Band-Limited Simulation of Analog Synthesizer Modules by Additive Synthesis

Amar Chaudhary
Center for New Music and Audio Technologies
University of California, Berkeley
amar@cnmat.berkeley.edu

March 12, 2001

Abstract

Analog synthesizers continue to be used by many musicians because of their distinctive timbres, intuitive real-time control and flexible patching. There has been recent interest in simulating the analog signal chain with digital techniques. However, a literal time-domain translation of analog VCOs and VCFs is surprisingly challenging. This paper presents a new strategy for digital simulation based on spectral descriptions of analog modules and additive synthesis techniques. A real-time software analog synthesizer has been implemented based on this strategy. Audio output is qualitatively close to that of analog synthesizers.

1 Introduction

Analog synthesizers continue to be used by many musicians because of their distinctive timbres, intuitive real-time control and flexible patching. Since digital synthesizers are cheaper to make and don't suffer from the tuning problems of their analog counterparts, there has been recent interest in simulating the analog signal chain with digital techniques. A literal time-domain translation of each analog functional module into digital form is surprisingly challenging. Creating band-limited sawtooth and square waves over a wide

variety of pitches is computationally expensive. Some filter topologies used in VCFs are not realizable digitally. When they are realizable, the accurate transformation of frequency and Q parameters to digital filter coefficients is very expensive.

We introduce a new strategy to avoid these problems based on operations on spectral descriptions of analog modules and additive synthesis techniques. This new approach avoids aliasing and filter design problems and preserves the control parameters of analog implementations (i.e., pitch, loudness, center frequency and bandwidth) without the need to convert to the problematic parameter space for digital filters. This strategy is also used to simulate analog noise generation.

Our implementation supports dynamic reconfiguration (i.e., "patching") of modules. Audio output is qualitatively close to that of analog synthesizers. The implementation runs in real time and responds to external controllers via MIDI or Open Sound Control [1].

The remainder of this paper is organized as follows: Section 2 explains why digital simulations of analog synthesizers are difficult. Section 3 describes our method based on additive synthesis. Section 4 discusses the application of this method to analog noise generation. Section 5 describes our implementation, and section 6 concludes the paper.

2 Why Digital Simulations of Analog Synthesizers are Difficult

Analog synthesizers typically use *subtractive synthesis*. In subtractive synthesis, sounds are produced by passing broadband signals through time-varying filters [2]. Analog synthesizers consist of modules that implement the steps of subtractive synthesis:

- *Voltage-controlled oscillators* (VCOs) produce broadband *excitation* signals at a given pitch.
- *Voltage-controlled filters* (VCFs) modify the timbre of signals. VCFs usually have controls for center frequency and Q (i.e., resonance).

- *Voltage-controlled amplifiers* (VCAs) scale the amplitudes of signals by a given gain.

The pitch, center frequency, Q and gain controls of these modules can be controlled directly or by using piecewise linear functions called *envelopes*, or low-frequency oscillators (LFOs). Figure 1 illustrates a typical chain of analog synthesizer modules.

VCOs generally produce a few types of waveforms, such as sawtooths, squares, triangles and pulse trains. Each of these waveforms can be described as an infinite sum of sinusoidal components, as shown in Table 1. As such signals are not band-limited, aliasing will occur producing unwanted frequency components in the output signal [3]. In order to prevent aliasing, the source signals must be *band-limited*, or contain only partials between the fundamental and the Nyquist frequency. For example, a band-limited square wave can be expressed using the following formula [1]:

$$\frac{1}{\pi}\sum_{i=1}^{N}\frac{1}{2i-1}cos(2(2i-1)\pi ft) \tag{1}$$

where f is the fundamental frequency of the square wave, and N is the number of partials. The number of partials is a function of the fundamental and the sampling rate:

$$N \leq \frac{R}{2f} \tag{2}$$

where R is the sampling rate. Because the fundamental varies while the Nyquist frequency $R/2$ remains constant, the number of partials in the band-limited spectrum changes. A time-domain implementation based on sample-playback or wavetable synthesis must store different versions of the waveform for different frequency ranges. This requires large amounts of dynamic storage [2].

