A FIXED-POINT RECURSIVE DIGITAL OSCILLATOR FOR ADDITIVE SYNTHESIS OF AUDIO

Todd Hodes, John Hauser, John Wawrzynek

Computer Science Division
University of California, Berkeley
{hodes,jhauser,johnw}@cs.berkeley.edu

Adrian Freed, David Wessel

Center for New Music
and Audio Technologies
University of California, Berkeley
{adrian,wessel}@cnmat.berkeley.edu

ABSTRACT

This paper summarizes our work adapting a recursive digital resonator for use on sixteen-bit fixed-point hardware. Our modified oscillator is a two-pole filter that maintains frequency precision at a cost of two additional operations per filter sample. The new filter's error properties are expressly matched to use in the range of frequencies relevant to additive synthesis of digital audio and sinusoidal modelling of speech in order to minimize the additional computational overhead. We present the algorithm, an error analysis, a performance analysis, and measurements of an implementation on a fixed-point vector microprocessor system.

1. INTRODUCTION

There are many benefits to the use of additive synthesis for sound production in computer music applications, and analogously, in sinusoidal modelling of speech. These include expressive musical control over fine timbral distinctions, perceptually relevant parameterizations, sample rate independence of timbre description, availability of many analysis techniques, high control bandwidth, and multiple dimensions for resource allocation/optimization [1]. Its use also leverages existing tools and structural manipulation techniques for the domain [2, 3]. The challenge of the additive technique lies in its appetite for large numbers of separately controllable sinusoidal partials.

Use of the sinusoidal additive technique requires two key decisions: which hardware architecture to use (general purpose, custom-designed ASIC, DSPs, etc.), and which sinusoid generation algorithm to use on that architecture. For the former, one promising avenue is the use of digital signal processors or vector (or "multimedia") processors as a natural fit to the data type and associated computational demands. Unfortunately, such architectures do not always support full-range (i.e., floating-point) arithmetic; fixed-pint may be all that is provided. Assuming the decision is made to target such an architecture, the second question becomes that of determining which algorithm to use given this constraint.

A number of sinusoidal partial production techniques have been reported in the literature. They all tend to fall into one of three classes: those that implement recursive filters, those using table-lookup, or those that work in the transform-domain using techniques such as the inverse fast fourier transform [4]. Recursive oscillators may be preferred over other approaches for one or more reasons: the inherent fine-grain exposure of data parallelism, the far more limited demand on the memory system compared to table-lookups, the lower induced latency than with a transform-domain approach, the latency flexibility, and/or the attainable phase accuracy.

This paper summarizes our developments for recursive oscillator generation on reduced-precision arithmetic hardware. We adapt a recursive digital resonator for use on fixed-point hardware, modifying the oscillating filter to maintain greatly enhanced frequency precision at the cost of two additional operations per filter sample. The new filter's error properties are expressly tailored for use in the range of frequencies relevant to digital audio, as opposed to general-purpose applications, in order to minimize computational overhead required to obtain the additional accuracy. We present the algorithm, an error analysis, a performance analysis, and measurements of an implementation of the algorithm on a fixed-point vector microprocessor system.

2. METHOD DESCRIPTION

The particular recursive form we use is a constant-gain digital resonator [5, 6]. Requiring only a single multiply, it is computationally less expensive than the waveguide oscillator [7] or the modified coupled form [8].

The challenge of using moderate-precision arithmetic units and numeric representations for recursive oscillators lies in

- addressing the error accumulation inherent in recursive methods (i.e., quantization-induced noise effects), and
- providing sufficient frequency coefficient resolution.

Our approach to these challenges is two-fold: keep individual oscillators short-lived via an "overlap-add" approach to exploit short-term fidelity [9], and modify the recursion calculation to increase frequency accuracy. For this paper we focus on the latter technique.

The general form of the digital resonator, with no damping or initialization impulse function, is:

$$x_n = 2\cos\left(\frac{2\pi f}{f_s}\right) x_{n-1} - x_{n-2}$$

where f_s is the sampling frequency and $f \in (0, f_s/2)$ is the (constant) desired frequency of oscillation.

