A Perception-Aware Low-Power Software Audio Decoder for Portable Devices

Samarjit Chakraborty Ye Wang Wendong Huang
Department of Computer Science
National University of Singapore
email: {samarjit, wangye, huangwd}@comp.nus.edu.sg

Abstract

We propose a new software audio decoder for processors supporting multiple discrete voltage-frequency operating points. The proposed decoding scheme allows the user to switch between multiple output quality levels, where each level is associated with a different rate at which the processor consumes energy. This will be an attractive feature in battery-powered portable audio players and mobile phones, where battery-life is often more crucial than the output quality, especially in noisy environments. Towards this, the frequency range of the decoder is partitioned into multiple groups, in accordance with their perceptual relevance. When a longer battery life is desired, only the most relevant frequency components are decoded, which allows the processor to be run at a lower voltage and frequency. We have implemented this scheme using the MP3 decoder and obtained up to 95% *savings in the energy consumed by the processor for AM quality output (in contrast to CD quality output, which is associated with the maximum energy consumption). This scheme is easy to implement, has no runtime overhead and does not involve any runtime voltage or frequency scaling.*

1 Introduction

Lately, there has been a considerable interest in power management schemes for portable devices running multimedia applications. In contrast to well-studied run-time techniques such as dynamic voltage scaling and dynamic power management, in this paper we address the problem of power consumption from a different perspective. We propose a new software audio decoding scheme that allows the *user* to switch between multiple output quality levels. Each such level is associated with a different rate of energy consumption, and hence battery lifetime. Our scheme is *perception-aware*, in the sense that the difference in the perceived output quality associated with the different levels is relatively small. But decoding the same audio clip at a lower output quality level leads to significant savings in the energy consumed by the processor.

A high-level block diagram of our multi-level perception-aware decoder is shown in Figure 1. The decoding level chosen by the user to decode any audio clip determines (i) the decoding scheme or algorithm that is run

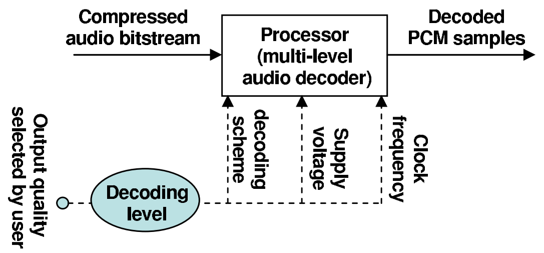
Compressed audio bitstream
Processor (multi-level audio decoder)
Decoded PCM samples
Output quality selected by user
Decoding level
decoding scheme
Supply voltage
Clock frequency

Figure 1: A multi-level perception-aware decoder.

on the processor, and (ii) the voltage and the frequency with which the processor is to be run. It may be noted that in contrast to many dynamic voltage/frequency scaling techniques, our scheme does not involve any runtime scaling of the processor voltage or frequency. Given a processor with a fixed number of voltage-frequency operating points, the decoding levels in this scheme can be tuned to match these operating points.

The proposed scheme relies on partitioning the frequency bandwidth of an audio decoder (like the MP3 decoder) into a number of groups, that is equal to the number of decoding levels (see Figure 1). These groups are ordered according to their perceptual relevance. If there are four levels of decoding, i.e. Levels 1–4, then the frequency bandwidth group that has the highest perceptual relevance is associated with Level 1 and the group that has the lowest perceptual relevance is associated with Level 4. As a result, a reasonable output quality can be obtained by decoding only the Level 1 bandwidth group. To obtain the best output quality, all the bandwidth groups are decoded (which is what a standard decoder does). Such a partitioning of the frequency bandwidth into four levels in the case of MP3 is shown in Table 1[1]. Column 2 in this table (showing the Decoded subband index) is explained in Section 2.

The attractiveness of this scheme stems from the fact that although the computational workload associated with the decoding process scales almost linearly with the decoding level, the lower frequency ranges have a much higher perceptual relevance compared to the higher ones. Therefore, when a clip is decoded at a lower level, by sacrificing only a small fraction of the output quality, the processor

[1]AM and FM stand for Amplitude and Frequency Modulation. Here we refer to their associated qualities in the context of radio broadcasting.

