Wavetable Synthesis Strategies for Mobile Devices*

ROBERT C. MAHER, *AES Member*

Department of Electrical and Computer Engineering, Montana State University, Bozeman, MT 59717-3780, USA

Table look-up (or fixed-wavetable) music synthesis has recently received new interest in the context of mobile applications such as personalized ring tones and pager notification signals: the market for consumers purchasing polyphonic ring tones exceeded US $4 billion worldwide in 2004. Table look-up (or fixed-wavetable) synthesis methods have been widely used for many years in music synthesizers. The limited amount of storage and transmission bandwidth available in such mobile devices makes the use of compressed synthesizer data desirable. The important considerations for implementing a wavetable music synthesizer in a small portable device are described. In particular, an efficient compression/decompression method for the synthesizer data is used in order to reduce the size of the synthesizer data bank with good fidelity while still maintaining the loop continuity constraints necessary for looped wavetable synthesis.

0 INTRODUCTION

Recently there have been commercial projects that require the use of stored waveform music synthesis for personalized ring and pager tones in mobile telephony [1]. While the availability of custom ring tones might seem at first glance to be a trivial frill for mobile phone users, AES members may be surprised to learn that the market for downloaded ring tones exceeded US $3.5 billion in worldwide sales during 2003, with future sales predicted to reach $5.2 billion per year by 2008 [2]. Considering that in 2003 the worldwide market for all recorded music was approximately $32.2 billion, the ring tone market represented fully 10% of entertainment audio sales in 2003—an astonishing statistic indeed. Although some mobile telephony devices allow playback of short digital audio recordings with compressed data, devices in the commercial marketplace also have to provide support for looped waveform music synthesis.

Digital fixed-waveform table-look-up methods have been used widely in music synthesis applications [3]–[5]. In its most basic form, a single cycle of the desired waveform is digitized and then simply played back repeatedly (looped) to create a sustained digital signal to be sent through a digital-to-analog converter for audition. In order to create musically interesting synthesized signals it is necessary to have a set of stored waveform tables that correspond to sounds from a variety of musical instruments, or from different pitch and amplitude ranges of the same instrument. The size of the set of synthesizer look up tables can become quite large if a wide range of timbres at a high level of quality is desired, and the complexity of selecting, manipulating, and possibly cross-fading various tables in real time can be significant. Nonetheless, the table look-up technique has been employed successfully in many commercial products and computer music synthesis systems.

The synthesis requirements must entail careful consideration of the wavetable synthesizer implementation choices, particularly the size of the wavetable data set. The extremely limited amount of storage, transmission bandwidth, and available computation—not to mention the prevalent cost-reduction pressure of the competitive marketplace—has required a simple and efficient means to reduce the size of stored synthesis wavetables while still allowing low-complexity decompression. This communication describes one such commercial wavetable synthesis implementation.

In order to achieve a reduction in the storage required for the wavetable data, the number of bits used to represent each sample must be reduced. It is, of course, desirable to perform the bit-rate reduction losslessly so that the original wavetable can be recovered exactly from the compressed data [6]. However, to achieve a useful data compression factor and minimal decoding complexity it was necessary to use a quantized (lossy) representation that would still meet the loop continuity and Nyquist constraints, as will be described.

*Manuscript received 2004 January 19; revised 2004 December 2 and 2005 January 7. Based on “Compression and Decompression of Wavetable Synthesis Data,” presented at the 115th Convention of the Audio Engineering Society, New York, 2003 October, convention paper 5937.

At the outset one might reasonably ask whether perceptual audio coders such as the MPEG family could be used to compress wavetable synthesizer data [7]. The unfortunate answer is no, because the compact wavetable synthesizer framework requires continuity between waveform loops so that no audible discontinuity is present. Thus a coder must maintain wave-shape integrity, which is not a goal of the standard perceptual audio coders. Furthermore, the wavetable decoding and signal generation occur at the signal sample rate, so minimizing the cost of computation is an important design factor.

A nonuniformly quantized differential pulse-code-modulation (DPCM) procedure is used in this implementation. The DPCM technique is a well-known waveform coding procedure, of low complexity, and convenient to modify for wavetable compression [8]. The encoder calculates the difference between an estimate of the current wavetable sample and its actual value. The difference is then quantized with a nonuniform characteristic, such as A-law, μ-law, or logarithmic. This allows more resolution for small signal details and relatively less accuracy for large signal changes. The output is decompressed by decoding the sample differences and summing them to obtain the output waveform.

