BRAHIM HOME

Modeling Analog Synthesis with DSPs

John Lane,* Dan Hoory,† Ed Martinez, and Patty Wang****
Motorola, Inc.
*400 Wilson Avenue
Satellite Beach, Florida 32937, USA
jlane@dsp.sps.mot.com
†dar.h@equator.com
**6501 William Cannon Drive, West
Austin, Texas 78735, USA
edm@dsp.sps.mot.com
pwang@dsp.sps.mot.com

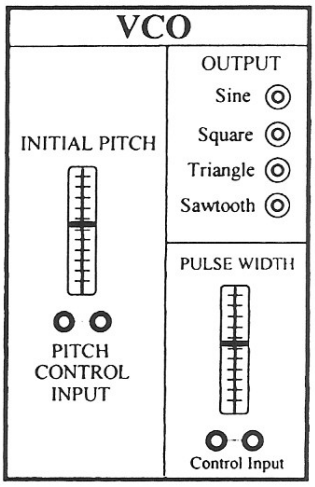
During the 1960s and early 1970s, practical music synthesis was implemented in analog electronics using subtractive synthesis techniques. Since that ten-year era, subtractive synthesis has been replaced by frequency modulation, wavetable, and sampling synthesis techniques, all of which lend themselves to digital processing systems. Subtractive synthesis derives its name from the way in which waveforms are generated. When a waveform with many harmonic partials is filtered by a parametric filter, the result can be a useful and interesting sound element. The real power in this technique is its extreme simplicity. Subtractive synthesis systems often produce a more creative environment for the user than do many of the digital synthesizers of recent times. The loss in popularity of this method was primarily due to the rapid advent of digital signal processing (DSP) and newer synthesis algorithms. It has often been a goal of electronic music systems to simulate the sound of acoustic instruments. In this sense, subtractive synthesis was a failure. However, by listening to the most successful compositions from the days of subtractive synthesis music, one can see that the real success of this technique was in creating original sounds, unlike any known physical instrument.

It is the goal of this article to describe practical DSP algorithms that duplicate the characteristics of subtractive synthesis by using the parametric form of the infinite impulse response (IIR) filter. The primary components of these early synthesizers were the voltage-controlled oscillator (VCO), the voltage-controlled amplifier (VCA), the voltage-controlled filter (VCF), the attack, decay, sustain and release (ADSR) envelope generator, and the low-frequency oscillator (LFO). In addition to these, other processing components included the audio mixer, graphic equalizer, parametric equalizer, and keyboard interface. A secondary goal is to demonstrate a few of the numerous applications of the parametric IIR filter. In many ways, the IIR filter structure represents a link between the analog and the digital worlds. Its behavior mimics its analog counterpart, but it can be directly controlled by digital means. The parametric version of the IIR filter lends itself to simple yet precise and effective control of filtering characteristics.

The following sections describe DSP algorithms that model the behavior of analog voltage-controlled modules commonly used in subtractive synthesis systems. In a DSP implementation, all input, output, and control signals of a module are discrete samples, rather than continuous analog signals as in the voltage-controlled synthesizer. However, the functionality of each DSP module has been matched with its voltage-controlled counterpart, so that from a system point of view, the only difference is that continuous signals are replaced with discrete signals. In some cases, such as the VCA (to be described), it is hardly worth defining a DSP counterpart, since the VCA is simply a single-multiply instruction in a software implementation. However, to maintain a faithful reproduction of the subtractive synthesis system, a DSP version of each voltage-controlled module will be defined. The phrase "voltage controlled" will be used to describe the module functionality, whether the analog or DSP implementation is implied. From a terminology standpoint, "voltage" could be replaced with

Computer Music Journal, 21:4, pp. 23–41, Winter 1997

Figure 1. Typical VCO block.



"Z-transform," so, for example, VCO would become ZCO. However, the terminology convention will be to use VCO for both the analog and digital modules. Another obvious difference between the two technologies is that patching between modules is done by connecting wires in an analog system and by passing software parameters in the DSP system.

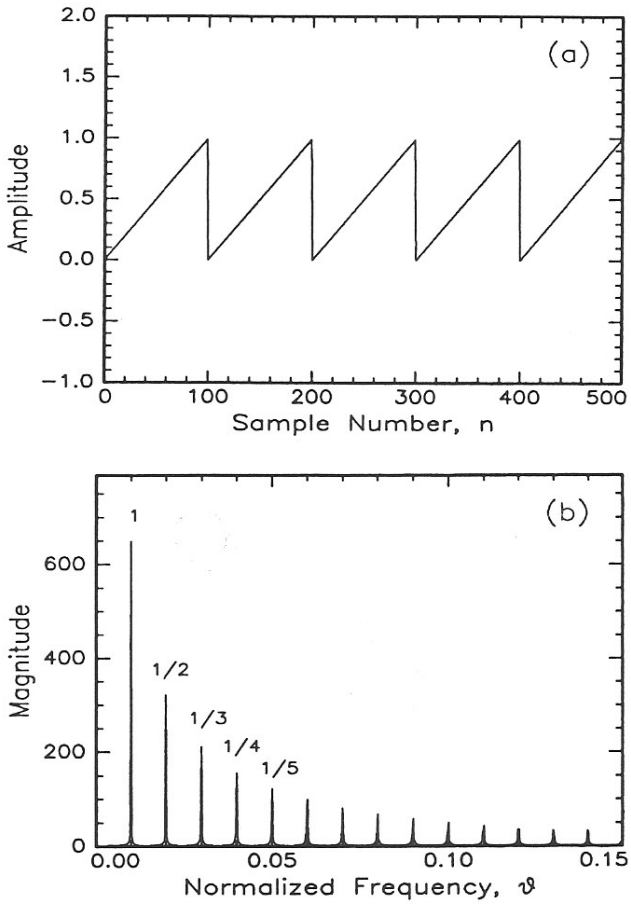
The Voltage-Controlled Oscillator

As shown in Figure 1, the traditional analog voltage-controlled oscillator (VCO) module includes the following output waveforms: square, sine, triangular, and sawtooth (Chamberlin 1987). In addition, the square wave can usually be modified by a pulse-width modulation (PWM) ratio that ranges from 0 to 1, where 0.5 yields a symmetrical square wave, that is, a 50-percent duty cycle. The VCO input is a control voltage that makes the output frequency proportional to the control-voltage level.

To duplicate the behavior of an analog VCO in a digital processing system, it might seem logical to implement a software waveform generator that simply produces the desired wave shape. For example, the sawtooth waveform shown in Figure 2a can be generated by the following formula:

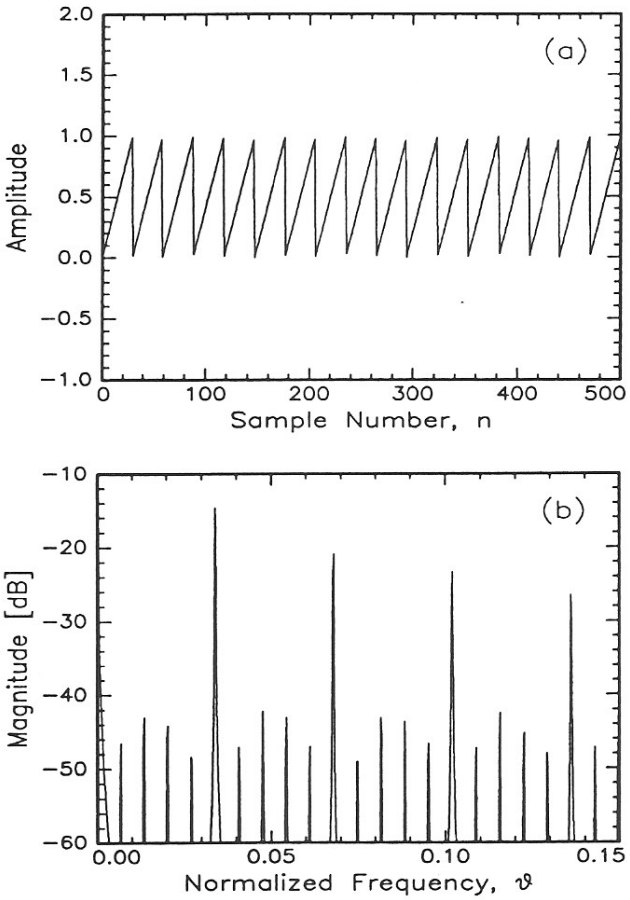
$$s[n] = A[nf_0/f_s - Int(nf_0/f_s)] \quad (1)$$

Figure 2. Sawtooth waveform generated from Equation 1 with f = 441 Hz and f_s = 44,100 Hz: (a) time domain; (b) linear-magnitude frequency spectrum as generated by a 4,096-point Fast Fourier Transform (FFT).



where A is the amplitude, n is the sample number, f_0 is the frequency, and f_s is the sample frequency. The *Int* function truncates the value of nf_0/f_s to the nearest integer. The magnitude spectrum of the sawtooth wave from Figure 2a is shown in Figure 2b. As expected, the amplitude of the harmonics is proportional to $1/m$, where m is the harmonic number. In the example shown in Figures 2a and 2b, f_0 = 441 Hz and f_s = 44,100 Hz. The problem with Equation 1 as a waveform generator becomes apparent when choosing a frequency of f_0 = 1,500 Hz, as shown in Figures 3a and 3b. The magnitude spectrum of Figure 3b shows considerable contamination from aliasing. In a typical DSP system, an anti-aliasing filter (a low-pass filter with a sharp cutoff) is used to suppress these unwanted compo-

Figure 3. Sawtooth waveform generated from Equation 1 with f = 1,500 Hz and f_s = 44,100 Hz: (a) time domain; (b) magnitude frequency spectrum in decibels (note aliasing contamination).

