# Power Reduction by Varying Sampling Rate

William R. Dieter
Electrical and Computer Engineering Department
University of Kentucky
Lexington, KY 40506–0046
dieter@engr.uky.edu

Srabosti Datta
Electrical and Computer Engineering Department
University of Kentucky
Lexington, KY 40506–0046
sdatt1@engr.uky.edu

Wong Key Kai
Electrical and Computer Engineering Department
University of Kentucky
Lexington, KY 40506–0046
kkwong0@uky.edu

## ABSTRACT

The rate at which a digital signal processing (DSP) system operates depends on the highest frequency component in the input signal. DSP applications must sample their inputs at a frequency at least twice the highest frequency in the input signal (i.e., the Nyquist rate) to accurately reproduce the signal. Typically a fixed sampling rate, guaranteed to always be high enough, is used. However, an input signal may have periods when the signal has little high frequency content as well as periods of silence. When the input signal has no perceptible high frequency components, the system can reduce its sampling rate, thereby reducing the number of samples processed per second, allowing the CPU speed to be scaled down without reducing output quality. This paper describes how to reduce power consumption in DSP applications by varying the amount of processing based on the input signal, and reports results of experiments with a prototype implementation. Experiments with a prototype show that when the system performs little processing, the added overhead of the variable sampling rate technique increased power consumption. When the system performs more processing, 18 FIR filters per frame, the power consumption was reduced to 40 % of the power required for a static sampling rate, while not reducing sound quality.

## Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems; C.3 [**Special-Purpose and Application-Based Systems**]: Signal processing systems; J.7 [**Computers in Other Systems**]: Real time

## General Terms

Performance, Experimentation

## Keywords

Power-aware, digital signal processing, frequency scaling, voltage scaling, real-time audio


*ISLPED'05,* August 8–10, 2005, San Diego, California, USA


## 1. INTRODUCTION

Digital signal processing (DSP) has found its way from high-performance applications like radar systems, affordable only by a few, to consumer-level applications like cell phones and power drills. Many of these new applications are portable battery-powered devices. However, the usefulness of these devices is limited by battery life, which depends on the device's power consumption.

Strategies for reducing power take advantage of opportunities at all levels of system design from transistors all the way up through the application software [15]. At the transistor level, the dynamic power consumed by a CMOS device is proportional to the clock frequency and the square of the supply voltage, i.e.:

$$P \propto V_{dd}^2 f,$$

where $V_{dd}$ is the supply voltage, and $f$ is the clock frequency [9]. Halving the clock frequency without changing $V_{dd}$ cuts power consumption in half, but does not affect the total energy consumed.

Energy is the time integral of power, so energy consumption is not reduced as long as the same number of clock cycles are required to do the same work. However, reducing clock frequency allows $V_{dd}$ to be reduced [3, 17] leading to energy savings. Though static power is an increasing fraction of total power, dynamic power is still the majority of power consumed. Thus, varying frequency and voltage can significantly reduce power and energy consumption.

The dynamic voltage (and frequency) scaling (DVS) algorithms proposed in the literature [2, 5, 6, 13, 14] exploit the relationship between power, voltage, and frequency to reduce energy consumption. Typical real-time DVS algorithms try to run the processor as slowly as possible while still guaranteeing that all jobs meet their deadlines. This strategy works well, especially when the average execution times of jobs is less than their worst case execution times. Though they can exploit slack produced by varying execution times, these algorithms do not directly change the number of clock cycles required by each job.

Many DSP applications have a fairly simple structure where a number of filters are periodically applied to a frame of samples. Most filters are a set of equations involving points in each input frame, with highly predictable execution times. DVS may seem less applicable because the amount of computation required for a given filter with a fixed buffer size does not vary more than a few cycles. However, the amount of work required of the system *does* vary based on the frequency content of the signal sampled in each frame.

In this work, we use a TI DSP to emulate a digital hearing aid to demonstrate the concept and measure results on real hardware.