Decoding level	Decoded subband index	Frequency range (Hz)	Perceived quality level
Level 1	0 – 7	0 – 5512.5	AM quality
Level 2	0 – 15	0 – 11025	Near FM quality
Level 3	0 – 23	0 – 16537.5	Near CD quality
Level 4	0 – 31	0 – 22050	CD quality

Table 1: Four decoding levels for the MP3 codec.

can be run at a much lower clock frequency (and voltage), when compared to a higher decoding level. This scheme does not rely on any specific hardware implementations of the decoder, or on any coprocessors to implement specific parts of the decoder. The significance of our work stems from the fact that currently many consumer electronics products are being built using general-purpose hardware platforms, or architecture templates [10] (e.g. OMAP from Texas Instruments and PrimeXsys from ARM). Hence, increasingly there is a shift in focus towards appropriate software-implementations of different functionalities, rather than tailor-made hardware for different applications. Many portable audio players (such as MP3 players) today indeed used software decoders. It is also easy to foresee that soon it would be common to use PDAs or mobile phones (with powerful but general-purpose voltage and frequency scalable processors) as portable audio/video players, by running a suitable decoder application. Our solution will be useful in a scenario like this, where hardwired audio/video decoder chips implementing a specific codec will be of limited use.

Related work: There exists a large volume of recent work on dynamically scaling the voltage and frequency of a processor in response to the variable workload involved in processing multimedia streams (see [1, 5, 6, 7] and the references therein). A second line of work proposes the use of buffers to smooth out the burstiness in multimedia streams and to decouple two architectural components having different processing rates. This enables periodically switching off the processor or running it at a lower frequency, thereby saving energy [3, 8, 11]. A number of papers have also addressed the problem of guaranteeing some Quality-of-Service (QoS) requirement associated with multimedia applications and at the same time minimizing the processor's energy consumption [14, 17]. This is again achieved by using appropriate scheduling techniques. Our work in this paper is fundamentally different from all the above proposals and does not involve any kind of scheduling or dynamic power management strategy.

Recently, a scheme for exploring QoS versus energy tradeoffs was presented in [12] for processing MPEG video streams. With frame resolution as the QoS metric, this work would enable a designer to evaluate potential energy savings by when video clips with different frame resolutions are decoded. In Section 3 of this paper, we present a framework that is used to compute the minimum processor frequency required to support each of the decoding levels in our scheme. The computed frequency is then used to estimate the energy consumption (or savings) associated with each level. This framework has similar goals to the work in [12]. However, this is not the main contribution of this paper. It only helps in estimating the effectiveness of our decoding scheme.

Our work has some similarities with the work in [15], where a perception-based partial encryption scheme for speech was presented. It is based on the observation that by encrypting only about 30 – 40% of a bitstream, sufficient content protection is achieved in wireless services. This leads to energy savings resulting from lower computational load on the processor. Although this basic idea is similar to what we also exploit in this paper, none of the results presented in [15] have any bearing with our work. Lastly, the concept of partitioning the frequency range of a codec based on perceptual relevance has also recently been exploited in [16] in the context of audio streaming over a network. Again, although we also rely on this basic idea of using perceptual relevance for prioritizing different frequency ranges, our work applies this concept to a very different problem, namely that of low-power decoding.

Organization of this paper: Although the basic scheme that we described above can be applied to most existing audio formats, for the remainder of this paper we will only consider the MPEG 1 Layer 3, also known as MP3, audio format. The experimental evaluations we have performed were also based on MP3. The main reason for us to choose MP3 is its widespread popularity. Additionally, compared to a single-layer codec like MP3, scalable codecs such as the MPEG 4 general audio usually incur a higher computational workload during the decoding process. This has been an additional reason for us to choose MP3.