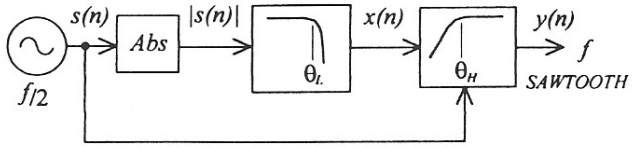


nents. However, as can be seen by examining Figure 3b, high frequencies can be filtered, but the low-frequency contamination is permanent.

The DSP Algorithm Solution

A solution for this problem is shown in the block diagram of Figure 4. The strategy of this approach is to generate a waveform where the harmonic partials fall off rapidly with harmonic number m. Applying a low-pass "brick-wall" filter then produces a waveform that can be further processed by a first-order, high-pass (with gain) tracking filter, so that the harmonic content of the signal is boosted, while the aliasing components are effectively suppressed. The result is a useful digital subtractive-synthesis waveform that has a frequency spectrum similar to a band-limited version of the sawtooth waveform. Figures 5a and 5b show the time and frequency results for f_0 = 441 Hz. (The low-pass filter is a fourth-order Butterworth filter with a cutoff frequency of 8,000 Hz.) The waveforms in Figures 5a and 5b are very similar to those in Figures 2a and 2b, with the harmonics falling off as approximately $1/m$ with increasing harmonic number. The high-pass cutoff frequency is set to $f_c = 16f_0$, so that up to 16 harmonics are boosted by approximately 6 dB/octave.

Figure 4. Block diagram for generating a band-limited approximation of a sawtooth waveform.



Setting f_0 = 1,500 Hz, the time and frequency waveforms are shown in Figures 6a and 6b. When these are compared to Figures 3a and 3b, it is evident that most of the aliasing contamination has been eliminated. To further reduce the aliasing components in this waveform, the low-pass filter characteristics can be adjusted by reducing the cutoff frequency and/or increasing the filter order. The ordering of the low-pass, brick-wall filter and first-order, high-pass filter of Figure 4 has no effect on the outcome of the signal processing. However, there may be subtle differences due to numerical round-off and truncation. The effect of the low-pass filter is to truncate the frequency response so that the aliasing components in the vicinity of the fold-over frequency (i.e., the Nyquist frequency, equal to $f_s/2$) are eliminated. Of course, all frequencies above the low-pass cutoff are eliminated, in addition to the aliasing components.

Fourier-Series Analysis of a Sawtooth Approximation

To better understand how the network in Figure 4 approximates a sawtooth, it is useful to compare the Fourier series of the sawtooth waveform $s_T(x)$ with the equivalent series for $|\sin(x/2)|$, as shown in

Figure 5. Sawtooth waveform generated from the network in Figure 4 with f = 441 Hz and f_s = 44,100 Hz. The low-pass filter is fourth-order (Butterworth) with cutoff frequency of 8,000 Hz: (a) time domain; (b) linear-magnitude frequency spectrum.

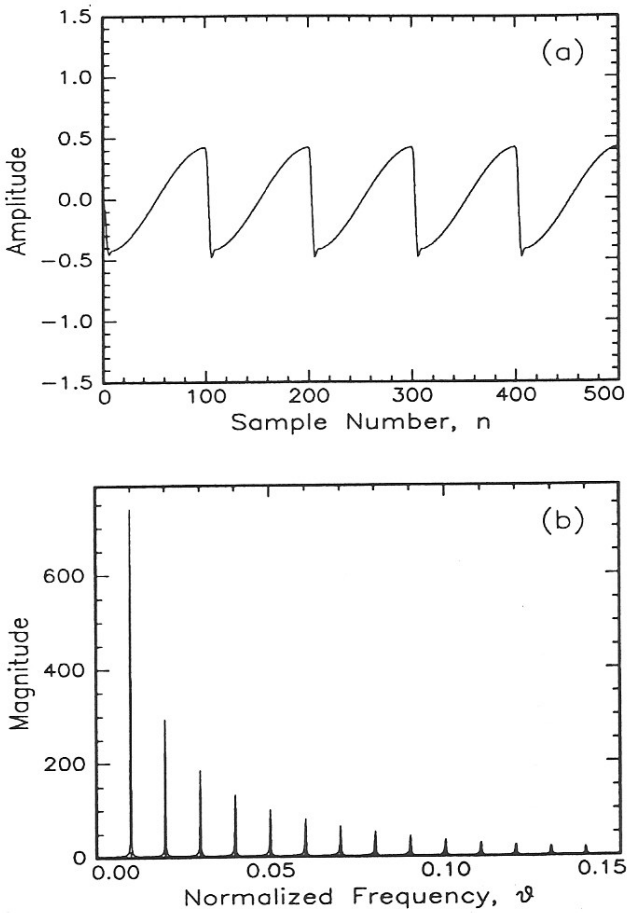


Figure 6. Sawtooth waveform generated from the network in Figure 4 with f = 1,500 Hz and f_s = 44,100 Hz. The low-pass filter is eighth-order (Butterworth) with cutoff frequency of 12,000 Hz): (a) time domain; (b) magnitude frequency spectrum in decibels.

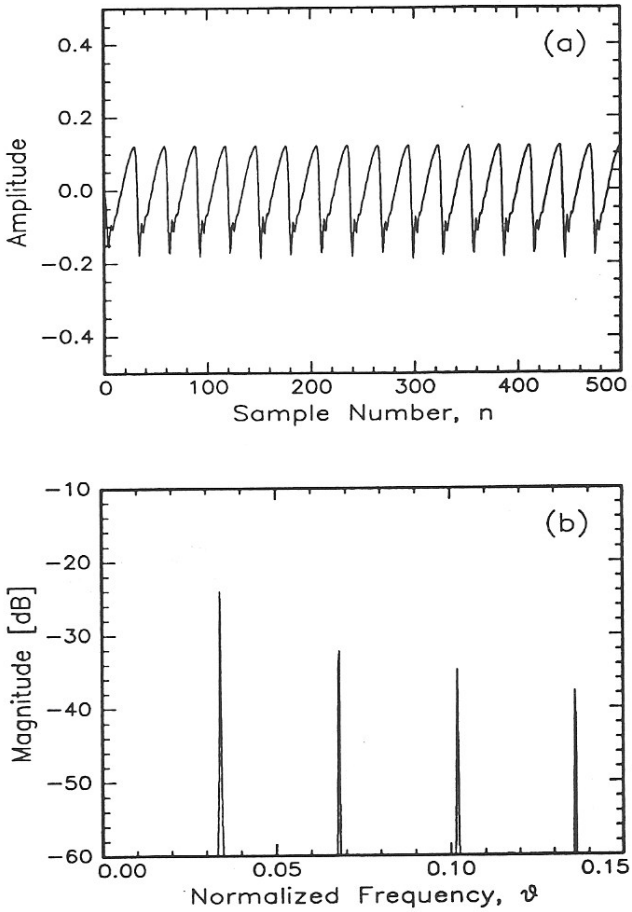


Table 1. Ignoring the DC component, the Fourier series coefficient of $|\sin(x/2)|$ is proportional to $[(m + 1/2)(m - 1/2)]^{-1}$, where m is the harmonic number (or Fourier coefficient index) of the fundamental frequency f_0 (Mathews and Walker 1970). Note that x in Table 1 can be replaced with $x = 2\pi f_0 t$ for the continuous time domain, or $x = 2\pi f_0 n/f_s$ for the discrete time domain. The first-order, high-pass filter in Figure 4 has a frequency response approximately equal to mf_0/f_c in the stop-band, where f_c is the high-pass-filter cutoff frequency. Factoring in the $\pi/2$ phase shift of the high-pass filter (Lane 1990), the resulting Fourier components approach those of the sawtooth waveform $s_T(x)$, proportional to $1/m$, as shown in the last row of Table 1.