A digital hearing aid provides an example of this concept, though any DSP application requiring significant processing power can use the same technique as long as there is variation in the frequency content of the input signal. For example, the audio frequency content in the typical office environment varies greatly. Sometimes the room is quiet. Conversation generates audio signals with primarily low frequency content. Other events like a telephone ringing, an alarm sounding, or music playing can add higher frequency content. A DSP-based hearing aid can exploit this fact by temporarily reducing its effective sampling rate during times when no significant high-frequency content is audible. Fewer input samples per second mean less computation for the filters to do on each frame, which leads to lower power consumption. Whenever a high frequency input is detected, the hearing aid can increase its sampling rate and processing speed to process the input at a rate high enough to preserve signal fidelity.

The hearing aid application takes advantage of human perception of sound to determine how much processing the signal needs. The psychoacoustic model and overall system design used in this application are described in Section 2. Section 3 describes the system implementation for this prototype, including hardware shortcomings we experienced. Measurements from running the system are given in Section 4. Section 5 discusses related work, and Section 6 concludes the paper.

# 2. SYSTEM MODEL

A hearing aid only needs to process those signals that a person with normal hearing would be able to hear. A person with normal hearing does not perceive every frequency equally well, for example. A hearing aid designed with the knowledge of this and other traits of the human auditory system can use less computationally intense filters than might otherwise be required. Like lossy audio compression algorithms, the system can introduce noise into the signal, as long as the noise does not affect the perceived quality of the output signal.

## 2.1 Absolute Threshold of Hearing

Though the frequency range of human hearing is generally considered to be 20 Hz to 20 KHz, not all frequencies are heard equally well. For example, tones at the extreme frequencies are more difficult to hear. Furthermore, the human ear can detect differences in pitch better at lower frequencies than at higher frequencies.

The *Absolute Threshold of Hearing* (ATH) indicates the quietest sound a person with normal hearing can perceive [12, 23]. The following equation has been found to approximate the absolute threshold of hearing, based on experiments playing a sinusoidal tone at a very low power:

$$ATH(f) = 3.64\left(\frac{f}{1000}\right)^{-0.8} - 6.5e^{\left(-0.6\frac{f}{1000}-3.3\right)^2} + 10^{-3}\left(\frac{f}{1000}\right)^4,$$

where f is the frequency of the input tone in hertz. Frequency components with power levels that fall below the ATH (measured in sound pressure level) can be discarded, as a listener with normal hearing will be unable to hear them. Moreover, the system can add noise to the output signal without reducing the signal's perceived quality, as long as the total power at each frequency stays below the ATH curve.

## 2.2 Hearing Aid Model

A simple digital hearing model aid consists of a microphone, A/D converter, DSP processor, D/A converter, and a speaker. The A/D converter converts analog signals from the microphone into digital samples, which are then processed by the DSP, converted to analog signals by the D/A converter, and sent to a speaker. According to sampling theory a signal must be sampled at a rate at least twice its bandwidth to be exactly reconstructed from its samples [19]. This sampling frequency is often called the *Nyquist sampling frequency* or the *Nyquist rate*. A form of distortion called *aliasing* occurs if the bandwidth of the sampled signal exceeds twice sampling frequency. In practice, systems use sampling rates slightly above the Nyquist rate. Since human hearing is generally considered to cover a range of 20 Hz to 20 KHz, typical audio sampling rates are 44.1 KHz or 48 KHz.

Samples are grouped into frames of approximately 20 ms worth of data, and filtered one frame at a time. At any given time the hearing aid is simultaneously transferring samples from the A/D converter to an input buffer, processing data from the previously sampled frame, and transferring samples from the previously processed buffer to the D/A converter. The 20 ms frame size is long enough to produce stationary signals, while not adding noticeable delay [21].

An ideal variable sampling rate hearing would sample the analog input signal at the minimum rate required to avoid aliasing for the current input frame. In other words, the sampling frequency would be lowered for frames when the signal has no audible high frequency components and raised when high frequency components are present. Processing would be carried out as in the standard hearing aid, with filter coefficients adjusted for the current sampling frequency. The execution time of the filters on a DSP depends on the number of samples and coefficients, but is independent of the filter coefficient values. At lower sampling rates, fewer samples are taken per 20 ms frame. Thus the system could slow down the CPU in proportion to the number of samples, so that the filter finishes computing the filtered frame as close to 20 ms as possible.