The rest of the paper is organized as follows. In the next section we describe our perception-aware multi-level decoding scheme based on the MP3 codec. We call the resulting decoder the PL-MP3 (**P**erception-aware **L**ow-power **MP3**) decoder. In Section 3 we very briefly introduce an analytical framework to compute the minimum frequency at which the processor should be run at each of the decoding levels. Details of this framework may be found in an extended version of this paper [4]. The computed frequency is then used to estimate the energy savings associated with Levels 1-3. It may be noted that although this analytical framework is novel and also fairly involved, it is not the primary contribution of this paper. It is only used in conjunction with the proposed decoding scheme in order to accurately compute the minimum processor frequency and hence maximize the energy savings. However, it turns out that this analytical framework also provides insights into the dependency between the minimum required processor frequency and the playout delay (or initial buffering time). Using this framework, in Section 4 we also show that up to 20% energy savings may be obtained by increasing the playout delay from 0.5 sec to 2 sec, even when a clip is fully decoded (i.e. at Level 4). Hence, the total energy savings at any of the decoding levels is due to a combination of partially decoding the frequency bandwidth and the playout delay chosen. Further, these two options may be chosen independently of each other as illustrated in Figure 6 in Section 4.

Our experimental evaluation of the proposed decoder in presented in Section 4 using a processor model based on the SimpleScalar instruction set simulator [2]. Finally, we conclude in Section 5 by listing some directions for future work.

2 The PL-MP3 Decoder

In this section we first outline some of the motivations behind the design of the PL-MP3 decoder. This is followed by a description of this decoder and its differences with a standard MP3 software decoder.

2.1 Perception-Awareness in Audio Decoding

Perceptual characteristics of individual users: In general, the high frequency bands are perceptually less important than the low frequency bands [13, 16]. There is little perceptual degradation if we leave some high frequency components un-decoded. A standard MP3 decoder will simply decode everything in the bitstream without considering the hearing ability of individual users with or without hearing loss. This could result in a significant amount of irrelevant computation, thereby wasting battery power. The proposed PL-MP3 decoder overcomes this problem by integrating an individual user's own judgment on the desired audio quality.

Listening environment: It is relatively rare for a portable audio player to be used in a quite environment, for example in the living room of one's home. It is far more common to use portable audio players on the move and in a variety of environments such as in a bus, train, or in a flight, using simple earpieces. These differences have important implications on the audio quality required. The PL-MP3 decoder enables the user to change the decoding profile to adapt to the listening environment, while a standard MP3 decoder cannot.

Service types and signal characteristics: Different applications and signals require different bandwidth. For example, a storytelling audio clip requires significantly less bandwidth compared to a music clip. The PL-MP3 decoder allows the user to choose an appropriate decoding profile suitable for the particular service and signal type, in the process also prolonging the battery life.

To the best of our knowledge, the above observations have not been exploited in any commercially available audio players. The PL-MP3 audio decoder allows users to control the tradeoff between the battery life and the decoded audio quality, with the knowledge that slightly degraded audio quality (this degradation may not even be perceptible to the particular user) can significantly increase the battery life of the player.

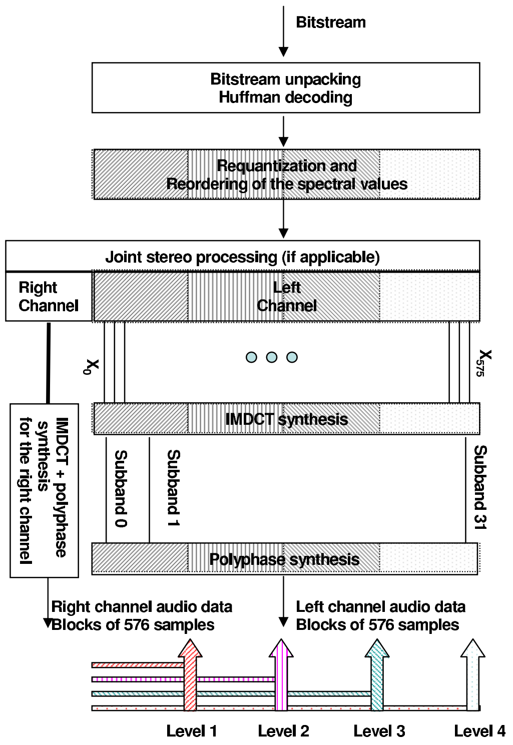

Figure 2: Block diagram of a standard MP3 decoder and the proposed PL-MP3 multi-level decoding scheme.