Difficulties also arise when converting analog VCFs to equivalent digital filters. The distinctive timbres of analog synthesizers are often a result of the

[1] All time-varying functions in this paper are expressed as continuous functions of time for clarity and convenience. However, it should be noted that time is a discrete value in the digital domain. Digital-domain time is expressed as $t = n/R$ where R is the sampling rate and $n = 0, 1, 2, \ldots$

strongly non-linear response characteristics of the particular VCF topology used, so these functions should be modeled as accurately as possible. We also want to preserve for musicians the familiar analog parameters, such as center frequency and Q. Traditional conversion techniques use an isomorphism, such as the bilinear transformation to map between the analog and digital domains [4]. These techniques often preserve the parameterization but not the frequency response characteristics of the original analog filter [5]. VCF topologies that include feedback may not be realizable at all using these techniques. For example, the Moog four-pole VCF [6] is unrealizable using the bilinear transform unless a unit delay is added to the resulting digital filter [7].
Such hand-tweaking of converted filters complicates or, in some cases, destroys the relationship between the analog control parameters and filter behavior [7]. Designing a digital filter directly from the impulse response or frequency response of a VCF[8] will likely result in a filter with a nontrivial mapping between analog parameters and filter coefficients that is too expensive to compute in real time.

In the following section, we present a method which overcomes or avoids the problems described above.

3 Simulation Using Additive Synthesis

Many of the difficulties discussed in the previous section can be avoided by operating on spectral descriptions used in *additive synthesis*. In additive synthesis, sounds are modeled as a sum of sinusoid functions whose amplitude, frequency and phase change over time:

$$x(t) = \sum_{i=1}^{N} A_i(t) \cos(\omega_i(t)t + \phi_i(t)) \tag{3}$$

where N is the number of sinusoids in the sound, $A_i(t)$, $\omega_i(t)$ and $\phi_i(t)$ represent the amplitude, frequency and phase, respectively, of the ith sinusoid at time t. The *spectral description* of the sound at time t is the set of all instantaneous frequency, amplitude and phase values $\{\omega_i(t), A_i(t), \phi_i(t)\}$. A spectral discription may contain hundreds of sinusoids in order to accurately model a sound. The spectral description is converted to a digital waveform

using inverse-transform or oscillator additive synthesis [9].

Analog synthesizer modules can be simulated using a sequence of spectral-domain functions, as illustrated in figure 2. This simulation method requires an efficient implementation of additive synthesis that can handle hundreds of partials in real time [9].

The spectral-domain VCO is a function that outputs the spectral description of band-limited excitation waveforms. The description of a particular waveform is the band-limited subset of the frequency, amplitude and phase coefficients used in its summation formula, as listed in table 1. For example, the spectral description of a square wave of pitch f would be $\{\{f, 1, 0\}, \{3f, 1/3, 0\}, \{5f, 1/5, 0\}, \ldots, \{(2N-1)f, 1/(2N-1), 0\}\}$, where N is the total number of partials, as described in equation 2.

The spectral-domain VCF is a function which scales the amplitudes of the spectral discription according to the frequency-response function of the analog filter. For example, the Moog four-pole VCF has the following frequency-response function:

$$H(j\omega) = \frac{\omega_c^4}{(j\omega - \omega_c)^4 + k\omega_c^4} \tag{4}$$

where ω_c is the center frequency and k is the Q control. Each partial in the output spectral description has the same frequency and phase values as the corresponding input partial, but its amplitude value has been scaled according to frequency-resonpose function:

$$\{\omega_i, A_i H(j\omega_i), \phi_i\} \tag{5}$$

By operating in the spectral domain, the center frequency and Q controls are applied directly to the signal without the need for an intermediate calculation of digital filter coefficients. The filter conversion problem has been avoided.

This method can be applied to other VCF designs. The frequency and phase responses of the filter are analyzed at different values of its control parameters. The collected data can be used to construct a table-lookup function that takes frequency, phase and the control parameters as inputs and produces an appropriate amplitude-scaling value.

The spectral-domain VCA is trivial. The amplitude of each sinusoid is scaled by the gain control of the VCA.

4 Simulation of Noise

The additive synthesis strategy can also be used to simulate analog noise generation. The VCO noise signal is represented as a set of noise bands, each with a center frequency and an amplitude. White noise is generated by using a constant amplitude over all frequency bands, and pink noise is generated by setting the amplitude of each band to be inversely proportional to its center frequency. The frequency response function of the simulated VCF is then used to scale the amplitude of the noise band. The scaled noise bands can then be converted to a digital waveform using an inverse transform method for generating broad-band signals [10].

5 Implementation

We have implemented an analog synthesizer simulation using the method described in this paper. It consists of two voices, each with its own VCO, Moog-style VCF and VCA. The VCO has a pitch control, the VCF has controls for center frequency and Q, and the VCA has a gain control. Each of these controls has an indepedent ADSR (i.e., attack, decay, sustain and release) envelope and LFO as well as a slider for direct control. The VCO uses a maximum of 100 sinusoids to generate spectral descriptions for sine waves, sawtooths, square waves, triangle waves, pulse trains and white and pink noise. The VCF applies a Moog transfer function to the source spectrum. The spectral descriptions from the two voices are merged together and the resulting spectrum is converted to an output audio signal using the inverse transform method of additive synthesis [9]. The signal chain of this software analog synthsizer is illustrated in figure 3.