However, the use of lossy coding for looped wavetables poses several peculiar problems. First the reconstruction obtained from the quantized differential data may not allow a "perfect" loop (amplitude match at the end points), which may cause an audible click or dc buildup as the waveform is cycled. Second the introduction of quantization errors into the wavetable data occurs after the antialiasing (or resampling) filters that are used by the sound designer in the looping process, and the resulting error signal is unlikely to be properly band limited, resulting in unintended aliasing in the wavetable signal. Finally the use of common statistical procedures such as dither and noise feedback to ameliorate the audible effects of the quantizer is not appropriate for wavetable coding, since the added signals are "frozen" into the stored wavetable and looped, rather than being statistically independent as in ordinary digital audio processing. Thus the use of compressed wavetable synthesis data requires special considerations.

The remaining sections of this communication are organized as follows. First a summary of wavetable principles and differential encoding is given. Next the low-complexity encode/decode scheme for wavetable data is presented. Finally several variations of the proposed technique are discussed.

1 WAVETABLE SYNTHESIS

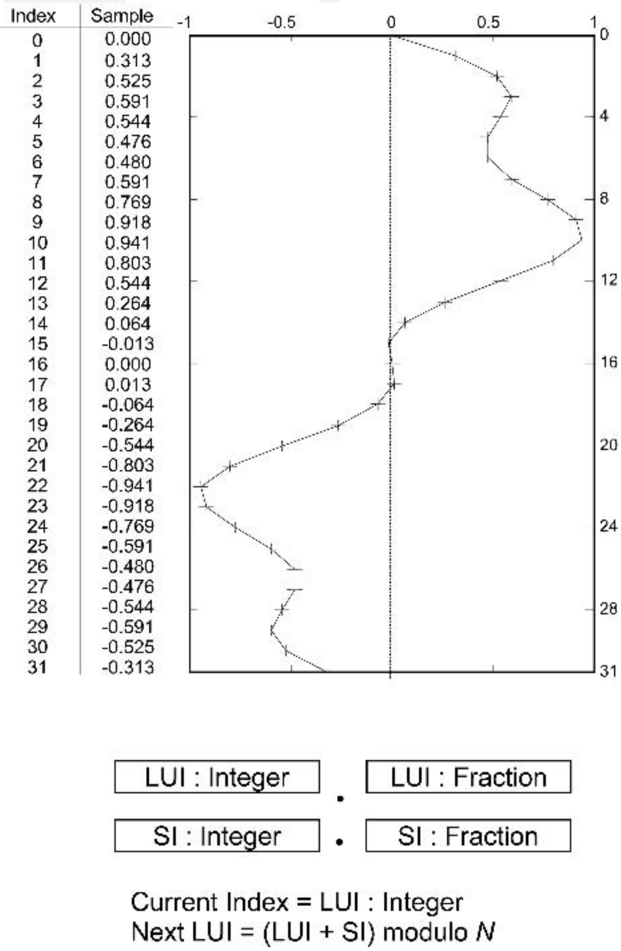
A fixed wavetable or stored-waveform synthesizer operates by repeatedly sending an array of waveform samples—the wavetable—through a digital-to-analog (D/A) converter. The basic schema of a wavetable synthesizer for $N = 32$ is shown in Fig. 1.

It is generally necessary to be able to synthesize a periodic waveform of arbitrary length, so the fixed wavetable is often filled with one or more cycles of a periodic waveform, which is then repeatedly cycled, or looped, as long as necessary. If the array of samples is of length N and the D/A converter is clocked at a fixed rate f_s samples per second, it requires N/f_s seconds to read every sample from the table. This corresponds to a waveform frequency of f_s/N Hz, assuming the table contains just one waveform cycle.

1.1 Resampling and Interpolation

For music synthesis it is usually necessary to produce waveforms with differing and perhaps time-varying fundamental frequencies, and certainly frequencies that are not integer divisors (f_s/N) of the sample frequency [9], [10]. This requires sample-rate conversion of the wavetable data, meaning that waveform samples in between the given samples of the wavetable will need to be interpolated. Higher order interpolation can provide a closer approximation to band-limited interpolation of the wavetable, at the expense of additional computation—more than the two adjacent wavetable samples are required for a higher order structure.

Sample-rate conversion condenses or expands the spectrum of the signal stored in the wavetable, and this can cause timbral changes (formant shifts), which are usually undesirable. Further, the resampling process may introduce spectral aliasing if the interpolation is not strictly band-limited or if the wavetable is critically sampled to begin with. Since it is



Index	Sample
0	0.000
1	0.313
2	0.525
3	0.591
4	0.544
5	0.476
6	0.480
7	0.591
8	0.769
9	0.918
10	0.941
11	0.803
12	0.544
13	0.264
14	0.064
15	-0.013
16	0.000
17	0.013
18	-0.064
19	-0.264
20	-0.544
21	-0.803
22	-0.941
23	-0.918
24	-0.769
25	-0.591
26	-0.480
27	-0.476
28	-0.544
29	-0.591
30	-0.525
31	-0.313

Fig. 1. Example wavetable of length $N = 32$. Look-up index (LUI) and sample increment (SI) are rational numbers used to set table repetition frequency.

advantageous to minimize the size (length) of the wavetable for practical reasons, there is a tradeoff between 1) using a longer wavetable with oversampled data to allow easier resampling, and 2) using a shorter wavetable with critically sampled data to conserve storage space.