High-Pass Tracking Filter

The first-order, high-pass filter in Figure 4 is implemented as a difference equation, and is derived directly from the transfer function of the high-pass network (Lane 1990):

$$\textit{Transfer Function:} \quad H(z) = \frac{\alpha(1 - z^{-1})}{1 - \gamma z^{-1}} \qquad (1)$$

Table 1. Fourier series of sawtooth waveform, absolute value of sine, and high-pass filtered absolute value of sine

Periodic Function	*Fourier Series*
$s_T(x) = \frac{x}{2\pi}, -\pi < x < \pi$	$\frac{1}{\pi}\left(\frac{\sin x}{1} + \frac{\sin 2x}{2} + \frac{\sin 3x}{3} + \ldots\right)$
$\lvert\sin(x/2)\rvert$	$\frac{2}{\pi} - \frac{4}{\pi}\left(\frac{\cos x}{1\cdot 3} + \frac{\cos 2x}{3\cdot 5} + \frac{\cos 3x}{5\cdot 7} + \ldots\right)$
$HPF\{\lvert\sin(x/2)\rvert\}$	$\frac{f_0}{\pi f_c}\left(\frac{1\cdot\sin x}{(1+1/2)\cdot(1-1/2)} + \frac{2\cdot\sin 2x}{(2+1/2)\cdot(2-1/2)} + \frac{3\cdot\sin 3x}{(3+1/2)\cdot(3-1/2)} + \ldots\right.$
	$\approx \frac{f_0}{\pi f_c}\left(\frac{\sin x}{1} + \frac{\sin 2x}{2} + \frac{\sin 3x}{3} + \ldots\right)$

Difference Equation:

$$y(n) = \alpha[x(n) - x(n-1)] + \gamma y(n-1) \quad (2b)$$

Filter Coefficients: $\gamma = \frac{\cos\theta}{1+\sin\theta_c} \quad \alpha = \frac{1+\gamma}{2}$ (2c)

where $\theta_c = 2\pi f_c/f_s$ is the normalized cutoff frequency. Since we are interested in the stop-band characteristics where the transfer function has a magnitude response of mf_0/f_c, the cutoff frequency needs to be high enough to yield this response for many of the harmonic partials of f_0. Setting the cutoff $f_c = 16f_0$ satisfies this requirement, resulting in a filter stop-band response of $m/16$. Note that the factor of 1/16 leads to a –24 dB reduction in output. To produce a waveform that has a normalized amplitude, a constant of 24 dB gain (factor of 16) is needed at the output of the high-pass filter.

Taylor-Series Approximation for High-Pass Coefficients

The filter coefficients described in Equation 2c must be evaluated every time the frequency parameter is modified. Typically, this occurs only when a new note is being played. In the worst case, thirty-second notes (demisemiquavers) may change at a frequency of 10 Hz or so. However, in a subtractive synthesis system, it is typical that the frequency control input to the VCO is modulated by another control signal, such as an LFO for vibrato effects, or perhaps an ADSR for special effects. Regardless of the source of frequency modulation, the frequency may need updating at a much faster rate than 10 Hz. In the case of a 5-Hz LFO modulation, for example, the update rate might be chosen as 500 Hz.

There are two coefficients in Equation 2c that need to be calculated for the high-pass filter: α and γ. The coefficient α involves only an add operation and a right shift (divide by two). The coefficient γ, which involves trigonometric functions as well as a divide operation, would require many instruction cycles if implemented directly from Equation 2c. A good approximation for γ can be evaluated using N terms of a Taylor-series expansion (TSE):

$$\gamma(\theta_c) = \sum_{k=0}^{N-1} c_k\,\theta_c^k \quad (3a)$$

$$c_k = \frac{d^k\gamma(\theta_c)}{d\theta_c^k}\Big|_{\theta_c=a} \quad (3b)$$

Using the following Mathematica function (Wolfram 1991), the TSE coefficients for γ can easily be determined:

```
Collect[Series[Cos[x]/(1+Sin[x]),{x,Pi/4,3}]
Collect[Series[Cos[x]/(1+Sin[x]),{x,3Pi/4,3}]
```

Table 2. Taylor series coefficients for high-pass filter coefficient γ

Range of θ_c	a	c_0	c_1	c_2	c_3
$0 \le \theta_c < \frac{\pi}{2}$	$\pi/4$	0.984948	−0.913189	0.295542	−0.0739418
$\frac{\pi}{2} \le \theta_c < \pi$	$3\pi/4$	1.2597	−1.24558	0.401343	−0.0739418

To keep the number of terms in the TSE series to a minimum, two sets of TSE coefficients can be used, as shown in Table 2. The TSE approximation requires far fewer instructions than the exact formula of Equation 2c. Figure 7 shows Motorola DSP56002 code for calculating the high-pass filter coefficients. Note that `Freqp` is equal to $f_0/(2f_s)$. Also, if $16f_0 > 0.45f_s$, then $16f_0$ is replaced with $0.45f_s$.

Oscillator Algorithm

The requirement for any algorithm used in this application is that the resulting frequency output must be very close to the desired frequency, that is, the error E_f must not exceed some maximum value, usually expressed in cents:

$$f_A = 2^{c/1200} f \qquad (4a)$$

$$E_f = 1200 \log_2 \frac{f_A}{f} = c \qquad (4b)$$

where f is the desired frequency, f_A is the actual frequency produced by the oscillator, and $c = 100$ is equivalent to one half-step on the Equal-tempered scale. If the error produced by the oscillator is within a few cents, most listeners will not be able to distinguish the difference (Pierce 1983).

The sine generator of Figure 4 can be implemented by a number of methods, most of which can be categorized as either recursive or non-recursive. Non-recursive methods generally maintain a current phase value $\phi(n)$, which is modulo 2π, by a method such as the following:

$\phi(n) = \phi(n-1) + \theta_0$
If $\phi(n) \ge 2\pi$ Then $\phi(n) = \phi(n) - 2\pi$

where $\theta_0 = 2\pi f_0 / f_s$. The phase value $\phi(n)$ is then used to calculate the current value of $\sin[\phi(n)]$ by a lookup table (with some interpolation method for improved accuracy), or is used in a power series such as the Taylor series or Chebyshev polynomial approximation (Press et al. 1989). The non-recursive methods have one particular advantage over the recursive methods: there are no discontinuities in the sine approximation when the frequency is abruptly changed, such as in the case of playing a sequence of notes. The disadvantages of the non-recursive methods are that they have less accuracy and they require more instructions to perform the algorithm.

One particularly interesting recursive sine-generation algorithm can be derived by finding a closed-form solution for the recursive sequence of Chebyshev polynomials. The results can be stated as:

$$T_n(\cos\theta) = 2\cos\theta\ T_{n-1}(\cos\theta) - T_{n-2}(\cos\theta) = \cos(n\theta) \qquad (5a)$$

$$T_0(\cos\theta) = 1$$
$$T_1(\cos\theta) = \cos\theta \qquad (5b)$$

where $\theta = 2\pi f / f_s$. If $s(n)$ in Figure 4 is initialized to the values of Equation 5b, $s(0) = 1$, and $s(1) = \cos\theta$, then the recursive formula from Equation 5a produces $\cos(n\theta)$:

$$s(n) = 2\cos\theta\, s(n-1) - s(n-2). \qquad (5c)$$

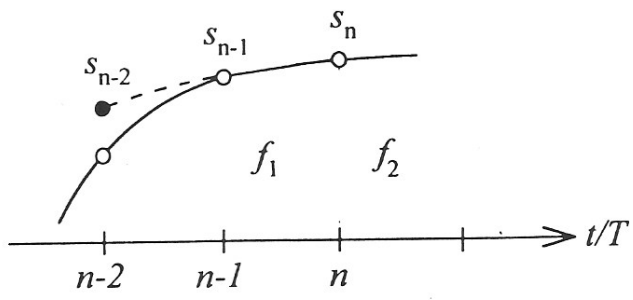
Choosing different initial conditions in Equation 5c will result in changing the amplitude and phase of $s(n)$. For example, if $s(0) = 0$ and $s(1) = \sin\theta$, then $s(n) = \sin(n\theta)$.