The most difficult theoretical question in building a variable sampling rate system is, "When should the A/D converter take the next sample?" Ideally, the system could adjust its frequency with each sample taken. However, determining the signal's frequency on a per sample basis would be difficult with existing hardware. For example, changing the A/D converter's sampling rate takes longer than the time between samples. When working with frames of samples, the frequency can be determined for each frame and be kept constant over an entire frame. However, there is no way to determine the frequency content of the end of the frame when the beginning of the frame is sampled.

The A/D converter in our prototype runs at a constant speed, sampling every frame at a rate, $f_{max}$, high enough to guarantee no aliasing. It then examines the entire frame to determine the highest audible frequency present, $f_a$. The

system selects the precomputed filter set with the lowest *effective sampling rate*, $f_i \geq 2f_a$, the minimum rate at which a frame can be sampled without aliasing, and resamples the frame for the selected filter set. The filter sampling frequencies, $f_i$, are chosen to be integer multiples of $f_{max}$, so resampling the frame is a matter of selecting every $k^{th}$ sample, where $k = f_{max}/f_i$.

To determine the maximum audible frequency, $f_a$, in a frame our prototype transforms the frame to the frequency domain with a Fast Fourier Transform (FFT). The FFT determines how strong the signal is at a discrete set of frequencies up to $f_{max}/2$. The prototype finds the highest frequency in the transformed signal with power greater than the ATH curve, described in Section 2.1. Hearing aids that do their filtering in the frequency domain already have to do an FFT, so the only added overhead would be comparing the FFT output to the ATH curve.

Often hearing aid filters are implemented in the time domain. If an FFT is too expensive, a set of time domain filters, with cutoff frequencies corresponding to the $f_i$'s, could be used to find $f_a$. Starting with the highest cutoff frequency, each high pass filter can be applied to the input buffer. The energy of the signal above the given frequency can be estimated by summing the square of the filtered sample values. The signal contains significant high frequency signals if the energy of the filtered signal is above a threshold determined by the ATH.

The prototype implementation described in this paper, computes $f_a$ using an FFT. If the amount of computation required to determine $f_a$ is greater than the amount of power saved due to frequency and voltage scaling, then scaling frequency and voltage is not worthwhile. As will be seen in Section 4, the payoff varies with the amount of computation required by the hearing aid filters.

## 3. IMPLEMENTATION

Measurements for the variable sampling rate hearing aid are based on a TI TMS320C5510 DSP Starter Kit. The TI TMS320C5510 is typical of DSP's used in digital hearing aids. The starter kit is based on a printed circuit board with a TMS320C5510 DSP processor and a stereo codec (a combined A/D and D/A converter) capable of sampling audio at rates from 32 KHz to 96 KHz. The DSP core can be scaled to many different frequencies from 6 MHz to 200 MHz. It can run at 1.1 V at frequencies less than or equal to 72 MHz. Otherwise it must run at 1.6 V [4, 20].

To get maximum benefit from frequency scaling, the system should support a different voltage for each frequency. Though our prototype only has two voltage choices, discarding unneeded samples reduces total energy consumed because the computational complexity of the filters is directly proportional to the number of samples per frame. Moreover, battery capacity depends non-linearly on the discharge rate. In lithium ion batteries, for example, not only does battery capacity decrease at high discharge rates, it also becomes more sensitive to peak power consumed than to average power [10]. Even if the voltage cannot be reduced, reducing the clock frequency reduces the peak power within each frame, compared with running the clock full speed and putting the CPU to sleep until the next frame.

The TMS320C5510 DSP Starter Kit has several problems related to frequency scaling. The TMS320C5510 uses an internal phase-locked loop (PLL), controlled by CPU registers, to generate the CPU clock frequency from a 24 MHz crystal oscillator. Calculations based on specifications from TI and experiments with frequency scaling have shown that locking to a new frequency can take from 20 $\mu s$ to 80 $\mu s$. When the frequency changes, the function units within the CPU cannot communicate with each other until the PLL locks to the new frequency.