2.2 Decoding the MP3 Bitstream

Like many other multimedia bitstreams, the MP3 bitstream has also a frame structure. A frame contains a header, an optional CRC for error protection, a set of control bits coded as side information, followed by the main data consisting of two granules which are the basic coding units in MP3. For stereo audio, each granule contains data for two channels. This data consists of the scale factors and the Huffman coded spectral data. It is also possible to have some ancillary data inserted at the end of each frame. The decoder processes the MP3 bitstream frame by frame (or granule by granule, if we look inside a frame).

The block diagram of a standard MP3 decoder [9] is shown in Figure 2, along with our new PL-MP3 decoding scheme. A standard MP3 decoder parses the bitstream, decodes the side information first, and then runs several signal processing modules to convert the MP3 bitstream to pulse code modulation (PCM) audio samples. Three modules which incur the most computational workload are de-quantization, inverse modified discrete cosine transform (IMDCT) and polyphase synthesis filterbank. The standard decoder decodes the entire frequency band, which corresponds to the highest computational workload. In the case of PL-MP3, depending on the decoding level, the above three modules (i.e. de-quantization, IMDCT and polyphase synthesis filterbank) process only a partial frequency range and thereby incur less computational cost. Due to space restrictions, we omit the technical details here. But the basic intuition is as follows. On an abstract level, the decoder processes a potentially infinite stream of data items or granules. Due to the real-time constraints imposed by the rate at which the decoded granules are consumed by the output device, a limited amount of time can be spent in processing each granule (which determines the processor frequency). At lower levels of decoding, by reducing the computational cost associated with processing each granule, the processor can be run at a lower frequency, thereby saving power.

2.3 Workload Partitioning

Analyzing the MP3 decoding procedure reveals that after the bitstream unpacking and Huffman decoding module, which require only a small percentage of the total computational workload (4% in our examples), the workload associated with all the subsequent modules can be partitioned. In principle, it is possible to design a scalable decoding scheme with a granularity that corresponds to all the 32 subbands defined in the MPEG 1 audio standard [9]. However, for the sake of simplicity, we partitioned these 32 subbands into only four groups, where each group corresponds to a decoding level (see Figure 2 and Table 1).

As discussed in Section 1, the decoding Level 1 covers the lowest frequency bandwidth (5.5 kHz) which we define as the *base layer*. Although the base layer occupies only a quarter of the total bandwidth and contributes to roughly a quarter of the total computational workload, it is perceptually the most relevant frequency band. The output audio quality corresponding to this level is certainly sufficient for services like news and sports commentary. Level 2 covers a bandwidth of 11 kHz and almost reaches the FM radio quality, which is sufficiently good even for listening to music clips, especially in noisy environments. Level 3 covers a bandwidth of 16.5 kHz and produces an output that is very close to CD quality. Finally, Level 4 corresponds to the standard MP3 decoder, which decodes the full bandwidth of 22 kHz. Levels 1, 2 and 3 therefore process only a part of the data representing the different frequency components, whereas Level 4 processes all the data and is therefore computationally most expensive. According to our experiments, the audio quality corresponding to levels 3 and 4 are almost indistinguishable in noisy environments, but are associated with substantially different rates of energy consumption.

3 Processor Frequency Computation

In this section we very briefly outline how to compute the minimum operating frequency of a processor in order to run our decoding algorithm at any particular decoding level (details may be found in [4]). The computed frequency can then be used to estimate the power consumption due to the processor. Our system model is shown in Figure 3. The processor running the decoder application has an *internal buffer b*. The decoded audio stream, which is a sequence of PCM samples, is written into a *playout buffer B*. This playout buffer is read by the output audio device at some specified rate.