Figure 8 illustrates the key to understanding what happens in Equation 5c when the frequency-

Figure 7. DSP56002 code for calculating high-pass filter coefficients.

```
; Design HPF with a normalized cutoff frequency, fc = 32 * Freqp:
;   gamma = Cos(fc)/(1+Sin(fc))
;   alpha = gamma/2 + 0.5
                                    ; (1) Calculate fc:
        move    x:Freqp,a           ; Get Freqp = Freq / 2
        rep     #5                  ; Multiply Freqp by 2^5 = 32
        asl     a                   ; a = 32 * Freqp = 16 * Freq
        move    #0.45,x0            ; Maximum normalized fc is 0.45
        cmp     x0,a    #$20,y0     ; Compare 16 * Freq with maximum fc
                                    ; Move $200000 = 0.25 into y0 for later
        tgt     x0,a                ; If (a > 0.45) then a = 0.45, i.e.,
                                    ;   if 16 * Freq > maximum fc then set
                                    ;     16 * Freq = maximum fc
                                    ;
                                    ; (2) Approximate gamma using a 2-region
                                    ;     3-term normalized TSE table:
                                    ;     Region I:  0 < a < 0.25
                                    ;     Region II: 0.25 <= a <= 0.5
        cmp     y0,a    #hpf_tbl1,r1 ; Determine region of fc, and set
        move    #hpf_tbl2,r2        ;   address register r1 to start of
        tgt     y0,b    r2,r1       ;   TSE table-1 if in region 1, or to
                                    ;   TSE table-2 if in region 2
        move    a,x0                ; Set x0 = fc
        move    a,x1    y:(r1)+,b   ; Set x1 = fc, and b = TSE coef, c[0]
                                    ; Perform 3-term TSE calculation,
                                    ;   for k = 1 to 3
        do      #3,hpf_tse          ;   k=1:
          mpyr    x0,x1,a y:(r1)+,y0 ;     a = fc^2              y0 = c[1]
          mac     x0,y0,b a,x0      ;     b = c[0] + c[1] * fc  x0 = fc^2
    hpf_tse                         ;   k=2
                                    ;     a = fc^3              y0 = c[2]
                                    ;     b = b + c[2] * fc^2   x0 = fc^3
                                    ;   k=3
                                    ;     a = fc^4              y0 = c[3]
                                    ;     b = b + c[3] * fc^3   x0 = fc^4
        rep     #5                  ; Scale result by 32 due to TSE coef definition
        asl     b       #$40,a      ; b = 32*{c[0]+c[1]*fc+c[2]*fc^2+c[3]*fc^3}
                                    ; a = $400000 = 0.5
        move    b,x:hpf_cof+1       ; hpf_cpf[1] = b = gamma
                                    ;
                                    ; (3) Calculate alpha:
        addr    a,b                 ; b = a + b/2 --> b = 0.5 + gamma/2
        move    b,x:hpf_cof         ; hpf_cof[0] = b = alpha
```

Figure 8. Modification of filter state $s(n - 2)$ when changing frequency in Equation 5c.



control input is modified. At sample time n, a new frequency f_2 is presented to the algorithm of Equation 5c. To prevent a discontinuity in the waveform, which can also lead to an inadvertent change in output amplitude, the filter state $s(n - 2)$ must be modified to satisfy the following equation:

$$s(n - 2) = \cos\{\cos^{-1}[s(n - 1)] - \theta_2\}. \quad (5d)$$

The next waveform sample $s(n)$ is calculated using $\theta_2 = 2\pi f_2/f_s$ and Equation 5d in Equation 5c. This procedure guarantees that the discontinuity due to the change in frequency will have a minimal effect. Note that in the example of Figure 8, f_2 is much less than f_1.

Another recursive method that generates a quadrature output, that is, sine and cosine, and does not have the continuity requirement of Equation 5d, can be derived from basic trigonometric identities:

$$\sin n\theta = \sin[\theta + (n - 1)\theta] = \cos\theta \sin[(n - 1)\theta] + \sin\theta \cos[(n - 1)\theta] \quad (6a)$$

$$\cos n\theta = \cos[\theta + (n - 1)\theta] = \cos\theta \cos[(n - 1)\theta] + \sin\theta \sin[(n - 1)\theta]. \quad (6b)$$

If c_0 is defined as $\cos\theta$, and s_0 is defined as $\sin\theta$, then the following recursive pair generates a quadrature output:

$$s(n) = c_0 s(n - 1) + s_0 c(n - 1) \quad (6c)$$

$$c(n) = c_0 c(n - 1) - s_0 s(n - 1) \quad (6d)$$

The initial conditions used to start Equations 6c and 6d are simply $s(0) = 0$ and $c(0) = 1$.

The disadvantage of the oscillator algorithm defined by Equations 6c and 6d is that the amplitude will eventually decay (or possibly grow) due to numerical error—in particular, the error in the coefficients c_0 and s_0. Experience has shown that the initial conditions need to be reapplied once every minute or so when using a 24-bit DSP, such as the DSP56002, to avoid any noticeable amplitude drift. This problem would be much worse and possibly intolerable when using a 16-bit DSP device, due to the increased numerical round-off error. Figure 9 shows DSP56002 code used to implement the quadrature oscillator defined by Equations 6c and 6d. Note that the frequency coefficients $c0$ and $s0$ can be sent as input from a host computer, or can be calculated on the DSP to a relatively high precision using the Taylor-series approximation (Lane et al. 1995).

Square and Triangle Waveform Approximations

The Fourier components of the combined signal $|\sin(x/2)| + g|\sin x|$, as shown in Figure 10, results in an approximate cancellation of even harmonics when the value of g is in the range of –0.25 to –0.20. Using the results from Table 1, the Fourier coefficient a_m of the mth even harmonic of the combined signal is:

$$a_m = \frac{4g}{\pi}[(m + 1)(m - 1)]^{-1} + \frac{1}{\pi}\left[\left(m + \frac{1}{2}\right)\left(m - \frac{1}{2}\right)\right]^{-1}, \quad m = 2, 4, 6 \ldots \quad (7)$$

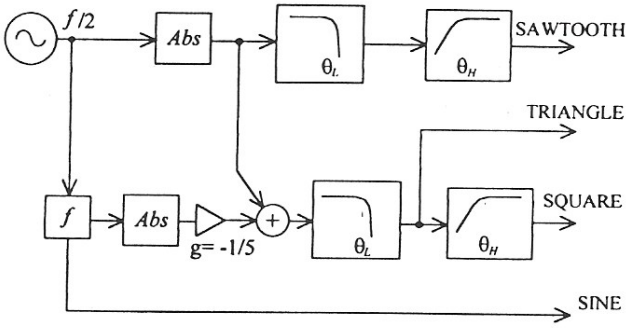
For a value of $g = -0.2$, the $m = 2$ is completely canceled, while the remaining even harmonics are reduced. Alternatively, a value of $g = -0.25$ approaches exact cancellation as the even-harmonic number gets large, with only a finite reduction in the second harmonic. By varying the factor g, the effect is similar to the pulse-width modulation (PWM) of a square wave, since PWM results in adding a finite number of even harmonics to the odd-harmonic spectrum of the square wave.

The two oscillators of Figure 10 are implemented

Figure 9. DSP56002 code for quadrature oscillator.

```
;--- update sinewave generators ---
proc move     y:cn,x0                  ; get c(n-1)
     move     y:sn,y0                  ; get s(n-1)

     move     y:s0,y1
     mpy      x0,y1,b y:c0,x1          ; b = s0*c(n-1)
     macr     x1,y0,b                  ; b = s0*c(n-1) + c0*s(n-1)
     mpy      x0,x1,a b,x0             ; a = c0*c(n-1)
     macr     -y0,y1,a                 ; a = c0*c(n-1) - s0*s(n-1)
     abs      a              a,x1

     move     b,y:sn                   ; s(n) = b
     abs      b              x1,y:cn   ; c(n) = a
```

Figure 10. Block diagram for generating band-limited approximations of the four basic VCO output signals.



as one single quadrature oscillator. Using Equations 6c and 6d, the $f/2$ oscillator is implemented where both $\sin(n\theta/2)$ and $\cos(n\theta/2)$ are available. Using a trigonometric identity, the f oscillator is directly calculated as:

$$\sin n\theta = 2\cos(n\theta/2)\sin(n\theta/2). \quad (8)$$

The waveform signal at the input of the high-pass filter in the lower part of Figure 10 is a close approximation of a band-limited triangle wave. The output of the high-pass filter is a band-limited version of a square wave. Figures 11a through 11h show the results of this method with f = 441 Hz, using a sixth-order low-pass filter with f_c = 9 kHz.