The modules were written as portable C++ objects that support dynamic "patching". The user interface was written using Tcl [11], a multi-platform scripting language and user interface toolkit. A screen capture of the user interface is shown in figure 4.

Our implementation has been tested on an Intel Pentium Pro 200Mhz running Windows NT 4.0, and an SGI O2 running Irix 6.3. Both implementations run in real time and respond to external controllers via MIDI or Open Sound Control. The external control messages can be mapped to any of the analog module parameters on either or both voices.

Listeners find this implementation qualitatively close to analog Moog synthesizers and confirm that the real-time controls agree with their intuition and experience with analog instruments.

6 Future Work

Our results have been very encouraging, both computationally and perceptually. Future work includes the analysis of additional VCF topologies, controls based on non-linear functions (e.g., the hyperbolic tangent), an improved user interface for dynamically adding and patching modules and optimization of spectral transform code for very large patches.

Some additional types of analog modules, such as ring modulators, can be more efficiently implemented as time-domain transforms. Such modules can easily be added at the end of a spectral-domain signal chain after the inverse transform operation, but require additional domain conversions if used in the middle of a signal chain.

Acknowledgements

I am grateful to Silicon Graphics Inc., the NSF Graduate Research Fellowship program and Gibson Music Inc. for their support of this research.

References

[1] M. Wright and A. Freed. "Open Sound Control: A New Protocol for Communicating with Sound Synthesizers." In *Proceedings of the 23rd International Computer Music Conference*, Thessoloniki, Greece, 1997. This paper can be found at `http://cnmat.cnmat.berkeley.edu/Research`.

[2] F.R. Moore. *Elements of Computer Music*. P T R Prentice Hall, Englewood Cliffs, NJ, 1990.

[3] T. Stilson and J. O. Smith. "Alias-free Digital Synthisis of Classic Analog Waveforms." Technical report, CCRMA, Stanford University, 1996. This paper can be found at `http://www-ccrma.stanford.edu/ stilti/papers`.

[4] K. Steiglitz. "The Equivalence of Digital and Analog Signal Processing." *Information and Control*, 8:455–467, 1965. Reprinted in *Digital Signal Processing*, IEEE Press, New York, 1972.

[5] S. J Orfandis. "Digital Parametric Equalizer Design with Prescribed Nyqyust-frequency Gain." *Journal of the Audio Engineering Society*, 45(6), June 1997.

[6] R. A. Moog. "A Voltage-controlled Low-pass High-pass Filter for Audio Signal Processing." processing". In *Audio Engineering Society Convention*, volume Preprint 413, October 1965.

[7] T. Stilson and J. O. Smith. "Analyzing the Moog VCF with Considerations for Digital Implementation." Technical report, CCRMA, Stanford University, 1996. This paper can be found at `http://www-ccrma.stanford.edu/ stilti/papers`.

[8] L. B. Jackson. *Digital Filters and Signal Processing, Second Edition*. Kluwer Academic Publishers, Boston, 1989.

[9] A. Freed. "Real-time Inverse Transform Additive Synthesis for Additive and Pitch Synchronous Noise and Sound Spatialization." AES 104th Convention, San Francisco, CA, 1998. This paper can be found at `http://cnmat.cnmat.berkeley.edu/Research`.

[10] A. Freed. "Bring Your Own Control to Additive Synthisis." In *Proceedings of the International Computer Music Conference*, Banff, Canada, 1995. This paper can be found at `http://cnmat.cnmat.berkeley.edu/Research`.

[11] B. B. Welch. *Practical Programming in Tcl & Tk, 2nd Edition*. Prentice Hall, Upper Saddle River, NJ, 1997.

Waveform	Summation
Sawtooth	$\frac{1}{\pi}\sum_{i=1}^{\infty}\frac{(-1)^i}{i}\cos(2\pi ift)$
Square	$\frac{1}{\pi}\sum_{i=1}^{\infty}\frac{1}{2i-1}\cos(2(2i-1)\pi ft)$
Triangle	$\frac{4}{\pi^2}\sum_{i=1}^{\infty}\frac{(-1)^i}{(2i-1)^2}\cos(2\pi ift)$
Pulse Train	$\sum_{i=1}^{\infty}\cos(2\pi ift)$

Table 1: Common VCO waveforms as infinite sums of sinusoidal components

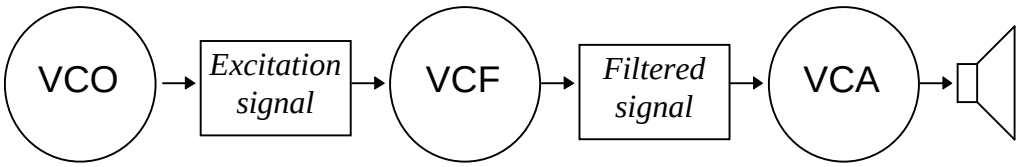

Figure 1: An analog signal chain. Throughout this paper, circles represent functional units, and rectangles represent signals or data.

