Implementation of Hearing Aid Signal Processing Algorithms on the TI DHP-100 Platform

Roger D. Chamberlain
Julius L. Goldstein
Darko Ivanovich

Roger D. Chamberlain, Julius L. Goldstein, and Darko Ivanovich, "Implementation of Hearing Aid Signal Processing Algorithms on the TI DHP-100 Platform," in *Proc. of 37th Asilomar Conf. on Signals, Systems and Computers,* November 2003.

BECS Technology, Inc.

and

Hearing Emulations, LLC

Implementation of Hearing Aid Signal Processing Algorithms on the TI DHP-100 Platform

Roger D. Chamberlain, Julius L. Goldstein, and Darko Ivanovich
roger@becs.com, jlg@hearem.com, darko@becs.com
BECS Technology, Inc., and Hearing Emulations, LLC, St. Louis, Missouri

***Abstract*-While the traditional deployment platform for digital hearing aids has been custom ICs, recent advances in the power efficiency of digital signal processors (DSPs) have generated interest in their use. A primary motivation for this interest is the increased algorithmic flexibility that a DSP provides over a fixed-function IC. The 'C54xx family of DSPs from Texas Instruments is among the leaders in power efficiency, and TI provides a development platform (the DHP-100) aimed at exploring the hearing aid design space. Here, we report on our experience porting hearing aid signal processing algorithms to the DHP-100 and report on the resulting performance, both for real-time response and power consumption.**

I. Introduction

Hearing aids are a demanding application for embedded computing, with both tight real-time requirements as well as stringent power constraints. Traditionally most digital hearing aids have been constructed using Application Specific Integrated Circuit (ASIC) technology, a technique that provides excellent utilization of available resources, but results in systems with fairly rigid and unalterable functional capabilities. There has been significant recent interest in the use of off-the-shelf programmable Digital Signal Processors (DSPs) for both testing candidate signal processing algorithms for hearing aids as well as deploying production aids. The primary advantage that DSPs provide is a dramatic increase in design flexibility, since the application is deployed in software rather than in dedicated hardware.

Here, we report on our development experience deploying a set of novel non-linear signal processing algorithms for hearing aids [1] onto the Texas Instruments DHP-100 platform [2,3]. The DHP-100 is a battery-powered, portable system that contains a low-power, fixed-point digital signal processor, stereo codec, and associated analog support circuitry suitable for audio signal processing applications. We describe the design experience as well as timing performance and power consumption for the resulting design.

The paper outline is as follows. Section II describes the hearing aid application itself, focusing on the required signal processing functions. Section III discusses the DHP-100 development platform from Texas Instruments. Section IV provides a description of the software development effort and the timing studies, both predicted and measured execution runtimes. Section V explores the power consumption and battery lifetime of the system, and Section VI concludes.

This research is supported by NIH/NIDCD SBIR Grant DC04028 to BECS Technology, Inc., in partnership with Hearing Emulations, LLC, J.L. Goldstein, PI.

II. Hearing Aid Application Description

Linear amplification with slowly adapting automatic gain control currently dominates advanced hearing aid design, and has been extensively studied [4]. The normal cochlea, however, uses non-linear, rapidly compressive amplification under efferent control [5,6], whose salient characteristics have been modeled [7] and are being studied systematically for use in multi-channel hearing aids [8,9,10]. Essentially, we are attempting to mimic the important aspects of a healthy ear to guide the design of future hearing aids. A block diagram of the required signal processing is shown in Fig. 1 [1].

The input signal comes in on the upper left side of the figure, is sampled at a rate of 16 kS/s, and is delivered to an all-pass filter in channel 5 (for equalization of the group delay across the channels), to a low-pass filter and down-sampler (so that signals in channels 4 through 1 are processed at a rate of 8 kS/s, lower for lower frequency channels), and to the adaptive compression threshold (ACT) unit described below. The band-pass filters separate the audio signal into octave channels. The figure shows a 5-channel system with an overall frequency range from 125 Hz to 4 kHz. Signal transduction in each channel is linear at low sound pressure levels (SPL) and compressive above an adaptive threshold. The transduction is shown via a non-linear amplifier in each channel. The output of the transducer is again band-pass filtered to remove undesired higher-order harmonics introduced by the non-linear amplification. The output of the second band-pass filter in each channel is first added to the output from any lower frequency channels, up-sampled, and low-pass filtered.

In the prototype implementation, the band-pass filters are 21-tap FIR filters designed by windowing and sampling IIR Butterworth band-pass impulse responses and are identical for all channels. The all-pass equalization filters are simply circular delay buffers. The low-pass filters are 21-tap FIR filters with a normalized cutoff frequency of 0.3π, also identical for all channels.

