems impossible to predict exactly (a table could approximate it), so $y'_b$ has a slight offset. Nor is it obvious how to get the DC offset correction exactly right while $f_1$ is changing. In any case, there is roundoff error. From these sources, $y'_b$ has an offset which is variable, but on the rough order of $10^{-6}$, or $-120$ dB.

As a consequence of all this, the integrator has to be leaky, as a perfect integrator has infinite DC gain. A first-order leaky integrator has finite DC gain, and if it is tuned to pass audio (attenuating 20 Hz by 0.5 dB), that DC gain is substantial, 60 dB. This brings the $y'_b$ offset up to a persistent $-60$ dB, enough to be bothersome.

A second-order leaky integrator has a DC gain of zero. It can be constructed by cascading a leaky integrator with a one-pole highpass:

$$H(z) = \frac{\pi}{1 - cz^{-1}} \frac{k(1 - z^{-1})}{1 - cz^{-1}}$$

where $k = (1 + c)/2$

and $c = 0.9992$ for $-0.5$ dB at 20 Hz ($f_s = 44.1\,\text{kHz}$)

The initial conditions for the integration are

$$y_b(0) = 1 - r_{01}$$

## 6 A more elegant synthesis

So far we have paralleled Stilson and Smith, developing syntheses for hard sync homologous to theirs for sawtooth. Now, the windowed-sinc approximation can be refined in two ways: rendering the integration step unnecessary, and eliminating the lookahead to the impulse center. These refinements translate back to the synthesis of bandlimited sawtooth and other waveforms.

### 6.1 Hard sync in terms of steps

The integration is complicated by the DC offset in $y'_b$, which requires us to tune parameters on the integrator, and which still leaks through transiently, as at startup. Why not eliminate the entire integration stage? Simply pre-integrate the windowed sinc to get what we might call a BLEP (a **b**and-**l**imited st**ep** function), whose final value we can ensure is 1. Now instead of placing impulses, place BLEPs.

Integrating by taking a running sum is not strictly correct, but the deviation is at high frequencies, falling to 0.25 dB at -2 octaves. The table being integrated here is typically oversampled several octaves more than that.

### 6.2 Minimum-phase impulses

Also, the windowed-sinc approximation is complicated by having to look ahead to place impulse centers. This lookahead comes about because the windowed sinc is symmetric, placing the bulk of its energy in the middle. This problem goes away if we consider the windowed sinc as an FIR filter, construct a minimum-phase filter with the same amplitude response, and use that instead.

Z-domain (polynomial) methods have some difficulty calculating a minimum-phase filter as long as this one, but working in the cepstral domain (Oppenheim and Schafer 1975) is robust and fast. The MATLAB function rceps (The MathWorks 2001) performs the desired operation, which in essence is to zero the upper half of the cepstrum.

Figure 5 shows the original linear-phase sinc pulse (with parameters as in Figure 4), and Figure 6 shows the minimum-phase pulse.

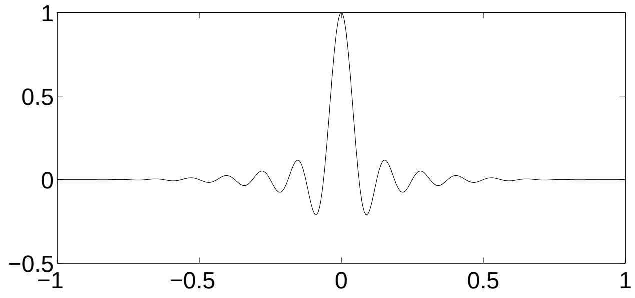

Figure 5: windowed-sinc pulse.

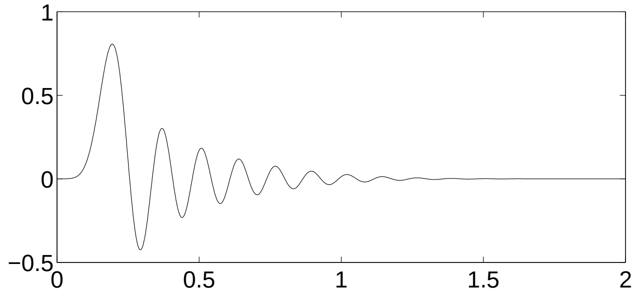

Figure 6: minimum-phase windowed-sinc pulse.