1.2 Other Considerations

In addition to the cyclical wavetable as described, it is common in practice to provide additional features for enhancing the flexibility and sound quality of the wavetable procedure. One important feature is to allow an attack waveform to be spliced in front of the cyclical wavetable, as depicted in Fig. 2.

The attack portion is played once at the onset of the musical note, while the cyclical wavetable is used to sustain the length of the note as necessary. The use of a one-shot attack segment allows for more realistic and/or musically interesting timbres.

Another common feature is to provide an amplitude envelope function to create a smoothly varying onset and release of the wavetable signal. More sophisticated wavetable synthesizers may include frequency-selective filters, provision for frequency vibrato and amplitude tremolo, layering of multiple wavetables, elaborate control interfaces, and so forth.

A critical requirement of a successful wavetable synthesis system is to have carefully prepared wavetable data. A skilled sound designer must perform the process of creating the wavetable source material and ensuring that the loop points are minimally audible.

2 WAVETABLE DATA COMPRESSION PROBLEM

In current mobile applications the size of the wavetable data for a synthesizer is problematic. A set of wavetables for a synthesizer supporting the 175 timbres required by the general MIDI level 1 (GM1) specification [11] is typically at least 1 megabyte (MB), and for high-quality applications it may be 4 MB, 12 MB, or even more. Tens of megabytes of storage may not be an issue in the context of a modern personal computer, but for small, inexpensive, and mobile devices such as cell phones or personal digital assistants (PDAs) even a 1-MB wavetable data set may be orders of magnitude too large for practical use.

The synthesizer designer has several options for decreasing the size of the wavetables, but generally this involves a corresponding loss of signal quality. Some common strategies used in commercial products to reduce the wavetable storage size include the following.

- The length of the one-shot attack sections can be compromised or eliminated.
- The same wavetable data can be reused for many different timbres, either without modification or with realtime changes to disguise the data using amplitude envelopes, filter settings, and so on.
- The synthesizer can be run at a lower sample rate, meaning fewer stored samples per cycle of the waveforms and a lower audio bandwidth.
- The number of bits used to represent each sample in the wavetables can be reduced, such as going from 16 bit per sample to 12 or 8 bit per sample.

The wavetable data can be encoded using a lossy waveform coder, as described next.

3 DIFFERENTIAL REPRESENTATION

Many musically useful signals have a broad autocorrelation function due to the relatively slow amplitude variations of the low-pass waveform. The high sample-to-sample correlation means that the adjacent sample difference signal $d[n] = x[n] - x[n - 1]$ will have a

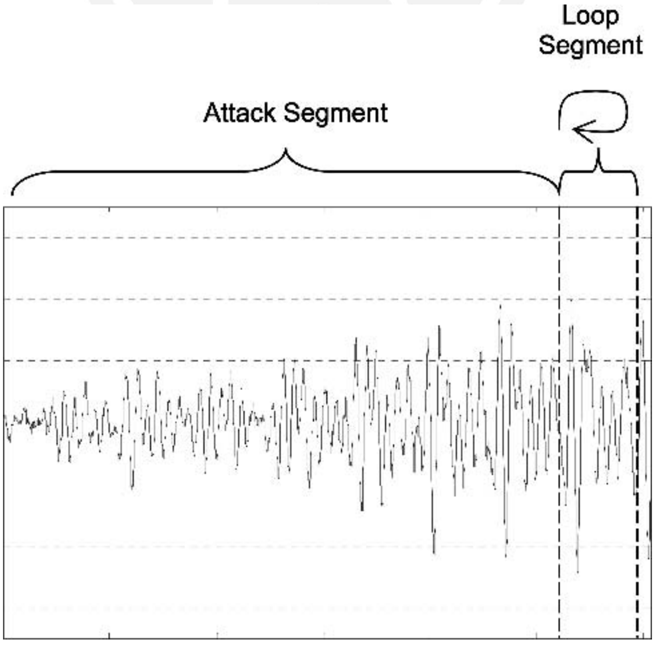


Fig. 2. Example wavetable with one-shot attack segment and sustaining loop segment.

smaller variance than the signal $x[n]$ itself. In other words, the signal is at least somewhat predictable from one sample to the next. We can exploit this redundancy to reduce the required bit rate and thereby represent the signal with fewer total bits [8]. A convenient form of the differential encoder is shown in Fig. 3. Thus the encoder and the decoder will track each other assuming there are no errors in the storage and retrieval process.