The Voltage-Controlled Amplifier

The DSP equivalent of the analog voltage-controlled amplifier (VCA), as shown in Figure 12, is simply a multiply instruction. For the 56000 family of Motorola digital signal processors (Motorola 1992), the multiply (MPY) syntax is:

```
MPY S1,S2,D
```

which is equivalent to the C syntax:

```
D = S1 * S2
```

where S1 and S2 are 24-bit registers and D is a 56-bit register. If S1 is set to a sample value $x(n)$, and S2 is set to a control "voltage" level $c(n)$, the output of the MPY process is the product of the two:

$$y(n) = c(n)x(n). \quad (9a)$$

In this way, the standard VCA process can be implemented on a sample-by-sample basis with one DSP instruction. The control signal $c(n)$ in Equation 9a is assumed to be always positive, as in the case of an analog VCA (Graham 1978), so that Equation 9a becomes a two-quadrant multiplication:

$$y(n) = |c(n)|x(n). \quad (9b)$$

Some VCAs include a balanced modulator option (PAIA 1974), which is a true four-quadrant multiplication, as originally described by Equation 9a. As is true with the analog VCA/balanced-modulator equivalent, the two multiplier inputs can be an audio/control pair, an audio/audio pair, or a control/control pair, although the first case is most typical.

Figure 11. VCO waveforms generated by network of Figure 10 with f = 440 Hz and a sixth-order, low-pass filter with f_c = 9 kHz: (a) sine wave; (b) FFT of sine wave; (c) triangle wave; (d) FFT of triangle wave; (e) sawtooth wave; (f) FFT of sawtooth wave; (g) square wave; (h) FFT of square wave.

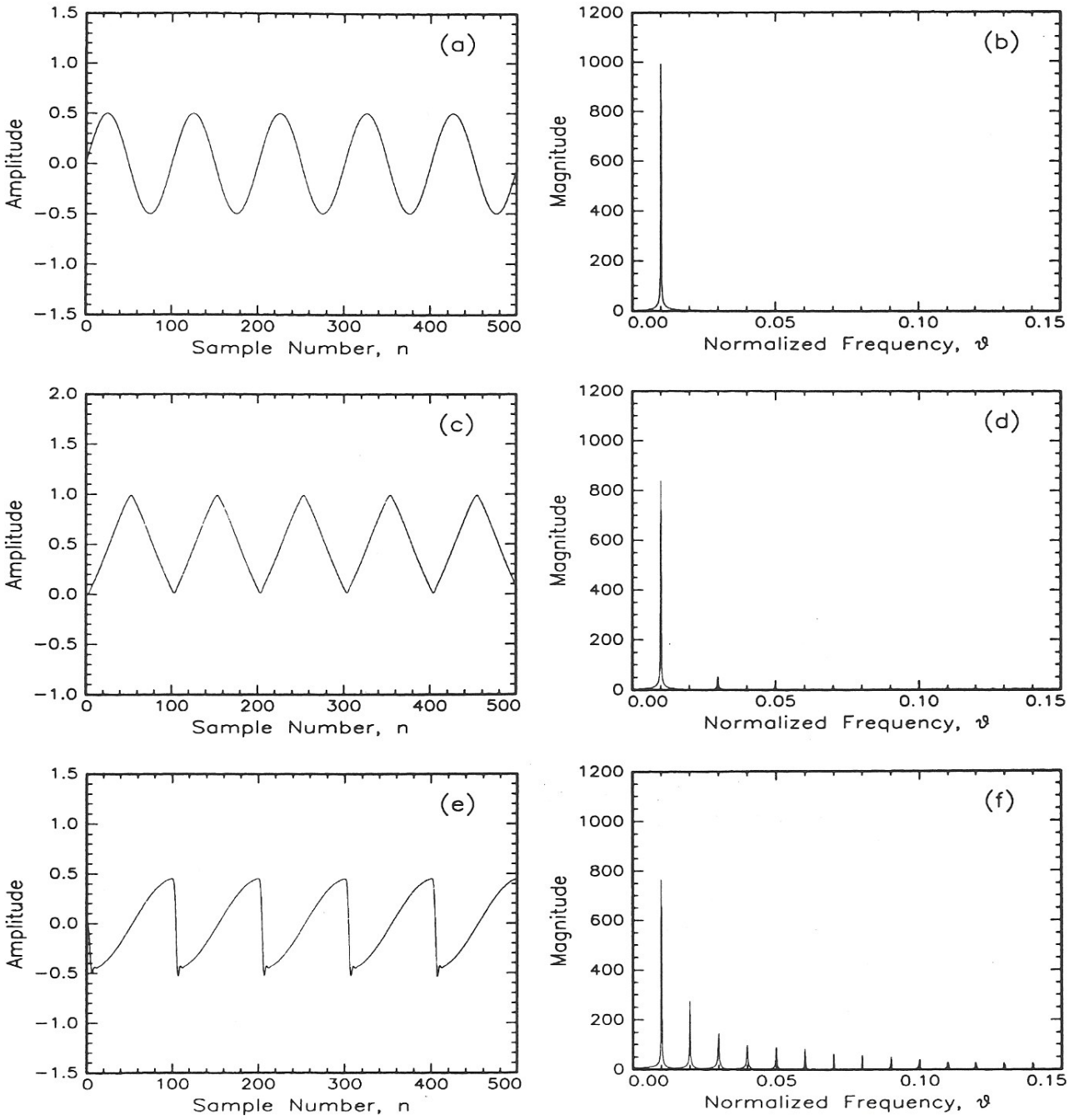


Figure 11. continued

Figure 12. Typical VCA block.

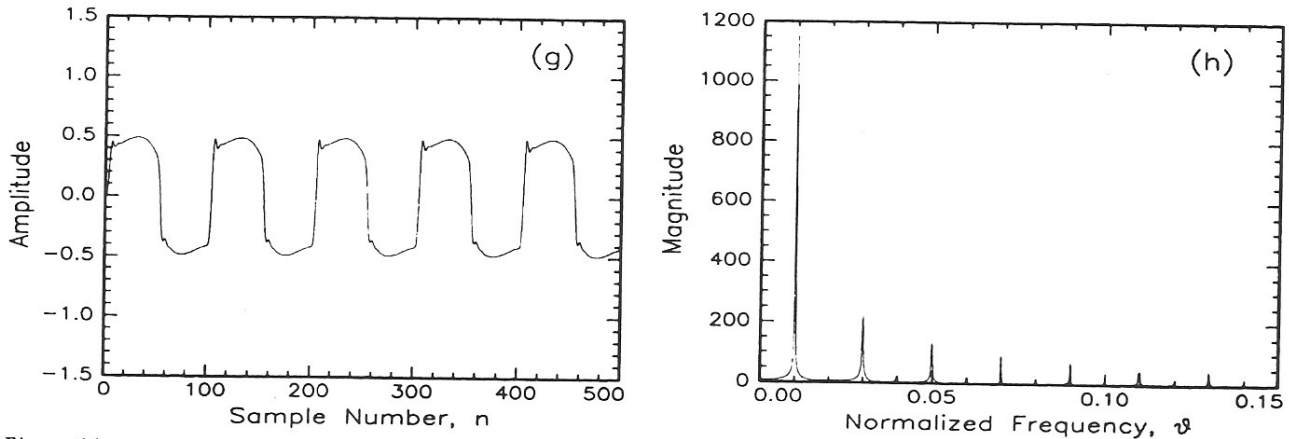


Figure 11

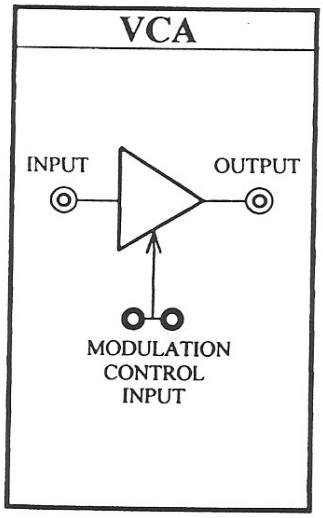


Figure 12

The Low-Frequency Oscillator

The low-frequency oscillator (LFO) is used as a control input to the VCO, VCA, or VCF. As shown in Figure 13, the LFO consists of two sections: a VCA and a VCO. The VCA section is used to adjust the output amplitude of the LFO signal, while the VCO section generates the output sine wave, and also provides a frequency-control input. In the DSP implementation of the LFO, the output VCA is simply a scaling multiply operation. The LFO could be duplicated by combining a VCO and VCA. However, in either a hardware or a software implementation, the cost is minimized by combining these functions into one and eliminating extra VCO features, such as the square, triangle, and sawtooth outputs.

Another way of reducing cost (that is, saving millions of instructions per second, or MIPS) is to implement a "distorted" sine wave by utilizing the limiting nature of the DSP. Since recursive oscillators have poles on the unit circle, shifting the pole slightly outside of the unit circle by making the pole radius a little larger than 1.0 results in a steadily growing oscillation. However, whenever a number exceeds 1.0 on the DSP56002 (or any 24-bit DSP based on a fixed-point fractional number system), the value is limited to $7fffff = 0.9999999 when it is stored in memory (note that both accumulators allow values from –256 to 255.9999999). Therefore, as the oscillation tries to grow, it is limited to a maximum amplitude of 0.9999999, but some distortion is introduced (the frequency spectrum contains harmonic components of the fundamental). The amount of distortion is dependent on the oscillator frequency, the sample frequency, and the amount by which the pole radius exceeds unity. In general, the LFO requirements are not very strict in the sense of requiring low harmonic distortion. Therefore, a good, low-MIPS LFO algorithm is im-

Figure 13. Typical LFO block.

