MODELLING MUSICAL INSTRUMENTS IN THE DIGITAL DOMAIN

S. G. Smith

Integrated Systems Group
Department of Electrical Engineering
University of Edinburgh
King's Buildings, Mayfield Road, Edinburgh EH9 3JL

Summary

Digital hardware may be used to generate waveforms as well as to process already-existing waveforms. Several digital synthesis techniques are in use today, most of whose advantages lie in their computational efficiency rather than their utility. Only the methods of synthesis known as additive and subtractive, however, are accompanied by a suitable analysis technique, allowing the accurate extraction of parameters from real waveforms for subsequent use in synthesis. With the advent of VLSI, these techniques are becoming economically attractive for analysis and synthesis of sound. An approach to musical signal processing, based on additive synthesis, is presented.

1. Introduction

The use of digital signal processing in music generation is a relatively recent development. Three broad classes of digital synthesis have been identified in the literature [1,2]: additive, subtractive, and "modulation" or "contrived" methods. Of the third class, frequency modulation [3] is the most widely used, although waveshaping [4] and sundry others have generated some interest. However hardware efficient these methods are, they will not realise their full potential while lacking a complementary analysis technique. Some efforts have been made to rectify this deficiency for FM [5].

Subtractive synthesis is well known in speech processing [6], and is the cornerstone of analog music synthesis. It is useful in that it mimics the way in which many musical instruments operate, i. e. by filtering out unwanted components from a harmonically rich input waveform (driving function). Moreover, modern methods of signal analysis such as maximum entropy [7] are ideal companion analysis techniques. However subtractive synthesis suffers from the problem of generating a driving function that is both wideband and bandlimited (i. e. unaliased). Methods of overcoming this problem have been suggested recently [8].

It is well known that any periodic waveform may be represented to any degree of accuracy by a weighted sum of sinusoids (Fourier series). This is the well known discrete Fourier transform (DFT) [9]. The reconstruction of a waveform which has been decomposed by this method is known as additive synthesis [10]. Thus the waveform generation problem may be reduced to that of generating a series of stable and spectrally pure sinusoids, which may then be weighted and accumulated to produce the desired output. A general expression for additive synthesis is

$$y(n) = \sum_{k=1}^{K} A_k(n)\sin((k\omega + 2\pi F_k(n))nT) \qquad (1)$$

where y(n) is the output at time nT (n being time index and T sample period), ω is the fundamental radian frequency, and k is the harmonic number ($1 \leq k \leq K$). A_k and F_k allow dynamic control of amplitude and frequency. Only this approach seems to offer the potential of complete control over sound output, albeit at higher hardware cost.

2. Some Useful Structures for Additive Synthesis

Various methods can be used to generate sinusoids digitally, most of which are either table lookup methods (with possible subsequent interpolation [11],) or direct generation via a recursion. Although in some applications the former technique can be advantageous, we are primarily concerned with implementations of the latter.

2.1. The First-Order Section

Any digital filter, with complex poles which lie on the unit circle of the z-plane, will oscillate in a stable manner in response to an impulse. A structure of particular interest (Fig. 1) is the first-order recursive filter with complex pole (a complex multiplier with data feedback and unity coefficient modulus) [9], as there exists a simple correspondence between the coefficients and the output frequency.

$$a = \cos\phi \qquad \text{and} \qquad b = \sin\phi \qquad (2)$$

where ϕ is the desired angular increment per sample period, and a and b are respectively the real and complex part of the complex coefficient. The z-transform of this structure is

$$F_1(z) = \frac{1}{1 - Wz^{-1}} \qquad \text{where} \quad W = a + jb \qquad (3)$$

and its matrix of rotation is $\begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$

This structure will rotate any vector presented at its

CH1945-5/84/0000-0210 $1.00

input through ϕ radians per sample period, producing exact phase and quadrature outputs. Its main disadvantage is that its stability depends on the complex pole lying on the unit circle. The expression for the pole is bivariate, and accordingly is vulnerable to quantisation effects. True rotation requires that the bivariate function $a^2 + b^2 = 1$. Inevitably quantisation error in coefficients a and b will force the pole off the circle, causing loss of stability. Nonetheless this structure has properties such as no internal word growth, minimal storage requirements and ease of pitch change, making it a prime candidate for an oscillator part.