Although some data reduction is possible due to the difference signal $d[n]$ being of smaller variance (or average amplitude) than $x[n]$, it is typically necessary to reduce the number of bits per sample still further to obtain a useful amount of compression (such as 30–50% reduction). The usual data reduction approaches for a differential waveform coder are to attempt further redundancy reduction by improving the predictor, and to quantize the difference signal more coarsely by allowing lossy coding, $x_r[n] \neq x[n]$. For the wavetable data compression task it is necessary to keep the decompression computation as fast and simple as possible while still obtaining satisfactory signal quality, so the coarse quantization approach is chosen.

It is clearly an advantage to minimize the audible difference between the original wavetable samples $x[n]$ and the reconstructed wavetable $x_r[n]$. A nonuniform quantizer Q is used to obtain essentially lossless reconstruction when $d[n]$ is small, while allowing some discrepancy if $d[n]$ is large. A quasi-logarithmic quantizer has been found to work well for this purpose.

3.1 Wavetable Compression Details

The use of a lossy differential representation introduces distortion. Conventional waveform coders often treat the coding distortion as an additive uncorrelated white-noise process on a sample-by-sample basis [8]. However, for wavetable compression the coding distortion is embedded in the decoded waveform and is therefore looped over and over along with the desired signal itself, so simple uncorrelated noise models are inappropriate.

The signal-correlated distortion components introduce aliasing when the synthesizer resamples the wavetable to change the musical pitch. Moreover, the correlation between the distortion components and the desired waveform can result in audible noise modulation when the sample rate is changed for vibrato or other musical pitch effects. It is also important to treat the end-of-loop condition carefully so that no discontinuity is introduced at the loop boundary.

These special considerations require extra care during the sound design phase. In particular the sound designer must be made aware of the audible characteristics introduced by the differential encoding process and choose wavetables that reflect the strengths and weaknesses of the encoding system.