### 6.3 Minimum-phase steps

These improvements can be combined, for no lookahead and no integration stage, by integrating the minimum-phase bandlimited impulse to form a minimum-phase bandlimited step (a MinBLEP), shown in Figure 7.

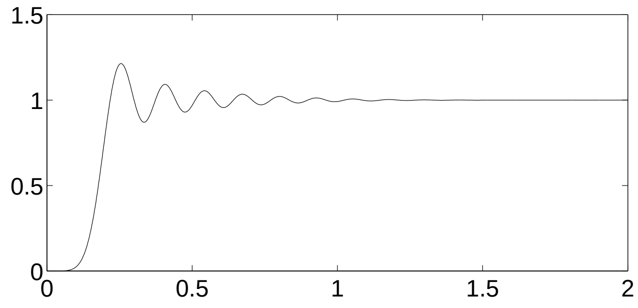

Figure 7: minimum-phase bandlimited step.

The MinBLEP solves directly the quite general problem of how to introduce a bandlimited discontinuity into a waveform: don't just jump to the new level, mix in a MinBLEP instead.

This is sufficient only if the first and higher derivatives are essentially continuous across the point of discontinuity, a property we name $\overline{C}1$ continuity. Square wave, sawtooth, and hard-synced square and sawtooth have this property, so their aliased digital syntheses can be corrected simply by replacing steps with MinBLEPs.

Hard-synced triangle, on the other hand, does not have $\overline{C}1$ continuity, but does have $\overline{C}2$. It could be synthesized from MinBLEPs and minimum-phase bandlimited ramps (which we will refrain from naming).

Hard-synced sine, finally, has no $\overline{C}n$ continuity. This approach cannot synthesize it exactly, but only approximate it; aliasing will fall off by 6dB/oct per derivative whose discontinuity is made bandlimited.

## 7 Efficiency

The per-sample cost of the MinBLEP approximation is roughly proportional to $f_1$, this being (as worked out in Section 5) the number of windowed-sinc impulses called for per sample. If an impulse is 32 samples long and $f_1$ is $1/32$, the cost is one table lookup and some bookkeeping overhead (including a branch). The number of table lookups scales with $f_1$.

It is true that users of hard sync do often sweep $f_1$ higher than $1/32$ (1.4 kHz at a sampling rate of 44.1 kHz). Very high $f_1$ is problematic for real-time implementation. The exact syntheses also take time linear in $f_1$, so they don't help. Finding an approximation that takes time sublinear in $f_1$ is an open problem.

## 8 Extensions

So far only hard sync has been addressed. In soft sync, the slave oscillator's phase is reset only if its current value lies within some window around zero. This is cumbersome to synthesize from BLITs, as its period of repetition can be arbitrarily long. However, the windowed-sinc or MinBLEP approximation of hard sync could be adapted to do this, even with variable hardness, by examining the slave phase at the end of each master cycle.

Further afield, the slave frequency can vary over the course of each master cycle. (The Korg Mono/Poly analog synthesizer allows this, for example.) Or the slave phase need not be reset to zero; it could be multiplied by a value between zero and one (or greater than one), or processed in other ways.

## 9 Conclusion

Anyone who has worked with analog synthesizers will remember the sound of hard sync. Digital synthesizers commonly omit hard sync, or render it with aliasing; this paper describes ways to synthesize it correctly. One of the techniques described, the minimum-phase bandlimited step, applies beyond hard sync, to all of the basic 'analog' waveforms having discontinuities, and further.

## References

Kaegi, W. and S. Tempelaars (1978). VOSIM—a new sound synthesis system. *Journal of the Audio Engineering Society 26*(6), 418–426.

Oppenheim, A. V. and R. W. Schafer (1975). *Digital Signal Processing*. Prentice-Hall.

Rodet, X. (1984). Time-domain formant wave-function synthesis. *Computer Music Journal 8*(3), 9–14.

Stilson, T. and J. Smith (1996). Alias-free digital synthesis of classic analog waveforms. In *Proc. International Computer Music Conference*. International Computer Music Association.

The MathWorks (2001). rceps (Signal Processing Toolbox). Online at `http://www.mathworks.com/access/helpdesk/help/toolbox/signal/rceps.shtml`.