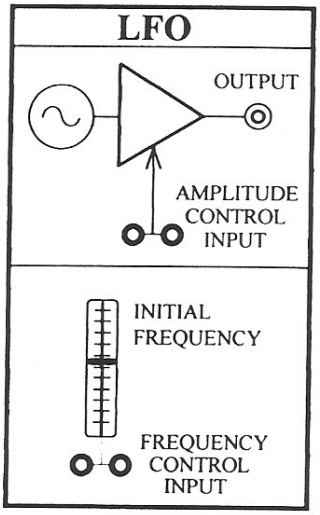
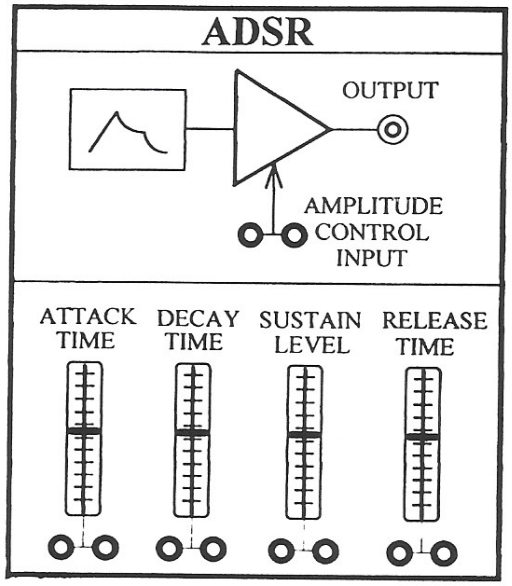


Figure 14. Typical ADSR block.



plemented using Equations 6c and 6d by approximating $c_0 = 1 \approx cos\ \theta_0$ and $s_0 = \theta_0 \approx \sin \theta_0$, resulting in the following recursive pair:

$$s(n) = s(n-1) + \theta_0\ c(n-1) \qquad (10a)$$

$$c(n) = c(n-1) - \theta_0\ s(n-1). \qquad (10b)$$

The initial conditions used to start the LFO oscillator defined by Equations 10a and 10b are simply $s(0) = 0$ and $c(0) = 1$. One advantage of the oscillator formulas of Equations 10a and 10b is that the oscillation will not decay and is therefore inherently stable, unlike the VCO formulas of Equations 6c and 6d, which require periodic reinitialization. Also, like the comparable VCO formulas, no discontinuities are created when the frequency is abruptly changed.

The ADSR Envelope Generator

To impose a percussive characteristic onto a VCO output (or any other signal source), the attack, decay, sustain, and release (ADSR) envelope generator is used (Graham 1978; Chamberlin 1987). The output of this processing block (see Figure 14) is a control signal used to apply an envelope to the signal source. For example, this process can convert an organ-like sound into a piano-like sound. Typically, the ADSR output is the control input of a VCA block, where the VCA input is connected to the VCO output.

The ADSR waveform consists of four parameters, three of which are time intervals. The first time interval is the attack time. As shown in Figure 15, the envelope follows a constant slope from 0 to 1.0 in the time interval specified by the parameter A. A has units of time samples at a pre-specified ADSR sample rate, f_A, where f_A is much lower than the signal sample rate f_s. In Figure 15, $f_A = 500$ Hz and $A = 100$ ADSR time samples, which is equal to 0.2 sec. The generating formula for the attack portion of the ADSR control signal is:

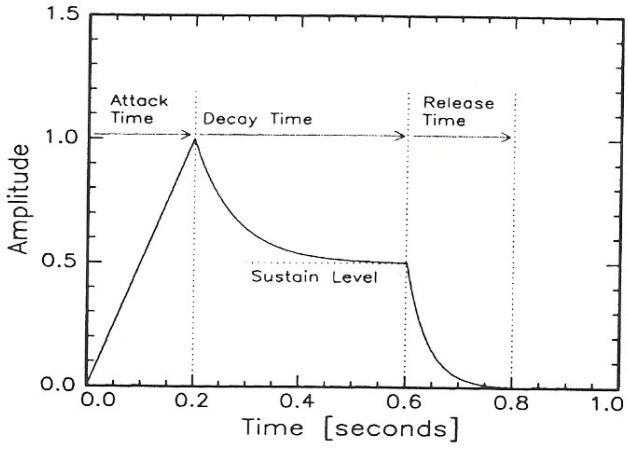
$$c(k) = k/A \qquad k = 0, \ldots, A-1 \qquad (11)$$

where k is the sample number at the ADSR sample rate f_A, and $c(k)$ is the ADSR control output signal.

The second portion of the ADSR waveform is the decay section, which is specified by the decay-time and sustain-level parameters, as shown in Figure 15. The generating formula for this segment of the envelope functions is:

$$c(k) = \gamma_D c(k-1) + (1 - \gamma_D)S, \quad k = A, \ldots, A + D - 1 \qquad (12a)$$

Figure 15. Typical ADSR control signal.



where γ_D is a decay-rate parameter, D is the decay time in ADSR sample units, and S is the sustain level ($0 \leq S < 1$). For the example shown in Figure 15, $\gamma_D = 0.975$, $D = 200$, and $S = 0.5$. A closed-form solution of Equation 12a can be found by utilizing the Z-transform:

$$c(k) = \gamma_D^{k-A}(1 - S) + S, \quad k = A, \ldots, A + D - 1. \quad (12b)$$

If an ADSR sample time k is specified ($k = 280$, for example) at which point the contribution to $c(k)$ from the first term on the right-hand side of Equation 12 is desired to be at 99 percent of its final value, the decay-rate parameter γ_D becomes:

$$\gamma_D = 10^{-2/(280 - A)} \approx 0.975 \quad (12c)$$

γ_D can be an independent parameter, or it can be defined by D using Equation 12c. For example, $\gamma_D = 10^{-2/D}$ defines the decay-rate parameter so that the ADSR output $c(k)$ will be at 99 percent of its final value following D ADSR samples after the start of the decay segment.

The third and final portion of the ADSR waveform is the release section, specified by the release-time parameter R, as shown in Figure 15. The generating formula for this segment of the envelope function is similar to that for the decay portion, but with a zero final level:

$$c(k) = \gamma_R c(k - 1), \quad k = A + D, \ldots, A + D + R - 1 \quad (13a)$$

where γ_R is a release decay-rate parameter and R is the decay time in ADSR sample units. For the example shown in Figure 15, $\gamma_D = 0.95$, and $R = 100$. A closed-form solution of Equation 13a can be found by again utilizing the Z-transform:

$$c(k) = \gamma_R^{k-A-D}S, \quad k = A + D, \ldots, A + D + R - 1 \quad (13b)$$

where again γ_R can be an independent parameter, or it can be defined by R using the equivalent of Equation 12c. For example, $\gamma_R = 10^{-2/R}$ defines the release decay-rate parameter so that the ADSR output $c(k)$ will be at 99 percent of its final value following R ADSR samples after the start of the release segment.

Equations 11, 12a, and 13a define an ADSR control signal using six parameters. Two of the six, the decay-rate parameters, can be defined as dependent variables so that only four ADSR parameters are used. As shown in Figure 14, the ADSR control signal can be scaled to any value between one and zero using a dedicated VCA section. Figure 16 shows example DSP56002 code for the ADSR algorithm described above.

The Voltage-Controlled Filter

The most important part of the subtractive synthesis system is the voltage-controlled filter (VCF). However, without the VCO, the VCF operation would be useless. A typical VCF block is shown in Figure 17. The DSP implementation of the VCF can be achieved using the filter network shown in Figure 18, along with the corresponding coefficient formulas. In the DSP software version of the VCF, the control signals are represented by f_c and d (or f_o and Q), which are the cutoff frequency and damping factor for a low-pass or high-pass filter (or center frequency and quality factor for a band-pass or band-stop filter). As in a typical analog subtractive synthesis system, these control signals can originate from several possible sources, including the LFO, ADSR, and VCO control input, as well as constant (DC) control values.

The digital filter is implemented in DSP software

Figure 16. DSP56002 code for ADSR.