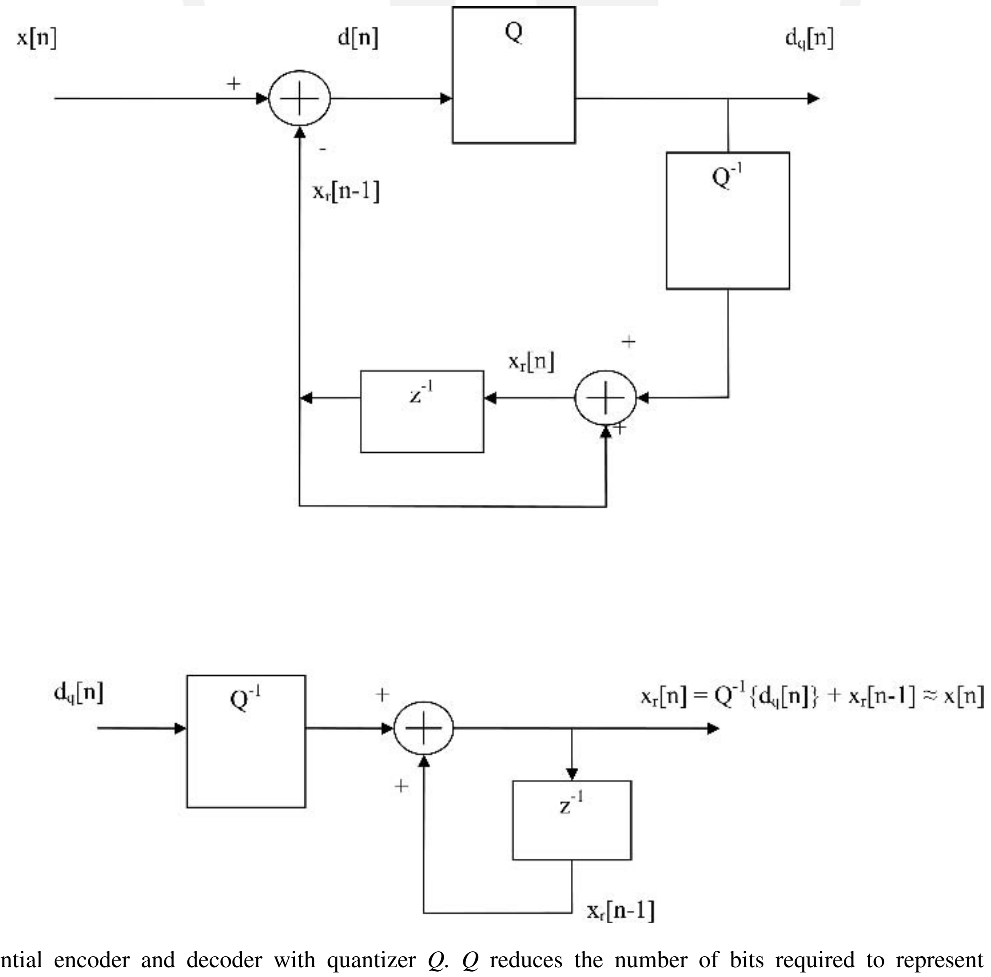


Fig. 3. Differential encoder and decoder with quantizer Q. Q reduces the number of bits required to represent $d[n]$, causing a discrepancy between $x[n]$ and recovered signal $x_r[n]$.

4 IMPLEMENTATION FRAMEWORK

Creating a wavetable sample set requires several steps. One common approach is to obtain recordings of the desired musical timbre, select appropriate attack and sustain (loop) segments, and apply sample-rate conversion and amplitude quantization to achieve a reasonable compromise between wavetable length, word size, and susceptibility to aliasing.

4.1 Encoding Framework

A skilled sound designer performs the encoding process once "at the factory" rather than being something the user must do in the field. Thus it is feasible to expend some sound design effort to adjust the precise sampling rate and amplitude to minimize the encoding error and distortion introduced by the lossy differential wavetable representation. The sound designer must be prepared to evaluate the intermediate results and adjust the waveform parameters as necessary.

4.1.1 Simple Encoding Procedure

An example of the wavetable encoding process is shown in Fig. 4. First the initial (unencoded) value of the wavetable is stored. Next the first difference is calculated using the method of Fig. 4 and a nonuniform quantizer *Q*. An explicit logarithmic formula derived for integer (nonfractional) data is shown, although a look-up table is used in practice. The sequence of quantized differences is determined, and the differential (encoded) values are stored. At the end of the encoded wavetable it may be helpful to calculate the exact unencoded difference between the last accumulated wavetable sample and the first sample of the loop (see decode method 3, Section 4.2.3). If a higher order predictor is used in the encoder, the initial values of the internal predictor coefficients may also be stored for use by the decoder.

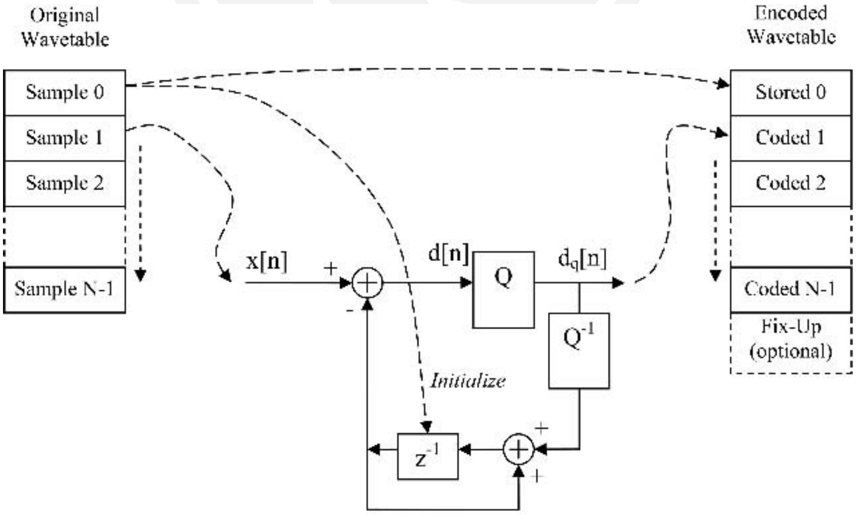
4.1.2 Distortion Minimization

The quantizer's nonuniform steps and lossy behavior make the encoder quite sensitive to the sample-to-sample amplitudes of the original wavetable data. An adaptive quantization procedure can be considered as a way to minimize the coding error, but the fixed nature of the wavetable and the loop constraint make any time-adaptive algorithms inappropriate.

Instead we use a systematic search for the waveform amplitude resulting in minimum distortion. The nonlinear nature of the differential quantization process prevents an easy analytical solution, but the availability of fast computers makes an iterative optimization feasible during sample set creation.

The process works as follows. During the sound design process the wavetable data are automatically encoded repeatedly with different amplitude scalings, and the total mean-square discrepancy between the decoded wavetable and the original data is calculated. The amplitude scaling that results in the lowest distortion (highest signal-to-error ratio) is then stored for use by the synthesizer. This simple search procedure is depicted in Fig. 5.