To implement this equation using only sixteen-bit fixed-point multiplies, we needed to (1) manage the fixed-point units with enough precision to maintain accuracy across the entire audible frequency range, while (2) taking special care to provide sufficient frequency coefficient resolution to account for human ability to distinguish subtle differences in low frequencies. Accuracy must be maintained across a broader range and with more precision for low-frequency partials than a simple sixteen-bit fixed-point representation supplies. Additionally, because the frequency coefficient multiplication is in the critical path, we want to minimize the computational overhead of the changes.

To quantify the issue, we use the experimentally determined just-noticeable difference (JND) psychoacoustic curve for frequency differentiation [10]. This curve can be used to reliably estimate the precision required for an encoding of the frequency coefficient so that inaccuracies remain below the approximate threshold of human differentiation. Using the JND curve, we can specify a minimum perceptible musical interval and then calculate the resolution necessary to maintain relative frequency accuracy. Doing so indicates that the low-frequency components require more precision than higher ones – which is intuitive, since we are calculating *relative* accuracy. Thus, to minimize perceived error, we remap the frequency coefficient representations in two ways: we employ an exponent internally to emulate floating-point range extension, and invert the bit representation to bias accuracy toward low frequencies rather than high frequencies. These changes require two new operations per filter sample: an add with constant shift and a variable shift. (The constant shift and mantissa multiply can be *fused* in some fixed-point arithmetic pipelines as a single operation, thereby requiring only one cycle.)

To understand the modifications to the filter, recall the original recurrence relation for our sine wave generator (with $\omega = \frac{2\pi f}{f_s}$). At low frequency, the coefficient $2\cos(w)$ is very close to two, and so in a floating-point format, lower frequencies synthesized using the formula will have less accuracy than higher-frequency oscillators due to the need to explicitly represent the leading ones (the high-order bits) of $2\cos(w)$. Numbers closer to zero benefit from the implicit encoding of leading zeros via a higher exponent. In other words, larger values require bits with larger "significance" (absolute value), forcing the least significant bits in the same word to also have higher significance, thus forcing higher worst-case quantization error. We can more effectively use the bits of the mantissa by reversing this relationship, recasting the equation as:

$$\begin{aligned} x_n &= 2\cos(w)x_{n-1} - x_{n-2} \\ x_n &= 2(1-\epsilon/2)x_{n-1} - x_{n-2} \\ x_n &= 2x_{n-1} - \epsilon x_{n-1} - x_{n-2} \end{aligned} \quad (1)$$

i.e., where $\cos w = (1 - \epsilon/2)$.

To represent ϵ, an unsigned sixteen-bit mantissa m is combined with an unsigned exponent e, biased so that the actual represented value is $\epsilon = 2^{2-e}m$. Thus, the exponent is also the right shift amount necessary to correct a 16b×16b → 32b multiply with ϵ as an operand. The two in the exponent allows ϵ to range from 0 to 4 when m is interpreted as a fractional amount and f ranges between zero and the Nyquist frequency.

What is achieved with this remapping of number representation is the ability to represent lower frequencies with more significant bits by mapping higher frequencies to representations with less significant digits. In particular, as $2\cos(2\pi f/f_s)$ varies from -2 to 2, we define ϵ to vary from 4 to 0. Smaller frequency values produce smaller values of ϵ, helping to satisfy our asymmetric accuracy requirements.

3. ERROR ANALYSIS

3.1. Relative Error

Relative frequency discrimination is based on the ratio of adjacent frequencies. To determine worst-case relative error, We wish to determine the maximum ratio between two adjacent ϵ values. Call these frequency coefficients ϵ_1 and ϵ_2, and their corresponding frequencies f_1 and f_2. From the definition of ω and the equation $\cos w = (1 - \epsilon/2)$,

$$\frac{f_1}{f_2} = \frac{\cos^{-1}(1-\epsilon_1/2)}{\cos^{-1}(1-\epsilon_2/2)} \quad (2)$$

The maximum value for ϵ that we can represent, which is 11.11111111111111_2, sets the highest-valued least significant bit of any number in the range to 2^{-14}. We have defined the exponent e to be zero for this location of the binary point, and by design, numbers with this exponent are maintained with the least accuracy. This ϵ value coincides with the maximum representable frequency. Because humans perceive pitch as the log of frequency, this achieves a good match of the number representation to the asymmetric accuracy requirements.