2.2. The Second-order Section (Goertzel Algorithm)

Goertzel [12] has effectively normalised the z-transform of the previous structure to produce a second-order filter, whose poles are real:

$$F_2(z) = \frac{1 - W^*z^{-1}}{(1 - Wz^{-1})(1 - W^*z^{-1})} = \frac{1 - W^*z^{-1}}{1 - 2\,\mathrm{Re}\{W\}z^{-1} + z^{-2}} \qquad (4)$$

This allows a reduction in hardware over the first-order filter. Fig. 2 shows the Goertzel flow-graph (after [9]) and a possible hardware implementation.

If we are willing to make do with only the real part of the complex exponential function, with zero starting phase, we may optimise the Goertzel structure (Fig. 2) to produce a stable cosine waveform using only one real multiplier. If quadrature outputs are desired, a second multiplier is required. The Goertzel structure preserves the simple relationship between coefficient and angular shift of the first-order structure, while stability is guaranteed due to the univariate nature of the pole. However as oscillator frequency approaches dc (or nyquist), internal word-growth becomes a problem. Furthermore, variation of angular shift (pitch change) is not as easily accomplished in this structure as in the first-order section. The Goertzel algorithm, then, is best suited to computations of the type

$$y(n) = \sum_{k=1}^{K} A_k(n)\cos(k\omega nT) \qquad (5)$$

i. e. no pitch change (c. f. eqn. 1).

2.3. The CORDIC Kernel

Some time ago, Volder [13] introduced the CORDIC (for COordinate Rotation DIgital Computer) algorithm, which was capable of both rotation and angle extraction using an iterative procedure. The heart of the algorithm was a recursion which was based on the first-order filter, using the approximations

$$\cos\phi \sim 1 \qquad \text{and} \qquad \sin\phi \sim \phi \qquad (6)$$

By restricting ϕ to negative powers of 2, Volder avoided true multiplication. Hence the iterative nature of the algorithm, as it "zeroed in" on the desired angular shift. Here the matrix of rotation is

$$\begin{bmatrix} 1 & -\phi \\ \phi & 1 \end{bmatrix}$$

Repeated application of this matrix leads to instability regardless of quantisation errors, and despite the univariate nature of the pole, as the coefficient modulus is bound to exceed unity. It is useful only at very low angles, where the approximation is valid. The scalar version of the CORDIC kernel equations (omitting the variable signing factor a_i [13]) is

$$y_{I+1} = y_I - \phi x_I \quad \text{and} \quad x_{I+1} = x_I + \phi y_I \qquad (7)$$

2.4. The Unterminated LDI

A simple modification (Fig. 3) of this structure (from a parallel to a serial computation) leads to the revised scalar equations

$$y_{I+1} = y_I - Cx_I \quad \text{and} \quad x_{I+1} = x_I + Cy_{I+1} \qquad (8)$$

and matrix of rotation $\begin{bmatrix} 1 & -C \\ C & 1 - C^2 \end{bmatrix}$

Note the change of variable - this is because we have lost the simple relationship between coefficient C and desired angular shift ϕ . The relationship between ϕ and C is now

$$\cos\phi = 1 - \tfrac{1}{2}C^2 \qquad (9)$$

which reveals that maximum angular shift attainable is $\pi/6$, as stability is lost if $C > 1$.

This structure is the unterminated version of the lossless discrete integrator (LDI) [14], whose poles are fixed on the unit circle, guaranteeing amplitude stability within the limits prescribed above. The outputs of this structure are no longer in quadrature, and to ensure unity output amplitude we must load the initial conditions

$$y_0 = 1 \qquad \text{and} \qquad x_0 = \tfrac{1}{2}C \qquad (10)$$

This structure also suffers from difficulty of dynamic alteration of angular shift, which is useful in additive synthesis. The step

$$x_{I+1} = x_I + \tfrac{1}{2}(C_{I+1} - C_I)y_I \qquad (11)$$

allows pitch change, although the approximations on which it is based are more hazardous than when vector phase and "previous frequency" are zero (our starting condition). The same step, with subtraction instead of addition, at the end of the recursion will restore the quadrature nature of the outputs.