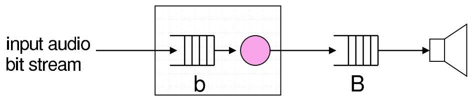

Figure 3: System model.

The input (constant rate) bitstream to be decoded is made up of a sequence of *granules*. The number of PCM samples per granule is a constant. Hence, for our analysis, we may consider that the playout buffer is read out by the output device at a constant rate expressed in terms of number of granules per second. The frequency at which the processor needs to be run is therefore determined by (i) the number of granules that need to be processed per second, in order to sustain the playout rate, and (ii) the number of processor cycles required to process each granule. There are two factors that make this frequency computation difficult. (i) The number of bits constituting a granule in the MP3 frame structure is variable. We have observed that the maximum number of bits per granule can almost be three times the minimum number of bits in a granule (where this minimum number is around 1200 bits). (ii) The number of processor cycles required to process each granule is also variable. Figure 4 shows the processor cycle requirement per granule, corresponding to a 160 kbits/sec bitrate audio clip, for a duration of around 30 secs. This figure shows the processor cycle requirement corresponding to the four decoding levels of our PL-MP3 decoder. In a nutshell, our analytical framework takes into account both these variabilities while computing the minimum processor frequency required to sustain each of the decoding levels. The same framework can also be used to compute the buffer size requirements for each of these levels.

4 Experimental Evaluation

We evaluated our decoder using two different classes of audio clips, those having a bitrate of 160 kbits/sec and the other class having a bitrate of 128 kbits/sec. In the former class, the average number of bits per granule is higher compared to the latter class. Our processor model (see Figure 3) was based on a Sim-Profile configuration of the SimpleScalar instruction set simulator [2]. Since we envisage that the proposed decoder will be run on a general purpose processor (such as those existing in a PDA, e.g. an Intel XScale 400MHz processor), we choose our processor to be a RISC processor (similar to a MIPS3000 processor) without any MP3 specific instructions.

We implemented the PL-MP3 decoder by modifying the original MP3 decoder source code available from the Fraun-

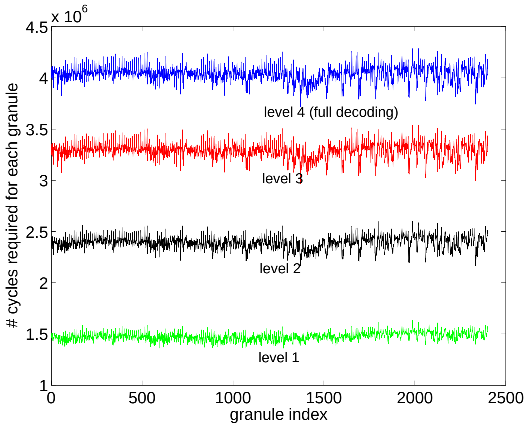

Figure 4: Variation in the processor cycle requirement per granule, for the different decoding levels.

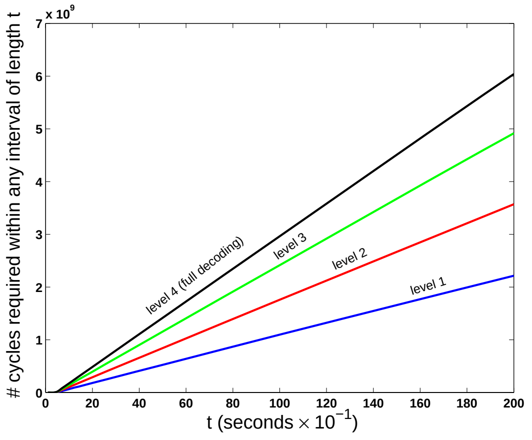

Figure 5: Cumulative processor cycle requirements versus time, corresponding to the different decoding levels for the high bitrate class of clips, with playback delay equal to 0.5 secs. Slope of these lines is equal to the processor frequency required.

hofer IIS website (also available from `www.mpeg.org`). All the audio clips we used had a sampling frequency of 44.1K PCM samples/sec per channel, which corresponds to CD quality audio. We experimented with three different playback delay values, equal to 0.5, 1 and 2 secs.