Taking any two adjacent numbers in the range, we can compute f_1/f_2 with Eq. 2. Evaluating this ratio for all possible adjacent pairs of epsilon values allows us to determine that it is maximized for $\epsilon_1 = 4 - 2^{-14}$ and $\epsilon_2 = 4 - 2^{-13}$, where $f_1/f_2 \approx 1.0010337$. This ratio is lower than the minimum frequency ratio humans are able to differentiate, a pitch difference of approximately four to five cents (about 1/25–1/20 of a semitone) [10]. The maximum error of our algorithm is actually less than two cents: $\sqrt[600]{2} \approx 1.001156$.

3.2. Absolute error

Two tones that are meant to have an exact ratio in their frequencies may instead generate beat frequencies due to frequency quantization. This effect, caused by *absolute error*, should be minimized.

Worst-case absolute error due to epsilon quantization is shown in Figure 1, which contains both a side-by-side comparison below 2000 Hz and a detail for the modified form. As expected, the recast filter maintains more precise absolute frequency than the original form.

Fundamentally, more than 16 bits of fractional coefficient are necessary to obtain 1 Hz absolute precision across the audible spectrum [11]. Our method maintains reasonable error bounds in sixteen bits of mantissa by scaling these bits with the exponent.

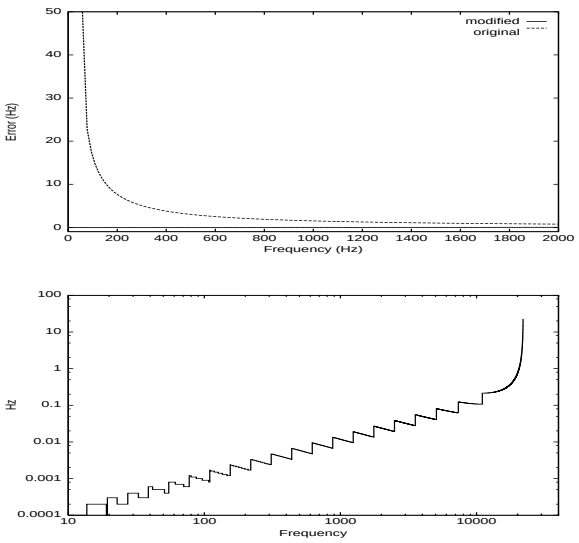

Figure 1: Worst-case absolute error due to frequency coefficient quantization. Top: comparison of original and modified. Bottom: modified form detail.

3.3. Rounding error

At each iteration of the recursive form, a small error is introduced due to rounding of the multiply result. Due to the recursion, this error isn't corrected until reinitialization of the state variables. Possible effects of this include degradation of the signal-to-noise ratio, degradation of the long-term phase accuracy, and a lack of amplitude stability — all of which can cause audible artifacts.

As has been illustrated elsewhere [8], this can be corrected via additional computation. To avoid this, we exploit our ability to reinitialize the computation at overlap-add frame boundaries, thereby allowing use of a non-self-correcting (but higher-performance) oscillator form.

4. FAST INITIALIZATION

The resonator can be initialized to a desired frequency and phase at sample x_0 by properly choosing the two state variables x_{-2} and x_{-1} using function evaluations in place of an initialization forcing function. The lookup values for a sinusoid with phase p and frequency f are:

$$x_{-1} = \sin\left(p - \frac{2\pi f}{f_s}\right) \quad \text{and} \quad x_{-2} = \sin\left(p - \frac{4\pi f}{f_s}\right)$$

These initializations must be accurate down to the low-order bits in a 32-bit fixed point representation, with the binary point set between the third and fourth bit positions in order to support a phase in the range $[0, 2\pi]$. Additionally, we need to compute the frequency coefficient $2 - 2\cos(\omega)$ to 32-bit accuracy.

We implement these initial evaluations by rewriting:

$$\begin{aligned} x_{-1} &= \sin(p - \omega) \\ &= \sin(p)\cos(\omega) - \cos(p)\sin(\omega) \end{aligned}$$

$$\begin{aligned} x_{-2} &= \sin(p - 2\omega) \\ &= \sin(p)\cos(2\omega) - \cos(p)\sin(2\omega) \\ &= \ldots \\ &= 2\cos(\omega)\sin(p - \omega) - \sin(p) \\ &= 2\cos(\omega)x_{-1} - \sin(p) \end{aligned}$$

This recasting allows us to require only the computation of $\sin(p)$, $\cos(p)$, $\sin(\omega)$, and $\cos(\omega)$.