To conclude this section, we see that there are many simple structures which will generate the complex exponential function. The univariate structures guarantee amplitude stability, but the bivariate structure has a simpler relationship between C and ϕ, and supports straightforward pitch-changing. In all cases accuracy of angular shift ϕ depends on coefficient quantisation.

3. Timbral Synthesis

The quality of a musical instrument which sets it apart from poor imitations is that of timbre. Timbre, although perceived by the listener in a largely subjective manner, can be linked with the amplitude evolution of spectral components in the sound. Thus, having created a harmonic series using individual

sinewave generators, it remains to recreate timbre by passing the individual sinusoids through a series of envelope functions.

3.1. Encoding the Envelope Function

An envelope can be approximated, in a piecewise linear manner, by a surface function in frequency and time. Three distinct envelopes are thought to make up the timbre of a musical instrument. These are (with example descriptions relevant to a stringed instrument model):

1) Fixed, time-independent (body of instrument)
2) Dependent on time and fundamental pitch (string of instrument)
3) Dependent on time and loudness (how hard string was plucked).

Furthermore, inspection of spectral evolutions of real instruments, as displayed in [15], reveals a distinct independence of odd and even harmonics, leading to the envelope generation structure shown in Fig. 4.

4. An Architecture for Additive Synthesis

We wish to realise the function of eqn. 1, while minimising the number of precomputed constants. The task splits into two main tasks - sinewave generation and envelope generation.

4.1. Envelope Generation

The envelopes, as stated earlier, are surface functions in time and frequency, and are approximated by linear segments. We need therefore only store data representing the breakpoints of the surface, which are triplets of numbers representing time, frequency and amplitude. The more breakpoints, the better the approximation to the desired surface. As we intend to generate harmonic series in increasing order of harmonic number k and (obviously) time-index n, we propose the storage, in correct order in a queue (first-in-first-out memory) Q1, of multiple instances of a pair of numbers defining the evolution of the envelope in the frequency direction. These are (using # to represent a number)

Q1. #1 - Amplitude increment in the frequency direction
#2 - number of frequency samples (k) for which the quantity #1 is valid.

The evolution in the time direction is encoded by storing a triplet of numbers in a secondary queue Q2 - these are

Q2. #3 - Increment of #1 in the time direction
#4 - Increment of #2 in the time direction
#5 - number of time samples (n) for which #3 and #4 are valid.

The surface function can be generated concurrently with the sinusoids as follows. Starting at n=1, k=1, as k increases we add #1 to the envelope, and decrement a copy of #2 until #4 = 0. We then read updated #3 and #4 from the queue, and continue. When k=K, increment n and repeat.

In a similar fashion, we decrement a copy of #5 with every increase in n, reading in new values of #3, #4 and #5 when #5 = 0. Q1 differs from Q2 in that it must contain 2 adders, to allow updating of #1 and #2 using #3 and #4 as they pass (once every k iterations). Fig. 5 shows a possible hardware configuration. Envelope 1 requires only Q1, due to its time-independence.

The length of these queues could be dynamically variable, to save on hardware. A certain value of #5 could mark "end of note". Provided enough amplitude resolution is used, this method should allow fairly efficient encoding of envelopes.

4.2. Sinewave Generation

The structure of Fig. 1 is best suited to sinewave generation, despite its inherent amplitude instability. We envisage the configuration of Fig. 6 for generation of a harmonic series.

On keypress (or whatever input medium is used) the oscillator initialises itself. The constants $\sin \phi$ and $\cos \phi$ (constituting W_1) are read from memory, and presented at the coefficients of the complex multiplier via multiplexing switch MUX1. Constants 1 and 0 (i. e. W_0) connect to the data port via MUX2. The data outputs pass through MUX3, until the coefficient queue is full (k = K).

