REAL-TIME SPEECH SYNTHESIS ON AN ULTRA LOW-RESOURCE, PROGRAMMABLE DSP SYSTEM

Hamid Sheikhzadeh, Etienne Cornu, Robert Brennan, and Todd Schneider
Dspfactory Ltd., 80 King Street South, Suite 206, Waterloo, Ontario, Canada N2J 1P5
E-mails: {hsheikh, rob.brennan, todd.schneider}@dspfactory.com, etienne.cornu@chamblon.com

ABSTRACT

An efficient implementation of a time-domain speech synthesis system on an ultra low-power, miniature, programmable block-floating-point DSP system is introduced. The DSP system, operating at a clock rate as low as 1.28 MHz, is well suited for speech and audio processing applications. Similar to the MBR-PSOLA technique, this time-domain synthesis method uses a normalized speech database generated by a high-quality harmonic synthesis. To reduce the memory usage and communication bandwidth, the normalized database is compressed using a block-adaptive, ADPCM approach. Listening tests comparing the synthetic speech quality on the DSP system and the same method implemented on a high-resource computer system show no degradations due to the memory, register length, or other low-resource limitations on the DSP system. The system consumes less than 1 mW at 1 volt.

1. INTRODUCTION

The objective of this research is to implement a real-time speech synthesis system on an ultra low-power (less than 1 mW at 1 volt), ultra small size, and low-cost platform (referred to as a "low-resource" platform in this paper). Achieving this objective requires the synthesis method to be optimized for low memory, and low computational resources. At the same time, it has to provide an acceptable synthesized speech quality with a minimal time-delay. A speech synthesizer working on a low-resource platform such as described here significantly extends the applications of the speech synthesis technology due to its lower cost, lower power, and smaller size.

There are various methods available to solve the speech synthesis problem. The most successful methods use an inventory of pre-recorded speech units (like diphones), and concatenate the units (with or without some prosodic modifications) to synthesize fluent speech with correct prosody. Through the employment of effective unit selection methods, one can achieve high quality synthesized speech and avoid the prosodic modification of speech units by recording a very large inventory of units and searching for optimal units to be concatenated at the synthesis stage. However, this requires a large memory to store the unit inventory, and the search for optimal units at the synthesis stage is complicated. Thus, such speech synthesis systems are not suitable for implementation on our low-resource platform (see Section 2).

The next alternative synthesizers to the unit selection systems are the class of small-unit concatenation systems that use less than a few thousand speech units. Due to many reasons (see [1] for a detailed discussion), mostly diphones have been used as speech units and as a result, "diphone concatenation systems" have been developed for many languages. The number of diphones used in most languages is less than a few thousand (e.g., 2000 for English and French), leading to lower memory usage (compared to unit selection methods) after data compression. Since time-domain diphone concatenation systems require less computation than spectral-domain methods [1,2], an obvious choice for a low-resource speech synthesizer is a time-domain diphone concatenation system. Amongst the various versions of these systems proposed in the literature, the Time-Domain Pitch-Synchronous Overlap and Add (TD-PSOLA) method [3,4] is very simple and offers a high speech quality if the problems of pitch, phase and spectral discontinuities are properly addressed.

For a low-resource implementation, it is desirable to do most of the computations and normalizations off-line to simplify the on-line synthesis. In the Multi-Band Resynthesis Pitch-Synchronous Overlap-Add (MBR-PSOLA) method [5], the speech unit database is carefully re-synthesized to obtain a constant-pitch, fixed-phase (up to a cut-off frequency) database for time-domain synthesis. This off-line re-synthesis stage is called normalization or re-harmonization. The two important problems of pitch and phase discontinuity are thus resolved offline, at a cost of minor quality losses in the normalization process. Due to the pitch and phase normalizations, the spectral discontinuities (across the units) can then be resolved using a simple time-domain interpolation. In a recently proposed method [2], Dutoit et al. have implemented a modified version of the MBR-PSOLA synthesis. According to this method, at the normalization stage of speech units at a constant pitch, for each analysis frame (of almost two pitch periods), only one of the two re-synthesized pitch periods is synthesized and stored. As reported, this leads to less memory usage and a better separation of periodic and stochastic waveforms.