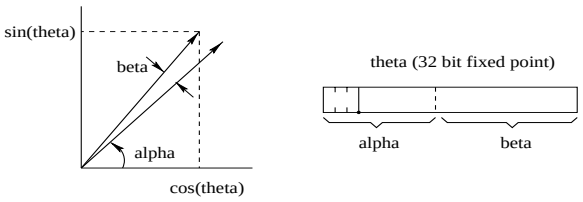

Figure 2: Calculating both $\sin(\theta)$ and $\cos(\theta)$ with a hybrid technique: table-lookup for α and Taylor expansion for β combine to give an accurate 32-bit result efficiently.

It may seem that this has actually increased the amount of work we need to perform because there are now four trigonometric evaluations rather than three (two initialization sines plus the cosine in the recursive form). However, this approach turns out to be more efficient by allowing for the judicious sharing of intermediate values in a tandem sine and cosine generation procedure. The tandem subroutine returns both $\sin(\theta)$ and $\cos(\theta)$ for $\theta \in [0, 2\pi]$ to full 32-bit fixed-point precision using a hybrid technique combining table-lookup and Taylor expansion. This keeps both the table size manageable (2048 entries of 32 bits) and the number of terms in the Taylor expansions small (two). It is implemented by separating θ into α and β as shown in Figure 2; α is the high-order 11 bits of θ, and β the remaining low-order bits. α is used in an exact (to one LSB) 11-bit $\rightarrow$ 32-bit table-lookups, while (guaranteed small) β is used in Taylor expansions:

$$\cos(\beta) \approx 1 - \frac{\beta}{2} \quad \text{and} \quad \sin(\beta) \approx \beta\left(1 - \frac{\beta^2}{6}\right)$$

The accuracy of expanding each to only two terms is guaranteed by limiting the size of β to only the low-order 21 bits of θ: the sum of the remaining terms in each expansion sequence, for all β, is less than the LSB. Finally, α and β are combined using the relationships:

$$\begin{aligned} \sin(\alpha + \beta) &= \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta) \\ \cos(\alpha + \beta) &= \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta) \end{aligned}$$

5. RELATED WORK

A similar approach has been used on the TI TMS32010 DSP [12], where they suggest using 32-bit multiplies (using three 16-bit stages) and modifying the program to hard-code shift values. The latter technique in inappropriate to vector architectures and the former unnecessary to achieve the accuracy needed for audio. Similarly, extensive work on variable representations for direct-form oscillators and their associated error properties has been reported in the

literature [13, 14], but none specifically matched to the important and practical problem domain of additive synthesis of audio or the equivalent usage in sinusoidal modelling of speech.

6. IMPLEMENTATION AND PERFORMANCE

We implemented our approach on the SPERT neural network and signal processing accelerator board [15]. The SPERT houses a T0 chip [16], which tightly couples a general-purpose scalar MIPS core to a high-performance vector coprocessor (comprising two vector arithmetic units and a vector load/store unit). T0 is representative of digital signal processing architectures in its use of fixed-point arithmetic.

Computing summations of oscillators for additive synthesis with an overlap-add approach using Eq. 1 and our pseudo-floating-point format, we need four multiplies, two variable shifts, two fused (constant) shifts and adds, and two regular adds:

$$x_n = 2x_{n-1} - \epsilon x_{n-1} + x_{n-2}$$
$$A_n = A_{n-1} + \Delta A$$
$$\text{out}_i = \text{out}_i + A_n \times x_n$$

We use 32-bit fixed-point intermediates for the amplitude, the amplitude delta, and the state variables. To satisfy alignment requirements, constant shifts are needed to convert them from 32-bit to 16-bit prior to a multiply. This requires an additional three arithmetic operations beyond those above (only three because ϵ's mantissa is already 16-bit). One of these shifts can be obtained from a prior iteration via software pipelining. This leads to a total of $9\frac{1}{n}$ vector arithmetic operations per sinusoid when unrolled n times. Unrolling four times due to trade-offs in register file pressure on T0, we achieve the following best-case performance: $\frac{2 \text{ sinusoids}}{\text{partial}} \times \frac{9\frac{1}{4} \text{ ops}}{\text{sine}} \times \frac{4 \text{ cycles}}{2 \text{ vector ops}} \times \frac{\text{vector}}{32 \text{ elements}} \Rightarrow \frac{37}{32}(\sim 1.15)\frac{\text{cycles}}{\text{partial}}$