An example output from the search procedure shows clearly the complicated relationship between the signal amplitude and the coding error (see Fig. 6). The measured signal-to-error ratio is plotted as a function of the amplitude scaling parameter applied to the input wavetable prior to encoding. The automatic procedure chooses the appropriate gain factor within the synthesizer design constraints on numerical range, overflow protection, and so forth. As in this example, the optimization procedure typically provides an improvement of several dB in the wavetable's raw signal-to-error ratio—the equivalent of nearly 1 bit of precision in this case. It is also possible to consider a perceptual quality model instead of the signal-to-error ratio during the optimization, but the simple error criterion has been found satisfactory in practice.



$$d_q[n] = Q\{d[n]\} = \mathrm{sgn}(d[n]) \cdot \mathrm{floor}\{0.5 + K \cdot \log_{10}(|d[n]|)\}$$

$$Q^{-1}\{d_q[n]\} = \mathrm{sgn}(d_q[n]) \cdot 10^{|d_q[n]|/K}$$

Fig. 4. Wavetable encoding process using log-based nonlinear quantizer. Coded values have fewer bits of precision than original samples. *K*—scaling constant chosen to provide desired encoded signal resolution.

The amplitude scaling parameter can often be combined with the other wavetable gain scaling parameters into a single factor within the synthesizer patch architecture, thereby avoiding any additional decoder computation. Thus the optimized encoding process requires little if any change to the decoder and synthesizer architecture.

4.2 Decoding Framework

While the encoding process can involve iterative procedures and the services of a skilled sound designer, the decoding process must be extremely efficient and sufficiently fast to allow real-time operation. We find that the precise nature of the decoder varies from one product platform to another, but it is generally necessary to keep the decoding complexity commensurate with the wavetable synthesis process itself. This implies that a look-up table in the decoder might be preferable to an algorithm to compute exponentials or transcendental functions. Three decoding scenarios are described next.

4.2.1 Method 1–Full Decode Buffer

The first wavetable reconstruction scenario is to use a decode buffer holding the entire decoded wavetable. The encoded data are read from storage and the entire wavetable loop is decoded into a buffer, or cache, for the synthesizer. This method requires enough buffer storage for the entire wavetable loop, but only the active wavetable data need to be present. The decode operation occurs only when a new musical note calls for a different wavetable, that is, a MIDI patch change.

The advantage of the full decode buffer is that it allows the synthesizer functions to be simplified since the decoding operation is accomplished outside of the sample-by-sample computation of the synthesizer. It is also possible for multiple concurrent synthesized voices to share a single cached wavetable. The primary disadvantages of this approach are the need to delay playback until the entire wavetable is decoded and the obvious need for

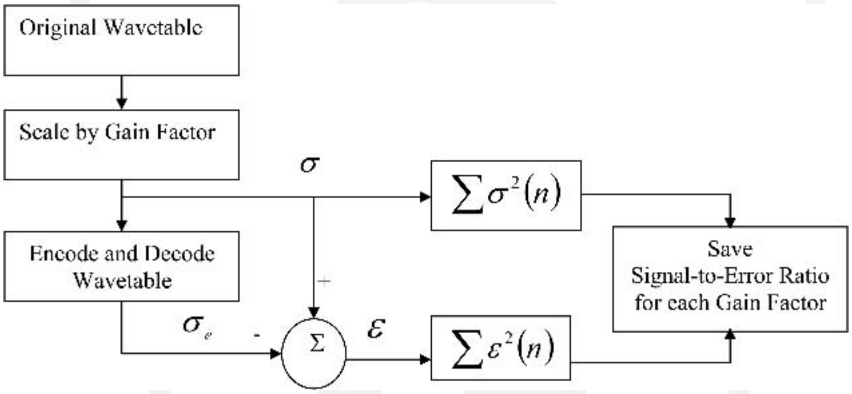


Fig. 5. Systematic procedure to optimize wavetable amplitude for best signal-to-error ratio. Nonlinear behavior of encoding process can result in coding error being sensitive to waveform amplitude.