In PC-based synthesis systems, speech is synthesized into temporary files that are played back when a part of the text (typically complete sentences) has been processed. In contrast, in a real-time system, the synthesis cannot be interrupted once it has started. Also, synthesis is not a straight-through process in which the input data can be simply synthesized as it is made available to the processor. The processor has to buffer enough data to account for variations in prosody. It also has to work on several frames at a time in order to perform interpolation between units while synthesis is taking place. In our implementation, a low-resource system performs these operations in real-time, thereby significantly off-loading a host processor while using only a very limited amount of resources.

In the next sections, we briefly describe the low-resource DSP system that is used, and the methods selected for speech synthesis and data compression. Next, we describe the implementation of the real-time synthesis method on the DSP system. Finally conclusions and future work are discussed.

2. THE DSP SYSTEM

Figure 1 shows a block diagram of the DSP system [6,7]. The DSP portion consists of three major components: a weighted overlap-add (WOLA) block-floating point filterbank, a 16-bit DSP core, and an input-output processor (IOP). The system is implemented on two ASICs. A digital chip on 0.18μ CMOS technology contains the DSP core, RAM, the WOLA filterbank, and the IOP. The mixed-signal portions are implemented on 1 μm CMOS. A separate, off-the-shelf E^2PROM provides the non-volatile storage. The RAM consists of two 4-kword data spaces and a 12-kword program memory space. Additional shared memory for the WOLA filterbank and the IOP is also provided. The core provides 1 MIPS/MHz operation and has a maximum clock rate of 4 MHz at 1 volt. At 1.8 volts, 30 MHz operation is also possible. The system operates from a 1 volt (single battery), and consumes a power of less than 1 mW. Prototype versions of the chipset are packaged into a 6.5 x 3.5 x 2.5 mm hybrid circuit.

3. TIME-DOMAIN SPEECH SYNTHESIS METHOD

We use a method similar, in principle, to MBR-PSOLA [5], with some modifications at the normalization stage. A speech unit database (e.g. a diphone database) is first processed offline by a spectral method to get a normalized database that has a nominal constant pitch frequency (F0=1/T0) and a phase that is fixed (up to a cut-off frequency less than 3 kHz) for all units. The normalization method could be any high-quality speech synthesis method that is capable of synthesizing speech at a constant pitch. Using speech synthesis systems such as the Harmonic + Noise Model (HNM) [8] and the hybrid Harmonic/Stochastic model (H/S) [9], the speech frames of around two pitch periods in duration are first analyzed. Then the constant-pitch and fixed-phase "elementary waveforms" are synthesized for each frame. The elementary waveforms can have duration of one pitch period (T0) if the synthesized elementary waveforms are assumed (as in [2]) to be perfectly periodic. However, if they are not assumed to be perfectly periodic, the elementary waveforms should have a length of more than one pitch period (2T0 or more). For naturally uttered speech, the perfect periodicity assumption does not hold for almost all the unvoiced sounds, and for many classes of voiced sounds such as voiced fricatives, diphthongs, and even for some vowels. This means that two consecutive pitch periods are not exactly the same for most voiced sounds. Thus, we relax the perfect periodic assumption and use an elementary waveform length of 2T0. As a result, time-varying and powerful methods like the HNM can be employed to achieve a better quality. The elementary waveforms are then multiplied by a Hanning window, and processed through an OLA stage to obtain a normalized speech waveform. The re-synthesized units in the database now can be used for a time-domain concatenation, without any concerns about the pitch and phase discontinuities. The spectral discontinuities can be removed through simple time-domain interpolation [5]. The interpolation process is limited to the voiced sounds. As a result of using synthesis models (such as HNM) that are capable of modelling the speech time variations within a few pitch periods, the system can achieve better speech quality. Also, the synthesized speech frames are now more compressible. To explain this, note that if the elementary waveforms were assumed to be one period long, there would be unavoidable discontinuities (at frame boundaries) in the normalized database due to the frame-to-frame acoustic variations. However, when the OLA synthesis is employed to obtain normalized speech using longer elementary units (of length 2T0), there won't be any jumps or discontinuities in the normalized units due to the OLA smoothing. As a result, the units can be compressed more by adaptive-predictive methods as described below.