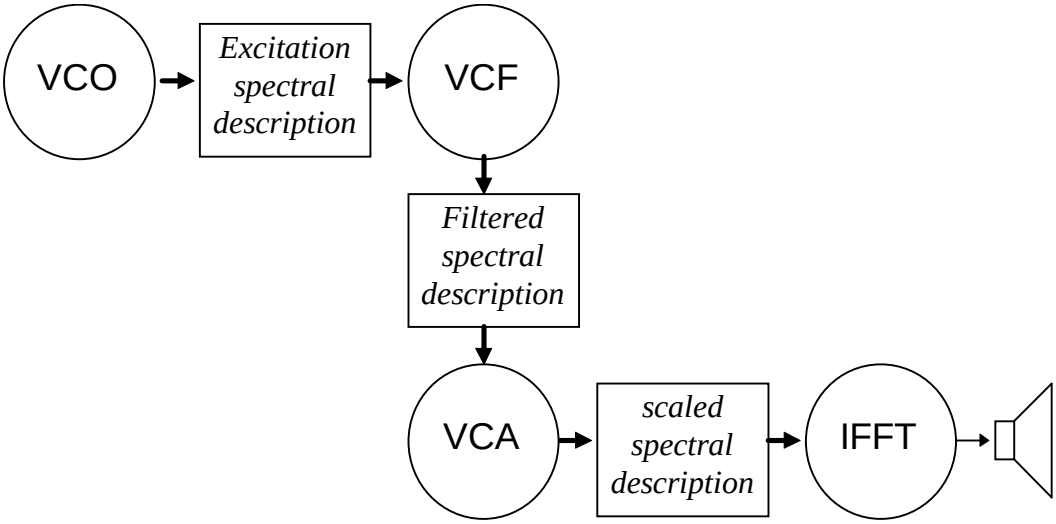

Figure 2: A spectral-domain version of the analog signal chain from figure 1. The signals have been replaced by spectral descriptions. The last spectral description is converted to an audio signal using an Inverse Fast Fourier Transform (IFFT).