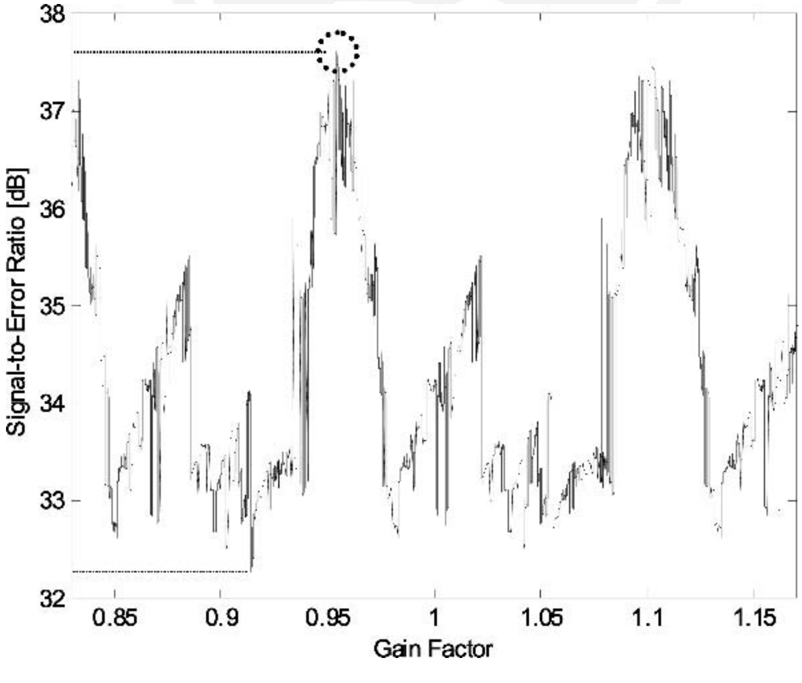


Fig. 6. Example signal-to-error ratio as a function of wavetable amplitude. Varying the signal amplitude and reencoding can find the best signal-to-noise ratio within acceptable amplitude limits. In this example a roughly 5-dB improvement is possible by selecting gain factor to maximize signal-to-error ratio, such as 0.96.

substantial buffer memory to hold the active wavetable cache.

4.2.2 Method 2–Accumulator with Loop Reset

In this alternative the synthesizer fetches and decodes only sufficient wavetable data "on the fly" to produce the current block of several output samples. An accumulator variable is loaded with the initial wavetable value and then each decoded wavetable sample is obtained by adding (accumulating) the sequence of differential values over the required LUI (look-up index) range for the current block. Once the current wavetable block is synthesized the last value of the accumulator is saved for use in the next block. When the LUI traverses the end of the wavetable, the accumulator is simply loaded again with the initial wavetable value and the process continues.

The accumulator method has the advantage that minimal buffer storage is required. It also allows the synthesis process to begin more rapidly than method 1, since only the immediately required wavetable samples need to be decoded, not the entire table. However, the accumulator method has the disadvantage that the wavetable fetch and decoding operations must occur over and over as long as the synthesizer sustains the note, thereby increasing the average computational cost per output sample. The inner loop of the synthesizer must also be modified to handle the additional wavetable decoding operations.

A more subtle disadvantage is that the differential encode/decode implies that every encoded wavetable element must be fetched and accumulated even if the intervening samples are not actually needed by the synthesizer—the current wavetable sample depends on all the prior differential values. For example, if the sample increment is greater than unity and only simple linear interpolation is performed, certain wavetable samples will be skipped as the LUI hops through the wavetable, but the skipped samples must still be computed due to the sequential differential representation of the wavetable.

4.2.3 Method 3—Accumulator with Loop Fix-Up

The third implementation alternative is similar to method 2, except a special fix-up value is used rather than resetting the decode accumulator at the end of the loop. The fix-up value allows a perfect loop accumulation by holding the precise unencoded difference between the accumulated differential values at the end of the wavetable and the value at the start of the loop. The fix-up is precalculated during the encoding process and stored at the end of the wavetable. A key advantage of this method is that it can be implemented with a symmetrical structure that does not require explicit end-of-loop calculations. By reserving one of the code words to indicate that the fix-up value is to be fetched, the decode/synthesis process automatically handles the loop transition without needing any side information. This approach is particularly suited to situations in which the sample fetch, decoding, and interpolation processes are performed by special-purpose hardware.

5 OTHER OPTIONS

We have recently simulated several alternatives for the wavetable compression/decompression task. Two such options, described briefly in this section, have the potential to improve the compression ratio for a given level of quality, and also to improve the efficiency of the synthesis process.

5.1 Log Encoding to Reduce Multiplies

In typical wavetable synthesis applications the samples recovered from the wavetable are interpolated using a multiplicative weighting, and then further multiplied by a time-varying amplitude envelope to simulate the attack and release characteristics of the desired timbre. The multiplication operations may be costly in a custom silicon implementation due to the size and complexity of the multiplier hardware, or the need to time multiplex access to the multiplier unit. However, by using a true logarithmic encoding of the wavetable differential values and representing the amplitude envelope in logarithmic form, it has been suggested that the interpolation and amplitude scaling could be performed by adding the logarithms rather than requiring explicit multiplies [12]. An antilog look-up table in the decoder can then be used to reconstruct the output waveform.

5.2 Model-Based Compression

Another encoding alternative consists of a linear prediction model. In this case the wavetable is considered to be a portion of the impulse response of a recursive digital filter. The encoding process involves determining the parameters of the filter model, selecting a suitable excitation function such as a noise burst or impulse, and minimizing the coding distortion. During synthesis the filter coefficients and initial filter state are fetched, and the excitation function is used to regenerate the wavetable. If the complexity of the filter and the excitation function can be minimized, the model approach can further reduce the storage needed for wavetable synthesis.