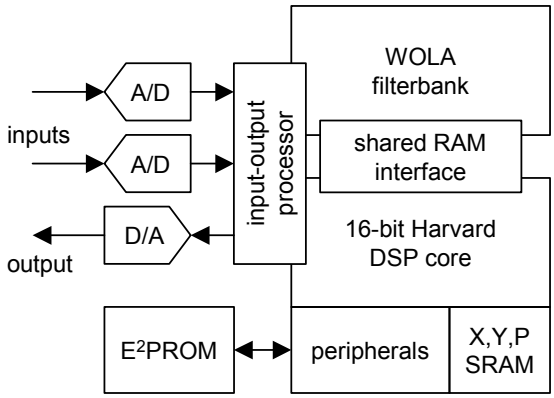

Figure 1: The DSP system block diagram

3.1. Database Compression

The normalized units all have the same pitch periods, and due to the phase normalization in the re-synthesis process, the consecutive frames are very similar, at least for the voiced sounds. The proposed compression method is based on exploiting both the interframe and intraframe correlation of the normalized speech. The database compression is done off-line and once for all speech units, and the voiced/unvoiced status of the frames is accurately known. Thus, rather than using a general-purpose speech compression technique, we employ a variant of the classical ADPCM carefully optimized to make use of the database features. The objective is to achieve a high compression ratio while preserving the decoder simplicity. To conserve resources, the decoder is constrained to employ only fixed-point additions and bit-shifting, with no multiplies.

The block diagram of the compression method is shown in Figure 2. For voiced frames, the difference between a sample value and the value of the corresponding sample in the previous period (frame prediction error) is found. For unvoiced sounds, the speech waveform itself replaces the frame prediction error. Since the consecutive frames are very similar for the voiced sounds, the frame prediction error has a smaller dynamic range than the speech waveform itself. Also the unvoiced sounds naturally have a smaller dynamic range than the voiced sounds. A block-adaptive differential PCM (DPCM) quantizer is then used to quantize the prediction error. A single quantization step is adapted for each block (one pitch period) as follows. First, the

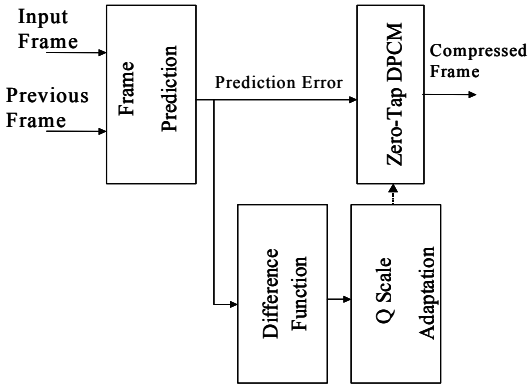

Figure 2: Data compression block diagram

(first-order) difference function of the prediction error is calculated, and the maximum of its absolute value is found. Based on this maximum value, the quantization step is then scaled for each period so that there is minimal data clipping in the quantization process. The frame prediction error is then scaled by the quantization scale, and compressed with a zero-tap DPCM quantizer. For each frame, the DPCM signal and the quantization scale are then stored. To simplify the decoding process, the scale factor is constrained to be a power of two. As a result, we need to simply bit-shift the samples (instead of multiplying/dividing them by the scale factor) at the decoding stage.

The decoding process is straightforward. First the compressed values of a frame are bit-shifted (using a single shift value for each frame) to compensate for the quantization scaling. Then two accumulations are applied: one over the frames and one inside each frame. The computation cost of the decoding method is thus two fixed-point additions and one bit-shift per sample. This is much less than the 4.9 (possibly floating point) operations per sample reported in [10]. The algorithm uses only 13 CPU cycles (on average) to decompress each 16-bit sample. Informal tests showed that the compression was transparent to the listener at a compression rate of 4. At a compression rate of 5.3 the quality dropped to toll quality. Although the compression ratio is less than that reported in [10] for a comparable quality, the computation cost is low enough for the method to be implementable on our low-resource platform.

4. USE OF THE DSP SYSTEM

The synthesis system provides the back-end of a complete text-to-speech system. The front-end, running on another platform (the host), takes the text to synthesize as input from a user. The front-end first converts the text into a sequence of diphone labels and calculates a number of prosodic parameters that control the speech pitch and rate. The front-end then extracts diphones from a compressed database, and transmits them to the synthesizer on the DSP core along with the prosodic parameters. The software on the DSP core decompresses the data and synthesizes it as specified by the prosodic parameters.