The implementation of the non-linear amplification is a piecewise linear function (on a log-log plot) illustrated in Fig. 2. At low signal levels (below the compression threshold) the transducer has a linear response with a high gain. Higher-level signals experience power-law compression.

The adaptive compression threshold (ACT) function is responsible for setting the instantaneous value of the compression threshold based on the RMS signal level. Techniques for implementing this control function are described in [1] and [10].

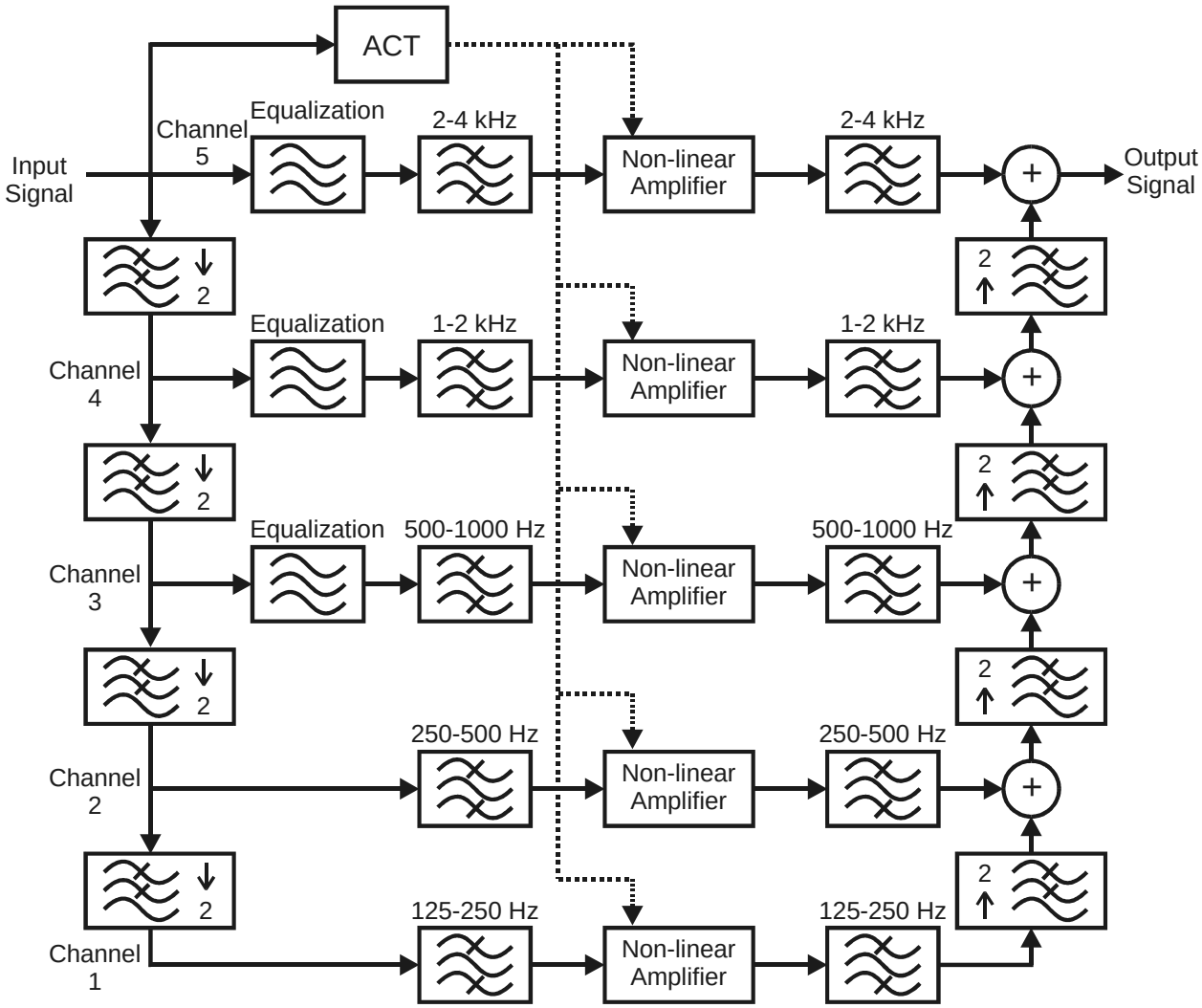

Fig. 1. Block diagram of multi-channel hearing aid signal processing [1].