The switches are then thrown (MUX1 - 3) and the circuit settles into steady state oscillation. Note the arithmetic capability of the coefficient queue - we feel that some "tweaking" of coefficients is required to overcome amplitude and frequency errors.

5. Analysis

We made the point earlier that a companion analysis technique is essential for serious music synthesis. The phase vocoder [16] is an example of an applicable parameter extraction technique - others can be found in [2]. Provided the synthesist has access to a good general-purpose computer with real time data-capture ability, he may analyse and parameter-extract at his leisure. Deconvolution techniques may be employed for separation of envelopes. We do not propose real-time analysis - only real-time performance (synthesis).

Software on the host computer should include exact models of the oscillator and envelope hardware. This allows the tweaking of coefficients mentioned earlier to be included at analysis time, and queued for subsequent performance. Other software might include an "envelope breakpoint editor" for creation of new sounds.

6. Implementation in Custom Silicon

We have based this paper on the premise that the availability of custom silicon is rapidly increasing [17]. In today's technology (e. g. 2 μm CMOS clocked at 32MHz with 32-bit integer arithmetic), using bit-serial hardware primitives for ease of construction and communication [17], we envisage two basic chips for synthesis:

1. An oscillator chip, containing eight physical first-order sections with multiplexing state

memory allowing each to realise 32 virtual oscillators band limited to 15kHz

2. An envelope chip, containing the set of six envelope sections required, duplicated eight times (queues would be off-board with serial interfaces).

Each chip pair would allow production of 256 independently controllable sinusoids in real time.

7. Conclusions

Some algorithms have been described for generation of sinusoids and envelope functions for real-time additive synthesis. They rely on computationally intensive but simple procedures, and are amenable to VLSI implementation. Some of the vector rotation hardware can also be used for "on-the-fly" coefficient generation in the FFT.

References

1. H. G. Alles, "Music Synthesis Using Real Time Digital Techniques," Proc. IEEE, Vol. **68** , (4) pp. 436 - 449 (April 1980).
2. J. A. Moorer, "Signal Processing Aspects of Computer Music: A Survey," Proc. IEEE, Vol. **65**, (8) pp. 1108 - 1137 (August 1977).
3. J. M. Chowning, "The Synthesis of Complex Audio Spectra by means of Frequency Modulation," J. Audio Eng. Soc., Vol. **21**, (7) pp. 526 - 534 (September 1973).
4. M. Le Brun, "Digital Waveshaping Synthesis," J. Audio Eng. Soc., Vol. **27**, (4) pp. 250 - 266 (April 1979).
5. J. H. Justice, "Analytic Signal Processing in Music Computation," IEEE Trans. ASSP, Vol. **ASSP-27**, (6) pp. 670 - 684 (December 1979).
6. J. Makhoul, "Linear Prediction: An Overview," Proc. IEEE, Vol. **63**, (4) pp. 561 - 580 (April 1975).
7. J. P. Burg, "Maximum Entropy Spectral Analysis," Proc. 37th Annual International Meeting of the Soc. Explor. Geophys., (Oklahoma City, October 1967).
8. T. W. Goeddel and S. C. Bass, "Complex Voice Generation for Digital Instruments," Proc ISCS'81, pp. 742 - 745 (Chicago, March 1981).
9. A. V. Oppenheim and R. W. Schafer, "Digital Signal Processing", Prentice-Hall (1975).
10. J. A. Moorer, "The Synthesis of Complex Audio Spectra by Means of Discrete Summation Formulae," J. Audio Eng. Soc, Vol. **24**, (9) pp. 717 - 727 (November 1976).
11. J. Tierney, C. M. Rader, and B. Gold, "A Digital Frequency Synthesizer," IEEE Trans. Audio Electroacoust., Vol. **AU-19**, pp. 48 - 56 (March 1971).
12. G. Goertzel, "An Algorithm for the Evaluation of Finite Trigonometric Series," Am. Math. Monthly, Vol. **65**, pp. 34 - 35 (1958).
13. J. E. Volder, "The CORDIC Trigonometric Computing Technique," IRE Trans. Electron. Comput., Vol. **EC-8**, pp. 330 - 334 (August 1959).
14. L. E. Turner, "Elimination of Constant-Input Limit Cycles in Recursive Digital Filters using a Generalised Minimum Norm," Proc. IEE Pt. G, Vol. **130**, (3) pp. 69 - 77 (June 1983).
15. J. A. Moorer and J. M. Grey, "Lexicon of Analysed Tones," Computer Music J., Vol. **1**, (1977).
16. M. R. Portnoff, "Implementation of the Digital Phase Vocoder Using the Fast Fourier Transform," IEEE Trans. ASSP, Vol. ASSP-24, (3) pp. 243 - 248 (June 1976).
17. P. B. Denyer and D. Renshaw, "An Approach to VLSI Signal Processing", book in preparation (November 1983).