4.1. Technical Challenges

The synthesis algorithm implemented on our DSP system operates in real-time; once activated by the host, the device is capable of synthesizing speech for as long as needed. It buffers only enough data needed for unit interpolation and to account for the variations in time scaling (for some period of time the system needs more or less input data to produce the same amount of output data). Interpolation occurs typically over up to 5 frames (about 35 milliseconds) and the entire system is designed so that up to 30 frames can be stored in memory at the same time, producing a latency of up to 210 milliseconds.

The sampling rate is 16kHz with a precision of 16 bits per sample. Therefore, with a clock speed of 1.28 MHz, the chip needs to output a 16-bit sample every 80 CPU cycles. Because of the variations in time scaling, i.e. if more input data is needed than is produced, not all of the 80 cycles are available for each sample. The fluctuations in time-scaling are controlled by the host. In our case, they are limited to about +/- 25% of the output rate. So, a general rule for the design of the software was that only about 60 cycles per output sample were available. During these 60 cycles, the chip must perform the following operations: accept the data from the host, decompress and store it; then retrieve it, perform interpolation, windowing, overlap-add and store it again; then finally retrieve it and send it to the output processor. There is also some overhead for buffer management. Accepting data from the host already takes an average of 30 cycles per 16-bit sample because an interrupt is generated for every 8-bits of data. Decompression and storage uses about 13 cycles per 16-bit sample. The interpolation, windowing and overlap-add stage take about 8 cycles per sample and finally the output operation takes 4 cycles per sample.

4.2. Implementation Architecture

The synthesis system includes the following components as shown in Figure 3:

- Host Interface module
- Data Decompression module
- Overlap-Add module

The synthesis system receives data of two types from the host: 1) Data frames, which are made of (compressed) frames containing L contiguous speech samples of a pitch period. 2) Prosody scripts, which include all the prosodic and interpolation information.

The Host Interface module accepts data packets from the host, determines their type (frame or prosody script) and dispatches them to the Data Decompression module.

4.2.1. Data Decompression Module

The Data Decompression module reads the compressed frames and prosody scripts, applies the decompression algorithm and stores the decompressed data into the corresponding buffer. Data frames are decompressed using the algorithm described in Section 3.1, that takes as input L-sample compressed speech frames. Prosody scripts vary in length, and are compressed to occupy 3 bytes per frame. To save more power and at the cost of a larger size, we can alternatively store the compressed speech unit database in a non-volatile memory that is directly accessible by the DSP system over existing interfaces.

4.2.2. Overlap-Add Module

The Overlap-Add module is the major part of the synthesis system. It loops through the prosody script entries sent by the

host. The prosodic information contained in the scripts include:

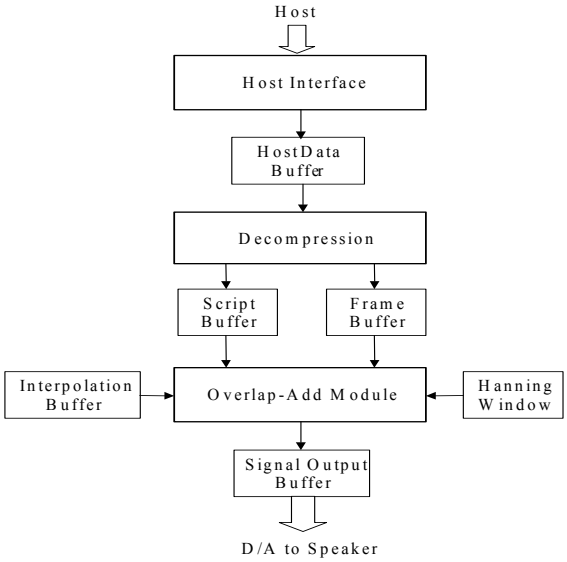

Figure 3: The Synthesis System Architecture

1) Shift. Amount by which to shift the data out to the signal buffer after the overlap-add. Shifted samples are stored in the signal buffer. They are then read by the IO-processor, transmitted to system's Digital to Analog converter and from there to the speaker.**2) Interpolation data.** Gives the diphone boundary and the interpolation depth, assumed to be equal for both sides of the interpolation at unit boundaries. **3) Frame reverse flag.** Repeated unvoiced frames are time-reversed by the Overlap-Add module to avoid tonal noise distortions [4].