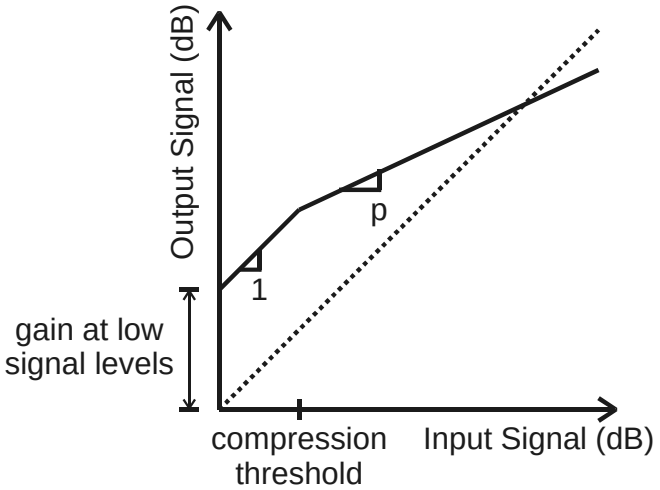

Fig. 2. Non-linear amplifier function.

III. The Texas Instruments DHP-100 Platform

Fig. 3 shows the salient features of the DHP-100 experimental platform. The DSP is a TI TMS320VC5402 fixed-point processor with a 16-bit word size that executes at 96 MHz. The stereo A/D and D/A functions are supported via a TI AIC-23 low-power codec. The input preamps and output audio amplifiers each have adjustable gain characteristics. In addition to that shown in the figure, the DHP-100 also contains a flash RAM (for retaining code) and power conditioning circuitry that enables it to execute from a 5 VDC power source or a pair of AA batteries.

The integrated development environment provided by TI is Code Composer Studio, currently version 2.2. This set of tools provides project management support, C/C++ compiler, assembler, linker, debugger, and profiler.

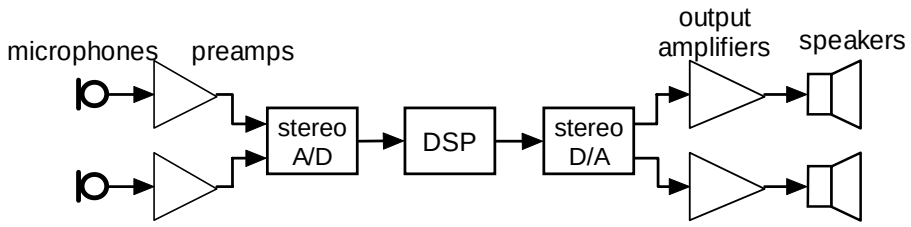

Fig. 3. Texas Instruments DHP-100 development platform..

IV. Software Development and Timing

The software development can be roughly decomposed into three pieces: input/output, filters, and non-linear amplifiers. The input/output functions deal with the interaction between the codec and the DSP, implementing the interrupt service routines, I/O buffers, etc. that provide the interface between the signal processing diagram of Fig. 1 and the real world. Subject to minor perturbations to adjust sample rate and the like, the majority of this code was provided by TI. In a similar manner, FIR filter routines are traditionally supplied (in optimized assembly) by the vendor of DSP systems, and the DHP-100 is no exception. A significant amount of effort, however, was necessary to implement the non-linear amplifiers.

The primary difficulty encountered in implementing the non-linear amplifiers was the inherent mismatch between the native numerical representation supported by the DSP chip and the numerical data types supported in the high-level