We simulated the decoding of several audio clips of duration 20 sec. From each such simulation, we collected five different traces: (i) the number of bits per granule, and (ii)-(v) the number of processor cycles required to process each granule for each of the four different decoding levels. From a collection of such traces, corresponding to a collection of audio clips, we computed the minimum processor frequency required for supporting each of the four decoding levels (see [4] for details). This procedure was implemented in Matlab (`www.mathworks.com`). Figure 5 shows the cumulative processor cycle requirements versus time. Hence, the slope of each of these lines represents the processor frequency corresponding to the different frequency levels. The minimum required frequency clearly increases as the decoding level is increased. Table 2 lists these

Playback delay	Level 4	Level 3	Level 2	Level 1
0.5 sec	302 MHz	246 MHz	178 MHz	111 MHz
1.0 sec	294 MHz	239 MHz	174 MHz	108 MHz
2.0 sec	279 MHz	227 MHz	165 MHz	102 MHz

Table 2: The minimum clock frequency at which the processor needs to be run for four different levels of decoding and for three different playback delays.

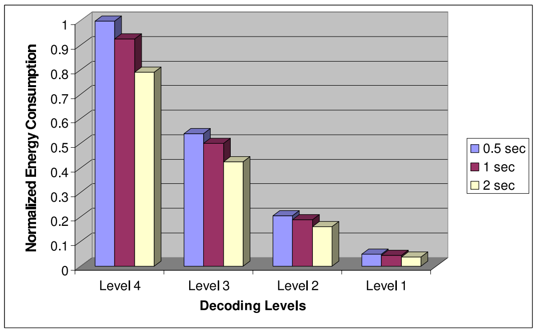

Figure 6: Normalized energy consumption for three different playback delays and the four decoding levels.

minimum required frequency values for the three different playback delay values, for the high bitrate class of clips. We obtained almost identical results for the low bitrate (128 kbits/sec) class of clips as well.

As the playback delay was increased from 0.5 sec to 1 and 2 secs, the minimum required frequency decreased for all the decoding levels. This is because the increased buffering time allows some of the burstiness in the workload (due to the variability in the number of processor cycles required to process the different granules) to be smoothened out. Figure 6 shows the normalized energy consumption for the high bitrate class of clips, for the four different decoding levels and the three different playback delay values. Clearly, the decoding level 4 along with a playback delay equal to 0.5 secs consumes the highest amount of energy. Compared to this base case, when an audio clip was decoded at level 1 with a playback delay of 2 secs, we computed an estimate of 95% savings in the energy consumed by the processor.

It may be noted that the results shown in Figure 6 do not take into account the static and dynamic power consumed by the buffer memory. It only shows the power consumed by the processor. The total buffer requirement, i.e. the size of the internal plus the playout buffer, increases as the playback delay is increased (see [4]). As a result, the effective energy savings resulting from increasing the playout delay would be slightly less than whose reported in Figure 6.

As mentioned before, the main result of this paper is the multi-level decoding scheme and an illustration of the potential energy savings that may be obtained from it. It is now fairly common knowledge that buffering can be used to smooth out the variability in multimedia workloads and this can be exploited to run a processor at a lower frequency, thereby saving energy. However, Figure 6 shows that the energy savings that can be achieved by using our multi-level decoding scheme is significantly larger than those that may be obtained using buffering alone. For example, using the standard MP3 decoder (i.e. Level 4), by increasing the playout delay from 0.5 to 2 secs, around 20% savings may be obtained. In contrast to this, more than 95% savings may be obtained by switching to Level 1 (albeit at the cost of a small sacrifice in the output quality) with a playout delay set to 0.5 sec. This difference will become even more pronounced if the power consumed by the buffers is taken into account (which we have neglected for the sake of simplicity). Clearly, the maximum savings may be obtained by using a combination of both.