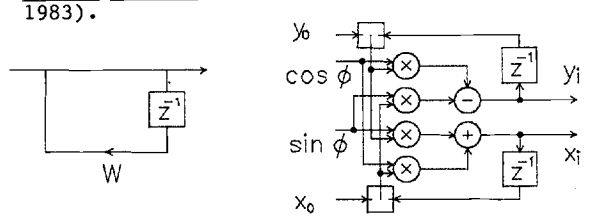

Fig. 1 First-order flowgraph and hardware

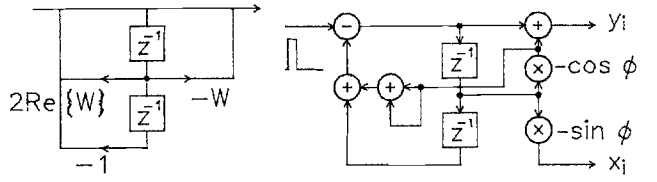

Fig. 2 Goertzel flowgraph and hardware

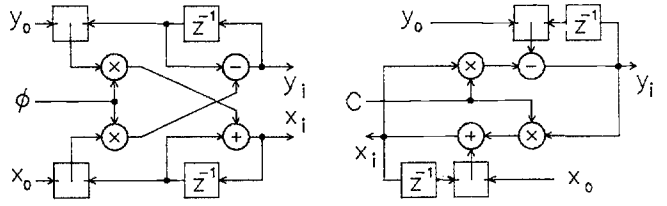

Fig. 3 CORDIC kernel and unterminated LDI

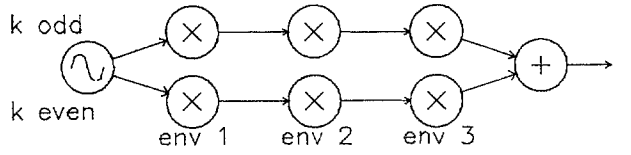

Fig. 4 Envelope Configuration

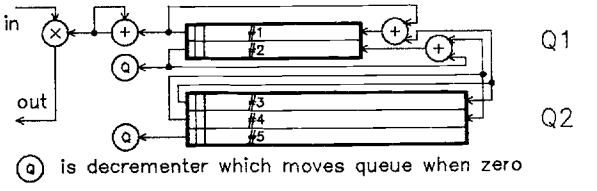

Fig. 5 Envelope parameter queues

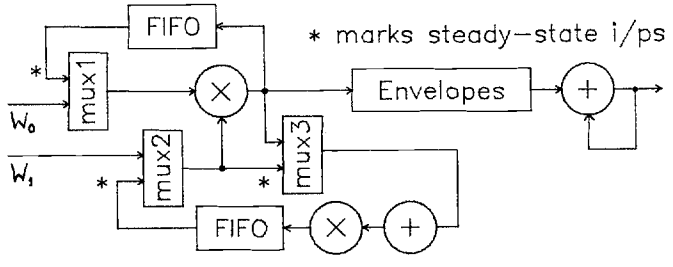

Fig. 6 Hardware for harmonic series generation