One of the affected function units is the direct memory access (DMA) controller that transfers samples to and from the codec. In the constant frequency implementation, the DMA controller moves an entire frame in the background while the DSP filters the current frame. The variable sampling rate implementation attempts to do the same thing, and thus does not run reliably at all the data points. If the function units could continue to run while the frequency changes, or the frequency change could complete between two samples, the power consumption would be the same as is reported here.

The time required for the PLL to lock to a new frequency is often longer than the time between samples, depending on the frequency to which it is changing. With a sampling rate of 48 KHz the time between samples is approximately 20.83 $\mu s$. Slowing the sampling rate to 32 KHz, the minimum sampling frequency supported by the codec, would increase the intersample time to 31.25 $\mu s$, which is still not long enough to guarantee that a frequency change transition will complete before the next sample is ready.

One way around this problem would be to use two PLL's connected to a CPU controlled multiplexer. While the CPU uses one PLL as it's clock, the other PLL can change to an arbitrary frequency. When the second PLL has stabilized, the CPU can switch to the second PLL quickly. IBM PowerPC 750FX RISC Microprocessor offers this feature. It can switch between two PLL's in three clock cycles with all function units continue to operate normally during the switch [7]. Unfortunately, the PowerPC 750FX would not be suitable for this application because draws at least an order of magnitude more power than the TMS320C5510, though it is a much faster processor.

Another alternative is to use a frequency divider to generate lower frequencies from a constant high frequency. Since no phase locking is required the frequency divider can quickly switch frequencies.

The experimental setup consists of a PC sound card to supply repeatable inputs to the "line in" on the starter kit. The starter kit provides a jumper to which a wire loop is connected to measure current and supply voltage of the CPU core. A Tektronix current probe and AM503B current probe amplifier connected to a Tektronix TDS 3012B digital oscilloscope measured current through the loop, while the second channel on the oscilloscope measured the core voltage. RMS power was computed using the oscilloscope's multiply and RMS functions. The RMS window was set wide enough to cover several frames. Multiple power consumption measurements were taken for each tested configuration and averaged. Energy consumption can be estimated by multiplying the average power measurement by the desired amount of time.

A CD recording of music provided a repeatable audio input source. The music test [18] represents a difficult case, because it frequently contains audible high frequency components not present in speech only signals.

Each of the data points was measured with a sampling

rate of $f_{max}$ = 48 KHz with a frame size of 1024, giving 21.3 ms per frame. Downsampling was implemented by dividing $f_{max}$ by a power of two. Thus possible effective sampling rates are 48 KHz, 24 KHz, 12 KHz, 6 KHz, 3 KHz, 1.5 KHz, 750 Hz. Lower frequencies were not used because the filters used do not operate on fewer than 16 samples.

Digital hearing aids often use filters like frequency shaping, adaptive noise reduction, interaural time delay, and multichannel amplitude compression to process their input signals [21]. To give a generic characterization of the technique as a function of processing load, we used a series of finite impulse response (FIR) filters applied to the input buffer repeatedly as a dummy processing load. In practice, each channel (left and right) uses between 1 and 50 FIR filters with 50 to 200 taps for frequency shaping, with 14 filters being common [21]. Similar filtering techniques are used for the other types of processing.

The FIR filter [22] is hand-tuned in assembly language for maximum execution speed. It is designed to use the TMS320C5510 pipeline efficiently including the the dual multipliers, which give the load a power consumption characteristic similar to highly optimized signal processing code.

## 4. RESULTS

Overhead and load execution time were measured using TI Code Composer Studio's profiling clock. Clock cycle counts stayed constant as the CPU frequency changed because all code and data are located in on-chip memory. The on-chip memory can be accessed in a single cycle using the same clock as the rest of the CPU. The FFT used to determine the maximum frequency content in the input signal was taken from the TI TMS320C55x DSPlib library [22]. It was measured to take less than 64,000 cycles on the TMS320C5510. Postprocessing to upsample the output waveform was measured to take less than 57,000 cycles. The FIR filter used to load the processor was also taken from the TI TMS320C55x DSPlib library [22]. It consistently took about 110,000 cycles to execute for 208 taps and 1024 samples.