Two sample MP3 audio clips that were decoded using the four decoding levels (Levels $1-4$) may be downloaded from [4]. The decoded PCM samples corresponding to each of the decoding levels were saved in `WAV` format and can be played using practically any media player, such as Winamp (`www.winamp.com`). These `wav` files will illustrate the output quality associated with each of the four decoding levels (see also Table 1).

5 Concluding Remarks

In this paper we presented a novel software audio decoder that allows the user to choose an output quality level. By making a small sacrifice in the output quality, it is possible to significantly enhance the battery life of a device running this decoder. Although the basic idea behind this decoder is intuitively simple, this observation is currently not exploited in any commercially available audio player. We are also not aware of any publication which systematically studied this option. We would like to point out that this scheme is conceptually similar to the “long-play” (LP) recording option available in some high-end DVD recorders. However, such a feature is implemented at the encoder side and the recorded data can only be played out at a fixed quality level. Our scheme is implemented at the decoder side and allows playout at multiple quality levels.

The scheme presented in this paper can be extended in several directions. In its current implementation, the original audio bitstream is not modified in any way. However, only a part of this bitstream is decoded and the rest is ignored. One possible improvement is to perform a fast compressed domain transcoding to remove parts of the bitstream that will not be decoded. The resulting bitstream will occupy less memory space. However, there will be less flexibility in controlling the output quality.

Currently our multi-level scheme is only based on partitioning the frequency bandwidth. In the case of multichannel encoding schemes it should also be possible to exploit the correlation between different channels to further reduce the computational workload involved in the decoding process. Additionally, it should also be possible to split the frequency bandwidth into more than four groups to allow a finer control on the output quality.

References

[1] A. Acquaviva, L. Benini, and B. Riccó. Software-controlled processor speed setting for low-power streaming multimedia. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(11), November 2001.

[2] T. Austin, E. Larson, and D. Ernst. SimpleScalar: An infrastructure for computer system modeling. *IEEE Computer*, 35(2):59–67, 2002.

[3] L. Cai and Y.-H. Lu. Dynamic power management using data buffers. In *DATE*, 2004.

[4] A perception-aware low-power software audio decoder for portable devices. http://www.comp.nus.edu.sg/˜samarjit/pl-mp3/.

[5] K. Choi, K. Dantu, W.-C. Cheng, and M. Pedram. Frame-based dynamic voltage and frequency scaling for a MPEG decoder. In *ICCAD*, 2002.

[6] K. Choi, R. Soma, and M. Pedram. Off-chip latency-driven dynamic voltage and frequency scaling for an mpeg decoding. In *DAC*, 2004.

[7] C. Huges, J. Srinivasan, and S. Adve. Saving energy with architectural and frequency adaptations for multimedia applications. In *34th Annual International Symposium on Microarchitecture (MICRO)*, 2001.

[8] C. Im, S. Ha, and H. Kim. Dynamic voltage scheduling with buffers for low-power multimedia applications. *ACM Transactions on Embedded Computing Systems*, 3(4):686–705, 2004.

[9] ISO/IEC 11172-3, Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, 1993.

[10] K. Keutzer, S. Malik, A. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli. System level design: Orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12), December 2000.

[11] Y.-H. Lu, L. Benini, and G. D. Micheli. Dynamic frequency scaling with buffer insertation for mixed workloads. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(11), November 2002.

[12] M. Mesarina and Y. Turner. Reduced energy decoding of mpeg streams. *Multimedia Systems*, 9(2):202–213, 2003.

[13] T. Mock. Music everywhere. *IEEE Spectrum*, Sep 2004.

[14] G. Qu and M. Potkonjak. Energy minimization with guaranteed quality of service. In *ISLPED*, 2000.

[15] A. Servetti and J. D. Martin. Perception-based partial encryption of compressed speech. *IEEE Transactions on Speech and Audio Processing*, 10(8), November 2002.

[16] Y. Wang, W. Huang, and J. Korhonen. A framework for robust and scalable audio streaming. In *ACM Multimedia*, 2004.

[17] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *19th ACM Symposium on Operating Systems Principles (SOSP)*, 2003.