6 CONCLUSION

The differential wavetable encoding method can achieve data compression factors of 30–50% with acceptable quality. This corresponds to representing the original 16-bit audio samples with only 8-bit nonlinear differential values, while still achieving 12-bit quality. As described in this communication, the peculiar coding problems presented by the looped wavetable synthesizer can be ameliorated to a great extent by careful sample set selection and effort in coding optimization during the encoding process, yet without affecting the inherent low complexity of the differential decoder. Thus a single compressed sample set delivered to the mobile playback device can be used repeatedly with low bandwidth synthesizer command messages, such as mobile MIDI [1].

7 REFERENCES

[1] MIDI Manufacturers Association, *GM Lite Specification and Guidelines for Mobile Applications* (MMA, La Habra, CA, 2001).

[2] "Ring Tones Bringing in Big Bucks," *Wired News/Reuters* (2004 Jan. 13); www.wired.com/news/business/0,1367,61903,00.html.

[3] M. Mathews, *The Technology of Computer Music* (MIT Press, Cambridge, MA, 1969).

[4] C. Dodge and T. Jerse, *Computer Music: Synthesis, Composition, and Performance,* 2nd ed. (Schirmer, New York, 1997).

[5] R. Bristow-Johnson, "Wavetable Synthesis 101, A Fundamental Perspective," presented at the 101st Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts),* vol. 44, p. 1176 (1996 Dec.), preprint 4400.

[6] M. Hans and R. W. Schafer, "Lossless Compression of Digital Audio," *IEEE Signal Process. Mag.,* vol. 18, no. 4, pp. 21–32 (2001).

[7] K. Brandenburg, "Perceptual Coding of High Quality Digital Audio," in *Applications of Digital Signal Processing to Audio and Acoustics,* M. Kahrs and K. Brandenburg, Eds. (Kluwer, Boston, MA, 1998).

[8] N. S. Jayant and P. Noll, *Digital Coding of Waveforms* (Prentice-Hall, Englewood Cliffs, NJ, 1984).

[9] D. Rossum, "An Analysis of Pitch-Shifting Algorithms," presented at the 87th Convention of the Audio Engineering Society, *J. Audio Eng. Soc. (Abstracts),* vol. 37, p. 1072 (1989 Dec.), preprint 2843.

[10] D. C. Massie, "Wavetable Sampling Synthesis," in *Applications of Digital Signal Processing to Audio and Acoustics,* M. Kahrs and K. Brandenburg, Eds. (Kluwer, Boston, MA, 1998).

[11] MIDI Manufacturers Association, *Complete MIDI 1.0 Detailed Specification,* version 96.1 (MMA, La Habra, CA, 2001).

[12] E. Lindemann, "Audio Data Decompression and Interpolation Apparatus and Method," U.S. patent 5,890,126 (1999).

THE AUTHOR

Robert C. (Rob) Maher was born in 1962 in Cambridge, UK, of American parents. He received a B.S. degree from Washington University, St. Louis, in 1984, an M.S. degree from the University of Wisconsin–Madison in 1985, and a Ph.D. degree from the University of Illinois–Urbana in 1989, all in electrical engineering. While a student he worked as a graduate research assistant with James Beauchamp at the University of Illinois Computer Music Project, supported by a National Science Foundation Graduate Fellowship and an Audio Engineering Society Educational Grant.

Dr. Maher was a professor with the Department of Electrical Engineering at the University of Nebraska–Lincoln from 1989 to 1997. In 1997 he left academia to join EuPhonics, Inc., of Boulder, CO, serving as vice president of engineering. When EuPhonics was acquired by 3Com Corporation in 1998 he became the engineering manager for audio product development for 3Com/U.S. Robotics. In 2001 he became sole proprietor of a digital audio signal processing consulting company, serving a variety of clients in the computer audio and multimedia industry. He also taught part-time as an adjunct professor at the University of Colorado, Boulder. He formally reentered the academic field in 2002 by joining the Department of Electrical and Computer Engineering at Montana State University, Bozeman, where he is currently an associate professor. His teaching and research interests lie in the application of advanced digital signal processing methods in audio engineering, environmental sound classification, and music synthesis.

Dr. Maher is a member of the Tau Beta Pi, Eta Kappa Nu, Phi Kappa Phi, and Sigma Xi honor societies and of several professional organizations, including the Audio Engineering Society, IEEE, ASA, and ASEE. He was the chair of the AES Colorado Section from 1999 to 2001 and served as papers cochair for the 2004 AES 117th Convention.