For each script entry, the Overlap-Add module performs the following operations.1) Build a 2L-sample frame from the L-sample frame referenced by the script and the L-sample frame that follows. 2) If necessary, reverse the frame. 3) Calculate the interpolation values at the unit boundaries. 4) If necessary, add a proper multiple of interpolation values to the L sample frame. Interpolation between frames is applied only at diphone boundaries when the two units are both voiced. 5) Apply a Hanning window to the frame. 6) Overlap-Add the 2L-sample frame at the beginning of the output signal queue. 7) Shift out the number of values specified in the script and adjust the signal queue pointer.

5. CONCLUSION AND FUTURE WORK

The proposed system was implemented on the low-resource DSP hardware platform described in [6,7] operating at a 1.28 MHz clock rate. This was made possible due to the optimized hardware architecture and the efficient instruction set of the DSP system. To the best of our knowledge, this speech synthesis system is of smaller size and lower power consumption than any other speech synthesizer ever built.

To test the performance of the system, we subjectively compared the speech synthesis quality on the hardware platform with the simulation results of the same system working on a PC without any data compression or register length limitations. The evaluations showed that the limitations imposed by the hardware, in terms of memory usage and quantization effects, did not lead to any noticeable speech quality degradation. However, the data compression had the greatest effect on the quality. While the quality degradation was not audible at a compression rate of 4, the quality dropped to a toll quality at a compression ratio of about 5.3.

We have demonstrated that an ultra-low power DSP system can be effectively used to perform the back-end tasks of a synthesis system. This would particularly be useful in environments where power consumption must be reduced to a minimum or where an embedded processor does not have the capabilities to synthesize speech. For example, it could be used in a personal digital assistant (PDA) to off-load the main processor in an efficient manner. The chip could also be used in conjunction with a micro-controller in embedded systems.

Our future work is directed towards the front-end text-to-phoneme (TTP) parser and prosodic analyzer. Currently, a host computer system (e.g. a PC or a PDA) functions as the front-end processor. Based on our evaluations, the processes at the front-end are more memory-bound rather than CPU-bound. Thus, we are now in the process of employing our hardware platform with some external memory for the TTP parsing and the prosodic analysis.

6. REFERENCES

[1] T. Dutoit, *An Introduction to Text-to-Speech Synthesis,* Kluwer Academic Publishers, 1997.

[2] T. Dutoit et al., "Envelope-Invariant Analytical Speech Resynthesis Using Periodic Signals Derived from Reharmonized Frame Spectrum", United States Patent 5,987,413, Nov. 16, 1999.

[3] F. Charpentier and M. G. Stella, "Diphone Synthesis Using an Overlap-Add Technique for Speech Waveforms Concatenation", *Proc. of the ICASSP 1986*, pp. 2015-2018.

[4] E. Moulines, and F. Charpentier, "Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones", *Speech Communication*, vol. 9, no. 5-6, 1990.

[5] T. Dutoit, and H. Leich, "MBR-PSOLA Text-to-Speech Synthesis Based on an MBE Re-Synthesis of the Segments Database", *Speech Communication,* vol. 13, pp. 435-440, Nov. 1993.

[6] R. Brennan and T. Schneider, "Filterbank Structure and Method for Filtering and Separating an Information Signal into Different Bands, Particularly for Audio Signal in Hearing Aids". United States Patent 6,236,731. WO 98/47313. April 16, 1997

[7] R. Brennan and T. Schneider, "A Flexible Filterbank Structure for Extensive Signal Manipulations in Digital Hearing Aids", *Proc. IEEE Int. Symp. Circuits and Systems,* pp.569-572, 1998.

[8] Y. Stylianou, "On the Implementation of the Harmonic Plus Noise Model for Concatenative Speech Synthesis", *Proc. ICASSP 2000,* pp. 957-960.

[9] T. Dutoit, and B. Gosselin, "On the Use of Hybrid Harmonic/Stochastic Model for TTS Synthesis-by-Concatenation", *Speech Communication,* 19, pp. 119-143.

[10] O. Van Der Vreken et al., "A Simple and Efficient Algorithm for the Compression of MBROLA Segment Databases", *Proc. Eurospeech 97,* Patras, pp. 241-245.