Figure 1 compares the power consumed by the standard hearing aid model to the variable sampling rate hearing aid as a function of the number of FIR filters run on each frame. The standard hearing aid was run at the smallest constant frequency required to guarantee the FIR filters had time to run during each frame. The variable sampling rate hearing aid scaled the frequency for each frame to the minimum required to run all the FIR filters on the possibly reduced size frame. A third curve represents a standard hearing aid sampling at 24 KHz, half the maximum sampling rate of the other two hearing aids.

At low loads the standard hearing aid uses less power than the variable rate hearing aid because both the constant rate and the variable rate hearing aid can run at a frequency low enough that $V_{dd}$ is always 1.1 V. The savings from scaling are small in this case because there is not a lower voltage for the variable rate hearing aid to use. Around 14 FIR filters, constant rate hearing aid consumed approximately 2.5 times more power than the variable rate hearing aid because the constant rate hearing aid is forced to run at a frequency that requires 1.6 V. Most frames contain little enough high frequency content that the variable sampling rate hearing aid can run at 1.1 V most of the time.

All of the curves show an increase in power consumption when the processor has to shift from running at 1.1 V to

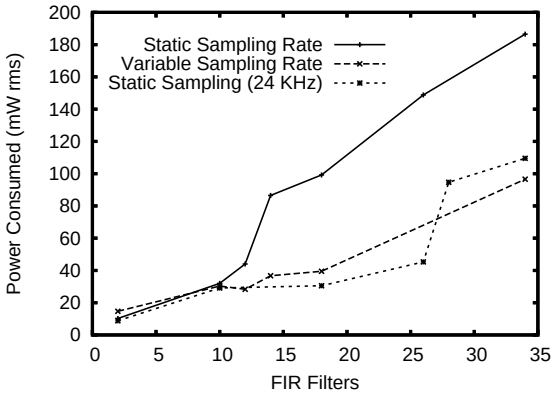


**Figure 1: Graph of system power as a function of load for the standard hearing aid model and the variable sampling rate hearing aid.**

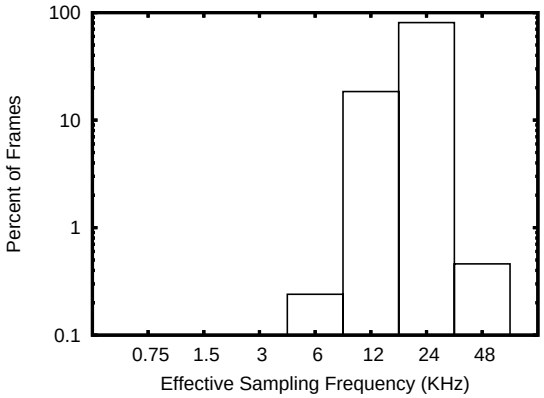


**Figure 2: Measured distribution of effective sampling rates for music test.**

1.6 V. For a given load, the fixed sampling rate hearing aid only needs one voltage level, just high enough to run at a speed fast enough to complete processing just in time for the next frame. The variable sampling rate hearing aid, however, would benefit from more voltage levels. Ideally a different voltage level would be available for frame size to maximize the power savings of switching frame sizes.

Though the standard hearing aid running at 24 KHz uses less power than the variable sampling rate hearing aid for many cases, it loses audible signals above 12 KHz. The histogram in Figure 2 shows, in log-log scale, how often the variable rate hearing aid is able to use each of the sampling rates from 750 Hz to 48 KHz. The 24 KHz and 12 KHz effective sampling rates are most commonly used. Both lower and higher sampling rates are also present, but represent fewer than 1 % of the frames.

The dotted curve in Figure 1 shows that reducing the fixed sampling rate hearing aid's sampling frequency to 24 KHz would compete well with the variable sampling rate hearing aid. This tradeoff may be worthwhile when less than one percent of the samples need to be sampled at a rate higher than 24 KHz. Even so, the variable rate hearing aid uses less power for high loads. The variable sampling rate system would likely also do better if more voltage levels were

available, since the largest advantage is seen when it is able to switch to a lower voltage than the static sampling rate system much of the time. Optimization of the up and down sampling code would also improve the variable rate hearing aid's performance.