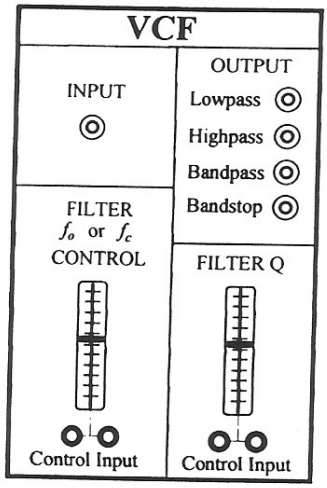
```
ASDR
    move y:ck,y0
    move y:k,a              ; If (k<>0) Then
    tst  a                  ;   y0=ck
    jne  skip_init_k        ; Else
    move #0,y0              ;   y0=0
skip_init_k                 ; Endif
    move a,r5
attack
    move x:asdrA,a
    move r5,x0
    cmp  x0,a               ; a - x0 = A - k
    jle  decay              ; If (k >= A) jump to decay
    move x:invA,b
    add  y0,b               ; b = c(k) = c(k-1) + 1/A
    jmp  adsrEnd

decay                       ; (decay & sustain)
    move x:adsrA,a
    move x:adsrD,x0
    add  x0,a
    move r5,x0
    cmp  x0,a               ; a - x0 = A+D - k
    jle  release            ; If (k >= A+D) jump to release
    move x:gammaD,x0
    mpy  x0,y0,b            ; b = gammaD*c(k-1)
    clr  a
    move #$800000,a1
    sub  x0,a1
    move a,x0               ; x0 = 1 - gammaD
    move x:adsrS,y0
    mac  x0,y0,b            ; b = gammaD*c(k-1) + (1- gammaD)*S
    jmp  adsrEnd
release
    move x:gammaR,x0
    mpy  x0,y0,b            ; b = gammaR*c(k-1)
adsrEnd
    move b,y:ck             ; update c(k)
    move (r5)+
    move r5,y:k             ; update k, k=k+1
```

using the infinite impulse response (IIR) recursion formula, similar to the first-order, high-pass filter discussed in the VCO section:

$$y(n) = 2\alpha x(n) + \alpha_1 x(n-1) + \alpha_2 x(n-1) + \gamma y(n-1) - \beta y(n-2) \quad (14)$$

where the parametric form of the filter coefficients (Lane 1990) is given in Figure 18, as a function of the control parameters f_c and d (or f_o and Q), where the normalized frequency is $\theta_0 = 2\pi f_0/f_s$. Note that the damping factor d is typically set to $\sqrt{2}$ for a maximally flat response. The filter input

Figure 17. Typical VCF block.

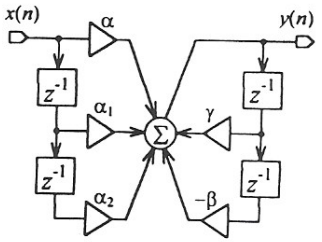


states $x(n-k)$ and filter output states $y(n-k)$ are stored in memory. The filter input is usually the VCO output, while the filter output may be routed to a digital mixer or directly to a digital-to-analog converter for final output.

In many practical applications, the frequency-control input of the VCF is a combination (summation) of an effects control signal and a pitch control signal. The pitch control signal is used to follow the frequency of the VCO tone source so that the spectral characteristics (or timbre) of the filtered VCF output are independent of the VCO pitch. Added to this pitch control input is the effects control signal, which may be an LFO or ADSR output.

One possible strategy for controlling the filter frequency is adding a DC offset to the pitch control signal. This has the effect of offsetting the filter's cutoff frequency (or center frequency) by an amount so that only specific harmonics of the original VCO tone are passed. Another way to filter a select number of partials from the input signal is to multiply the pitch control signal by a constant value. For example, multiplying the pitch control signal by a constant value of 2, and using this as the VCF control input, will cause the filter cutoff (or center) frequency to always be centered about the second harmonic partial of the input signal, regardless of the frequency of the input.

Figure 18. Parametric digital filter network and coefficient formulas.



coefficient	Lowpass	Highpass	Bandpass	Bandstop
α	$\frac{1}{4}\left(\frac{1}{2}+\beta-\gamma\right)$	$\frac{1}{4}\left(\frac{1}{2}+\beta+\gamma\right)$	$\frac{1}{2}\left(\frac{1}{2}-\beta\right)$	$\frac{1}{2}\left(\frac{1}{2}+\beta\right)$
α_1	2α	-2α	0	$-\gamma$
α_2	α	α	$-\alpha$	α
β	$\frac{1}{2}\cdot\frac{1-\frac{d}{2}\sin\theta_c}{1+\frac{d}{2}\sin\theta_c}$	$\frac{1}{2}\cdot\frac{1-\frac{d}{2}\sin\theta_c}{1+\frac{d}{2}\sin\theta_c}$	$\frac{1}{2}\cdot\frac{1-\tan\left(\frac{\theta_0}{2Q}\right)}{1+\tan\left(\frac{\theta_0}{2Q}\right)}$	$\frac{1}{2}\cdot\frac{1-\tan\left(\frac{\theta_0}{2Q}\right)}{1+\tan\left(\frac{\theta_0}{2Q}\right)}$
γ	$\left(\frac{1}{2}+\beta\right)\cos\theta_c$	$\left(\frac{1}{2}+\beta\right)\cos\theta_c$	$\left(\frac{1}{2}+\beta\right)\cos\theta_0$	$\left(\frac{1}{2}+\beta\right)\cos\theta_0$

Efficient Approximations for Band-Pass Filter Coefficients

The filter coefficients described in Figure 18 must be evaluated every time the frequency-control parameter is modified. As in the case of the VCO's high-pass tracking filter, approximations to the exact formulas can be found, resulting in coefficient calculations that require fewer DSP instructions, and thus fewer MIPS. Using a Taylor-series expansion method with $a = 0$ in Equation 3b, approximations for the parametric low-pass and high-pass coefficients from Figure 18 become:

$$\beta \approx \frac{1}{2}\left(1 - \sqrt{2}\theta_c + \theta_c^2\right) \qquad (15a)$$

$$\gamma \approx \left(\frac{1}{2} + \beta\right)\left(1 - \frac{1}{2}\theta_c^2\right). \qquad (15b)$$

These approximations for β and γ give good results for $\theta_c < \pi/2$.

In the case of band-pass and band-stop filters, from Figure 18 an immediate simplification is

Figure 19. DSP56002 code for parametric band-pass filter.

```
; BPF with a normalized center frequency, f0 = theta0
;   beta  = vcf_beta0 (user supplied, program constant, less than 0.5,
;           as beta approaches 0.5, the filter Q goes to infinity)
;   alpha = (0.5 - beta)/2
;   gamma = (0.5+beta)*Cos(f0)

                                ; (1) Initialize work registers:
      move a,y:xn               ; xn is input sample = x(n)
      move #1,m4                ; Set r4 address register modulo = 2
      move m4,m5                ; Set r5 address register modulo = 5
      move y:vcf_xptr,r4        ; Retrieve input state address register
      move y:vcf_yptr,r5        ; Retrieve output state address register
      move #vcf_cof,r0          ; Initialize coefficient address register
                                ; (2) Calculate alpha:
      move #vcf_beta0,x0        ; x0 = vcf_beta0 = beta
      move #0.5,a               ; Set a = 0.5
      sub x0,a                  ; Set a = 0.5 - beta
      asr a                     ; Set a = (0.5 - beta)/2 = alpha
      move a,x:(r0)+            ; BPF coef, c[0] = alpha
      neg a                     ; Set a = - alpha
      clr a a,x:(r0)+           ; BPF coef, c[1] = -alpha
                                ;   and set a = 0
      move a,x:(r0)+            ; BPF coef, c[2] = 0
                                ; (3) Approximate gamma:
      move #0.5,a               ; Set a = 0.5
      add x0,a                  ; Set a = 0.5 + beta
      move a,y1                 ; Set y1 = a = 0.5 + beta
      move x:theta0,y0          ; Copy filter center freq, theta0, to y0
      clr a                     ; Set a = 0
      move #$800000,a1          ; Set a = 1.0
      mpy y0,y0,b               ; b = f0^2 = theta0^2
      asr b                     ; b = b/2 = 0.5*f0^2
      move b,y0                 ; Copy b to y0 = 0.5*f0^2
      sub y0,a                  ; a = a - y0 = 1.0 - 0.5*f0^2
      move a,y0                 ; Copy a to y0 = 1 - 0.5*f0^2
      mpy y0,y1,a               ; a = (0.5 + beta)(1 - 0.5*f0^2) = gamma
      move a,x:(r0)+            ; BPF coef, c[3] = gamma
      move x0,x:(r0)+           ; BPF coef, c[4] = beta
                                ; (4) Filter input sample, xn:
      move #vcf_cof,r0          ; Initialize coefficient address register
      move y:xn,y1              ; Copy xn to y1
      IIR                       ; IIR filter macro
      move x1,a                 ; Copy filter output in x1 to a
      move r4, y:vcf_xptr       ; Save input state address register
      move r5,y:vcf_yptr        ; Save output state address register
      rts                       ; Return from VCF subroutine
```