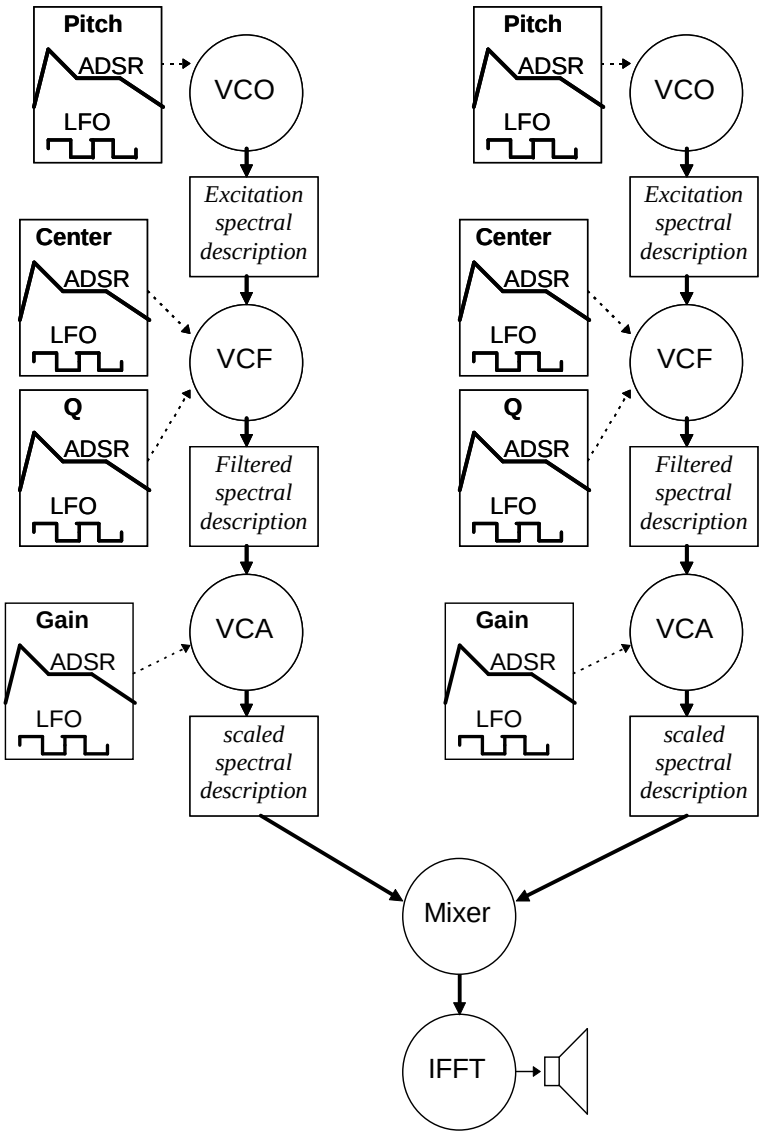

Figure 3: The signal chain and control structure of the two-voice analog synthesizer implementation. Each control parameter has its own ADSR envelope and LFO.

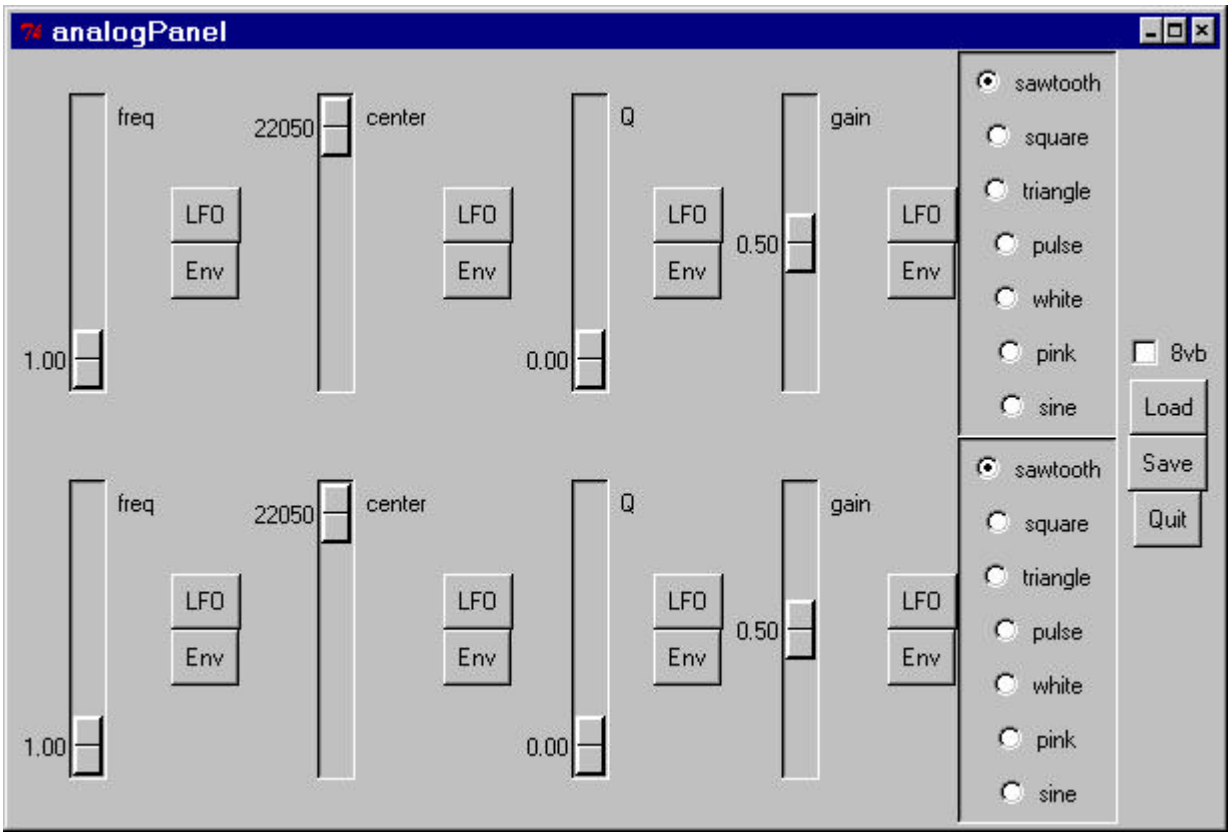

Figure 4: The main interface panel of the analog synthesizer. Controls for pitch, center frequency, Q, gain and oscillator waveform are included for each voice. Buttons are included to pop up an LFO or envelope editing panel for each control parameter.