## 5. RELATED WORK

Most DVS algorithms [2, 5, 6, 13, 14] estimate the slack available at scheduling points and adjust the clock frequency to keep the CPU busy at as low a clock frequency as possible. Rather than only updating slack at scheduling points, AbouGhazaleh, Childers, et. al [1] introduce compiler support to insert power management points into a program. At each power management point the compiler provides updated WCET information, which is periodically used to update the current clock frequency. This method is different than typical DVS scheduling algorithms in that the power management points allow the system to get information about the amount of dynamic slack available from a job before the job finishes.

Multimedia applications often take advantage of limitations of human perception. For prerecorded video streams, buffering extra frames before processing them has been shown to increase slack time in the system when the amount of compressed video data per frame varies widely [8]. Buffering several frames lets the scheduler spread longer jobs over several periods, taking slack from periods with lower execution time requirements. Buffering could be applied in addition to varying the sampling rate, but the added delay may become noticeable in the hearing-aid application.

All of these algorithms use static and dynamic slack in the schedule to reduce energy consumption. In contrast, our method introduces slack into the schedule by changing the amount of work the application requires based on the current input. Other DVS techniques could be combined with ours to take advantage of this slack. For example, compiler inserted power management points would remove the need for the programmer to manually determine the clock frequency for each filter speed. Other DVS algorithms could schedule variable sampling rate signal processing tasks with other tasks in a system with other real-time tasks.

In cases where schedulability cannot be guaranteed, previous work has studied how to maximize system value while meeting energy and timing constraints for jobs that complete [16]. Frames with little high frequency content could be viewed as having lower value, thus getting less computation. Even though some samples are discarded, the difference in signal output should be indistinguishable from a frame in which all samples were processed, so the discarded work had zero value. Moreover, all frames are guaranteed to be processed by their deadlines, regardless of the clock frequency. In other words, the techniques described in previous work trade quality for energy consumption. This work reduces energy consumption without reducing perceived quality.

Lossy audio compression algorithms [12, 23] commonly use psychoacoustic principles to reduce the amount of data in a stored or transmitted audio stream. Our approach is similar in that we improve the economy of the system by imperceptibly changing the signal. It is different in that we reduce the amount of computation required to process the signal, rather than reducing the amount of space required to store it. Compression algorithms ideally remove the maximum amount of information, which can take a significant amount of computation. In contrast, our method seeks to discard inaudible high frequency signal components with minimal extra computation.

## 6. CONCLUSIONS AND FUTURE WORK

The variable sampling rate technique reduces power required to as little as 40 % of the power required for the static sampling rate, depending on the amount of processing per frame. As the amount of computation per frame decreases the technique becomes less effective due to the overhead of determining the frequency content of a frame. Optimizations of the current prototype should increase the range for which this technique is useful. A wider range of voltages and less computationally complex method for determining the sampling frequency would increase the effectiveness of the technique.

Faster frequency switching or a codec that can buffer several samples is needed to avoid missing samples. Many DVS algorithms proposed by others only change frequency at longer intervals. In those cases the 20 $\mu s$ to 80 $\mu s$ to change frequencies is reasonable. Even for the hearing aid application the frequency switching time is only a small percentage of the frame size. Buffers in the codec to hold samples while the CPU is unavailable or a second PLL would prevent dropped samples.

We are currently developing a set of time domain bandpass filters, which we expect to require much less computation than the FFT/ATH combination. Low power switched capacitor filter banks have been used to measure frequency response in several bands [11]. Adding such circuitry to the A/D converter could dramatically increase the efficiency of determining the highest frequency components in the system. If the processing stage operates in the frequency domain, then a filter bank is unnecessary because the FFT is essentially free.

Digital hearing aids are just one energy sensitive DSP application. The variable sampling rate technique can be applied to any signal processing application with significant variation in input frequencies for which the CPU power is a substantial portion of the entire system power.