Figure 19. continued

```
IIR macro                                ; (5) Implements filter difference equation:
ori #$08,MR                              ; Turn on DSP56002 "scale mode"
move    x:(r0)+,x0                                ; Set BPF coef, x0 = c[0]
mpy x0,y1,b  x:(r0)+,x0 y:(r4)+,y0       ; b = c[0] * x(n)
                                         ;    x0 = c[1], y0 = x(n-2)
mac x0,y0,b  x:(r0)+,x0 y:(r4)+,y0       ; b = b + c[1] * x(n-2)
                                         ;    x0 = c[2], y0 = x(n-1)
mac x0,y0,b  x:(r0)+,x0 y:(r5)+,y0       ; b = b + c[2] * x(n-1)
                                         ;    x0 = c[3], y0 = y(n-1)
mac x0,y0,b  x:(r0)+,x0 y:(r5),y0        ; b = b + c[3] * y(n-1)
                                         ;    x0 = c[4], y0 = y(n-2)
macr -x0,y0,b            y1,y:(r4)+      ; b = b - c[4] * y(n-2)
                                         ;    x(n-1) = x(n)

move b,y:(r5) b,x1                       ; y(n-1) = 2 * b, x1 = 2 * b =
                                         ;    y(n) (note factor of 2 due to
                                         ;    scale mode)
andi #$F7,MR                             ; turn off scale mode
endm                                     ; end of macro
```

found, without any loss of accuracy, by setting θ_0/Q equal to a constant bandwidth. This is a useful mode for the VCF, since it will pass (or stop) a constant number of harmonic partials, independent of the value of θ_0. In this case, β and α become constants, independent of the value of θ_0, that only need to be calculated once for a specific bandwidth. The only remaining coefficient is γ, which can be approximated by the formula given by Equation 15b. The example code segment in Figure 19 shows this method of implementing the band-pass VCF with a constant bandwidth.

An Example Patch

Figure 20 shows a typical module patch where a keyboard controller is used as the user input. Two important parameters associated with many types of user input are the pitch output and the trigger output. The pitch signal controls the VCO, while the trigger signal controls the ADSR in Figure 20. There are two types of triggers: a pulse trigger that sets the ADSR decimated time index k to zero and allows the ADSR to count through the A, D, and R time intervals; and a level trigger that is used to hold the ADSR envelope in the sustain region as long as the trigger level is on. The pulse trigger could be used to synthesize a piano-like sound, for example, whereas the level trigger could be used to create an organ-like sound.

The example of Figure 20 shows a standard patch where the pitch-control input of the VCO is also routed to the filter-control input of the VCF. This combination of the triangle wave and low-pass filter produces a spectrum with relatively few harmonic partials. Coupled with short ADSR attack and decay times, the resulting sound will approximate an electric piano.

The example of Figure 21 shows a nonstandard patch where the pitch control input of the VCO is modulated by an LFO module and the frequency-control input of the VCF is the output of an ADSR block. This particular patch is used to evaluate the DSP software under worst-case MIPS loading, since the control parameters of the VCO and VCF are

Figure 20. Example patch of subtractive synthesis module, "Electric Piano."

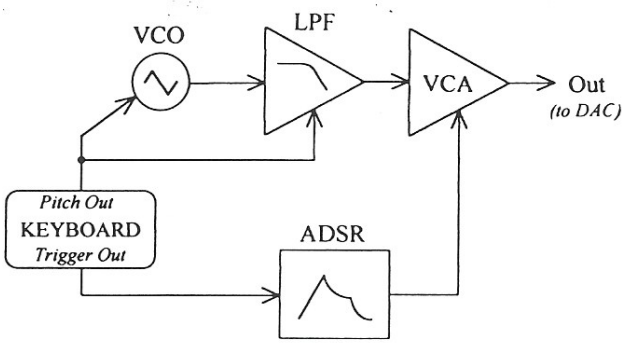
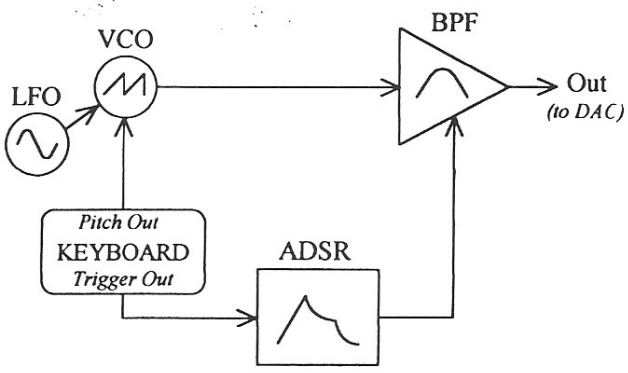


Figure 21. Another patch, "Filter Slide," for measuring worst-case MIPS usage.



continuously changing with time. This patch, "filter slide," also demonstrates the effect of moving (or sliding) the VCF control frequency back and forth over the harmonically rich spectrum of the VCO's sawtooth output. In addition, the LFO modulates the VCO control to produce a vibrato effect.

The example patches in Figures 20 and 21 both consume roughly 20 to 25 percent of the available MIPS on an 80-MHz Motorola DSP56002 digital signal processor, with a sample frequency of 44.1 kHz. The exact value of MIPS usage is also dependent on the details of the overall system integration and the extent of code optimization performed in the software development phase. Using these values as a worst-case MIPS benchmark, where the patch examples in Figures 20 and 21 contain four processing blocks, one DSP device could therefore implement 16 to 20 blocks. Under conditions of less loading (for example, if control parameters are updated less often), the number of processing blocks could easily be doubled. It is also obvious that some processing blocks require far fewer MIPS than others, so that the above estimates are only intended to serve as a guideline for typical patch configurations.

Like the analog synthesizers of the past, the number of possible patch combinations is immense. Unlike the analog counterpart, pitch stability is not a problem with the DSP subtractive synthesizer. There are many other advantages of the DSP synthesizer, such as the ability to perform patching from software. The second generation of subtractive synthesis machines incorporated microprocessor-controlled patching, which made the analog synthesizer much more practical from a performance standpoint. This feature is inherent in the DSP version. Also, as previously discussed, modules such as the VCA are trivial from a software-implementation point of view.

Two goals were attempted in this article. The first was to describe a systematic method of modeling analog synthesis with DSPs. This DSP algorithm design was deliberately equated with the design of the traditional analog subtractive synthesizer. To maintain a faithful reproduction of the subtractive synthesis system, a DSP implementation of each of the primary voltage-controlled modules was presented with mathematical analysis and sample DSP code. The second goal was to demonstrate an application using the parametric (tunable) IIR filter (Lane 1990). Both goals are connected by a common approach: first, find the exact mathematical relationship describing the system behavior, if at all possible; then, implement numerical approximations, as a final step, for practical implementation. The methods used in this article to model the analog synthesizer are based on this rationale, the same technique that originally defined the parametric IIR filter. This analysis and design philosophy may seem subtle in principal, but can lead to tremendous benefits in the long run.

References

Chamberlin, H. 1987. *Musical Applications of Microprocessors* (2nd ed.). Indianapolis, Indiana: Hayden Books.

Graham, R. D. 1978. *A Foundation for Electronic Music.* Osaka, Japan: Roland Corporation.

Lane, J. E. 1990. "Pitch Detection Using a Tunable IIR Filter." *Computer Music Journal* 14(3):46–59.

Lane, J. E., D. Hoory, and G. Hawkins. 1995. "Implementing Mathematical Functions on the Motorola DSP56000 Series of Digital Signal Processors." *DSP & Multimedia Technology* 4(6):28–38.

Mathews, J., and R. L. Walker. 1970. *Mathematical Methods of Physics* (2nd ed.). New York: W. A. Benjamin.

Motorola DSP Division. 1992. *DSP56000 Digital Signal Processor Family Manual.* Phoenix, Arizona: Motorola Literature Distribution Center.

PAIA Electronics, Inc. 1974. *4710 Balanced Modulator.* Oklahoma City, Oklahoma: PAIA Electronics, Inc.

Pierce, J. R. 1983. *The Science of Musical Sound.* New York: W. H. Freeman.

Press, W. H., et al. 1989. *Numerical Recipes, The Art of Scientific Computing.* New York: Cambridge University Press.

Wolfram, S. 1991. *Mathematica, A System for Doing Mathematics by Computer* (2nd ed.). Redwood City, California: Addison-Wesley.