Thus, for our SPERT board performing 8 operations per cycle (peak) with a 40 MHz clock rate and at a 44.1 kHz sampling rate, a theoretical maximum of 768 partials can be achieved in real time excluding all overhead. The current implementation on the prototype accelerator, including overhead, supports up to 608 simultaneous real-time partials with frame lengths of 5.8 ms or greater, or about 1.5 cycles per partial per sample.

7. CONCLUSIONS

We have summarized our work adapting a recursive digital resonator for use on sixteen-bit fixed-point hardware with error properties expressly matched for use in the range of frequencies relevant to additive synthesis of digital audio or sinusoidal modeling of speech. The new technique allows for reliable computation of frequencies impossible to handle with the direct form, while keeping additional frequency precision costs down to only two additional operations per filter sample. We presented the algorithm, an error analysis, a performance analysis, and measurements of an implementation on a fixed-point vector microprocessor system. More detailed information is available elsewhere [17].

8. REFERENCES

[1] P. Cook, R. Bargar, X. Serra, and A. Freed, "Creating and Manipulating Sound to Enhance Computer Graphics," *SIGCOMM tutorial session*, 1996.

[2] A. Freed, "Bring Your Own Control Additive Synthesis," *International Computer Music Conference*, 1995.

[3] Center for New Music and Audio Technologies, "CNMAT Additive Synthesis Toolkit (CAST) project." http://cnmat.cnmat.berkeley.edu/CAST/, 1996.

[4] A. Freed, "Real-time Inverse Transform Additive Synthesis for Additive and Pitch Synchronous Noise and Sound Spatialization," *Proceedings of the Audio Engineering Society 104th Convention*, 1998.

[5] J. Smith and J. Angell, "A Constant-Gain Digital Resonator Tuned by a Single Coefficient," *Computer Music Journal*, no. 4, 1982.

[6] J. Wawrzynek and C. Mead, "A VLSI Architecture for Sound Synthesis," in *VLSI Signal Processing: A Bit-Serial Approach* (Denyer and Renshaw, eds.), Reading: Addison Wesley, 1985.

[7] J. Smith and P. Cook, "The Second-Order Digital Waveguide Oscillator," *Proc. Int. Conf. Computer Music*, 1992.

[8] J. Gorden and J. Smith, "A Sine Generation Algorithm for VLSI Applications," *Proc. Int. Conf. Computer Music*, 1985.

[9] M. Goodwin and A. Kogon, "Overlap-Add Synthesis of Nonstationary Sinusoids," *Proc. Int. Computer Music Conf.*, 1995.

[10] J. O. Pickles, *An Introduction to the Physiology of Hearing*. Academic Press: Orlando, Florida, 1982.

[11] J. Wawrzynek, *VLSI Concurrent Computation for Music Synthesis*. PhD thesis, California Institute of Technology, 1987.

[12] A. I. Abu-El-Haija, M. M. Al-Ibrahim, and M. A. Al-Jarrah, "Recursive Digital Sine Wave Oscillators Using the TMS32010 DSP," *IMTC WEPM 7-6*, 1994.

[13] B. Gold and C. M. Rader, *Digital Processing of Signals*. New York: McGraw-Hill, 1969.

[14] J. I. Acha, J. Payan-Somet, and A. Civi, "Design of Very-low-frequency Digital Oscillators," *IEEE Proceedings G (Electronic Circuits and Systems)*, vol. 131, pp. 93–102, 1984.

[15] J. Wawrzynek *et al.*, "SPERT-II: A Vector Microprocessor System," *IEEE Computer*, vol. 29, pp. 79–86, March 1996.

[16] K. Asanovic *et al.*, "T0: A Single-Chip Vector Microprocessor with Reconfigurable Pipelines," *Proceedings 22nd European Solid-State Circuits Conference*, September 1996.

[17] T. Hodes, "Recursive Oscillators on a Fixed-Point Vector Microprocessor for High Performance Phase-Accurate Real-Time Additive Synthesis," Master's thesis, Univeristity of California, Berkeley, 1997.