## 7. ACKNOWLEDGMENTS

The authors would like to thank Kevin Donohue and Art Radun for use of lab space and test equipment used in measuring results. Mike Strain at Digital Spectrum and Todd Sanning at TI were of assistance in troubleshooting our starter kit.

## 8. REFERENCES

[1] N. AbouGhazaleh, B. Childers, D. Mossé, R. Melhem, and M. Craven. Energy management for real-time embedded applications with compiler support. In *ACM SIGPLAN Joint Conference LCTES'03*, June 2003.

[2] H. Aydin, R. Melhem, D. Mossé, and P. M. Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *IEEE Real-Time System Symposium*, December 2001.

[3] K. A. Bowman, B. L. Austin, J. C. Eble, X. Tang, and J. D. Meindl. A physical alpha-power law MOSFET model. *IEEE Journal of Solid-State Circuits*, 34(10):1410–1414, October 1999.

[4] R. Cyran. Using the power scaling library on the TMS320C5510. Technical Report SPRA848, Texas Instruments, P.O. Box 655303 Dallas, Texas 75265, October 2002.

[5] A. Dudani, F. Mueller, and Y. Zhu. Energy-conserving feedback EDF scheduling for embedded systems with real-time constarints. In *ACM SIGPLAN Joint Conference LCTES'02*, June 2002.

[6] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and DVS processors. In *Proceedings of the 2001 international symposium on Low power electronics and design*, pages 46–51. ACM Press, 2001.

[7] IBM. *IBM PowerPC 750FX RISC Microprocessor User's Manual, Version 1.01*, February 2003.

[8] C. Im and S. Ha. Dynamic voltage scaling for real-time multi-task scheduling using buffers. In *Proceedings of the 2004 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools*, pages 88–94. ACM Press, 2004.

[9] S.-M. Kang and Y. Leblebici. *CMOS Digital Integrated Circuits Analysis & Design*. McGraw-Hill Science/Engineering/Math, third edition, October 2002.

[10] T. L. Martin and D. P. Siewiorek. Non-ideal battery properties and low power operation in wearable computing. In *International Symposium on Wearable Computers*, pages 101–106, October 1999.

[11] H. McDermott. A programmable sound processor for advanced hearing aid research. *IEEE Transactions on Rehabilitation Engineering*, 6(1):53–59, March 1998.

[12] T. Painter and A. Spanias. A review of algorithms for perceptual coding of digital audio signals. Technical report, Arizona State University, http://www.eas.asu.edu/ speech/ndtc/dsp97.ps, 1997.

[13] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the Symposium on Operating Systems Principles*, October 2001.

[14] G. Quan and X. S. Hu. Minimal energy fixed-priority scheduling for variable voltage processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1062–1071, August 2003.

[15] J. Rabaey and M. Pedram. *Low Power Design Methodologies*. Kluwer Academic Press, Norwell,Mass, 1995.

[16] C. Rusu, R. Melhem, and D. Mossé. Maximizing the system value while satisfying time and energy constraints. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 1–10. IEEE, 2002.

[17] T. Sakurai and A. R. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, April 1990.

[18] C. Santana. The Best of Santana. CD Recording, Sony Music Entertainment, 1998.

[19] C. E. Shannon. Communication in the presence of noise. *Proceedings of the Institute of Radio Engineers*, 37(1):10–21, January 1949.

[20] Spectrum Digital, 12502 Exchange Drive, Suite 440, Staffor, TX 77477. *TMS320VC5510 DSK Technical reference*, October 2002.

[21] T. Stetzler, N. Magotra, P. Gelabert, P. Kasthuri, and S. Bangalore. Low-power real-time programmable dsp development platform for digital hearing aids. Technical Report SPRA657, Texas Instruments, P.O. Box 655303 Dallas, Texas 75265, April 2000.

[22] Texas Instruments, P.O. Box 655303 Dallas, Texas 75265. *TMS320C55x DSP Library Programmer's Reference*, November 2003.

[23] K. Tsutsui, H. Suzuki, O. Shimoyoshi, M. Sonohara, K. Akagiri, and R. M. Heddle. ATRAC: Adaptive transform acoustic coding for minidisc. In *93rd Audio Engineering Society Convention in San Fransisco*, October 1992.