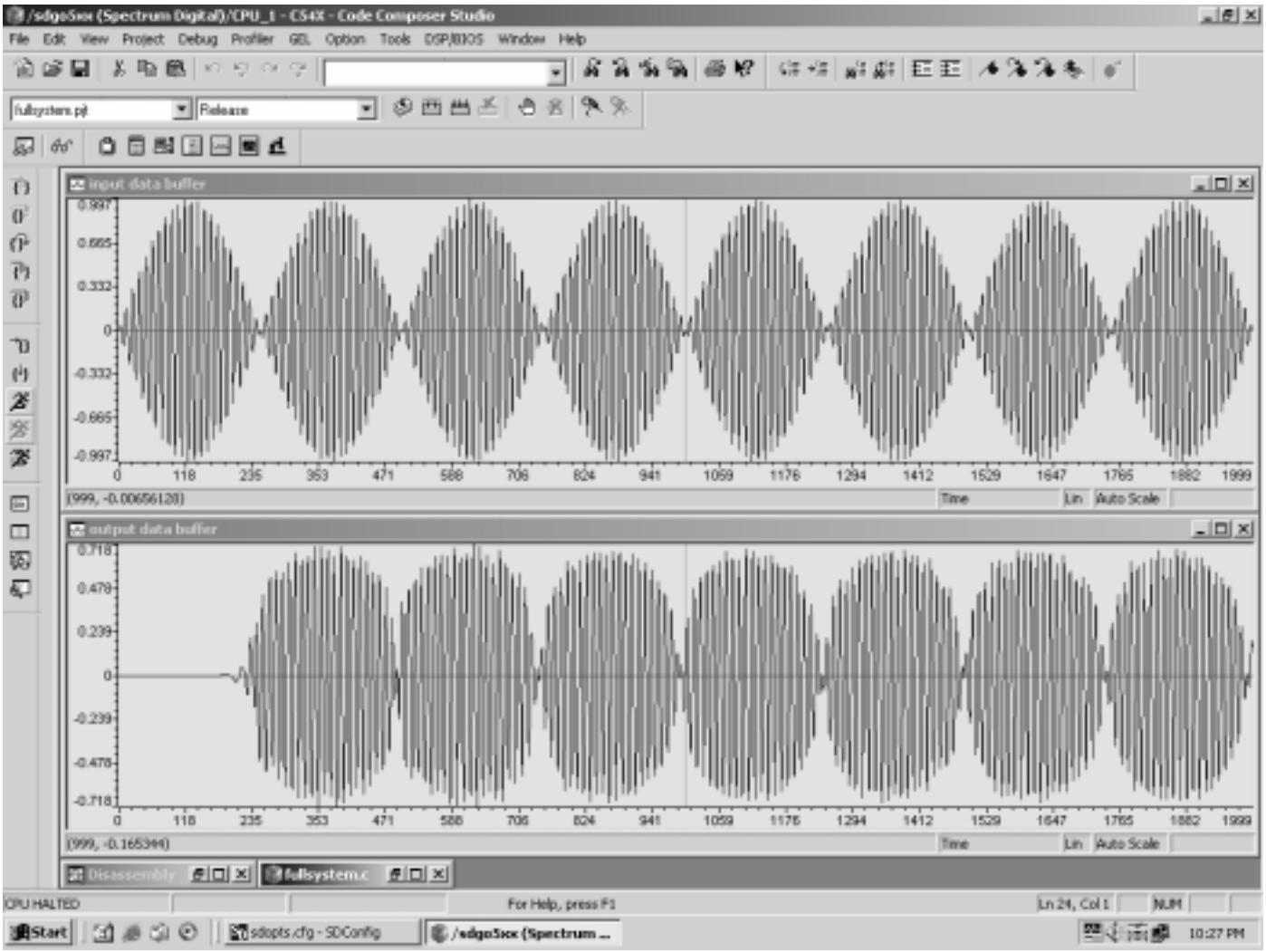

Fig 4. Input and output waveforms as displayed by the integrated development environment.

language (C/C++). While the DSP's native representation is a Q0.15 format (implying 15 bit positions to the right of the radix point), the C language has no reasonable mechanisms to support this format. As a result, variables that store signals to be interpreted in Q0.15 format are declared as integers in C, and it is the responsibility of the programmer to ensure that mathematical operations are correct.

We will illustrate the above point through the example of the non-linear amplifier function, shown below.

$$y = \begin{cases} G_1 x, & x \le x_T \\ G_2 x^p, & x > x_T \end{cases}$$

Here, x is the input signal value, G_1 is the gain at low signal amplitudes (below the threshold x_T), and p is the compression power ($0.25 \le p \le 0.5$). Although only shown for positive x, the function is odd-symmetric about zero (i.e., sign preserving).

The input x is stored in Q0.15 format, but since G_1 is greater than 1, it cannot be stored as Q0.15. In our implementation, G_1 is declared as a 32-bit integer and stored in Q15.16 format. As a result, the product of x and G_1 must be shifted 16 bits to the right if y is to be stored in Q0.15 format. The C code for this is as follows.

```
y = (G1 * x)>>16;
```

Above threshold, it gets significantly more complicated. We exploit the relationship $x^p = e^{p \ln x}$, and use vendor-supplied routines for ln x and e^x. However, the ranges of each intermediate value must be carefully assessed to ensure that numerical errors do not result.

Example input and output waveforms are illustrated in Fig. 4. The figure represents one of the useful characteristics of the integrated development environment provided by TI, the ability to observe waveforms in the memory space of the DSP. In this particular view, we have asked the system to plot two regions of memory, label the x axis via sample count, and interpret the values in Q0.15 format. The input waveform is a modulated sine wave, ranging over the entire valid scope of the number space (–1 to 1) and the output waveform reflects the compressive nature of the non-linear amplification.

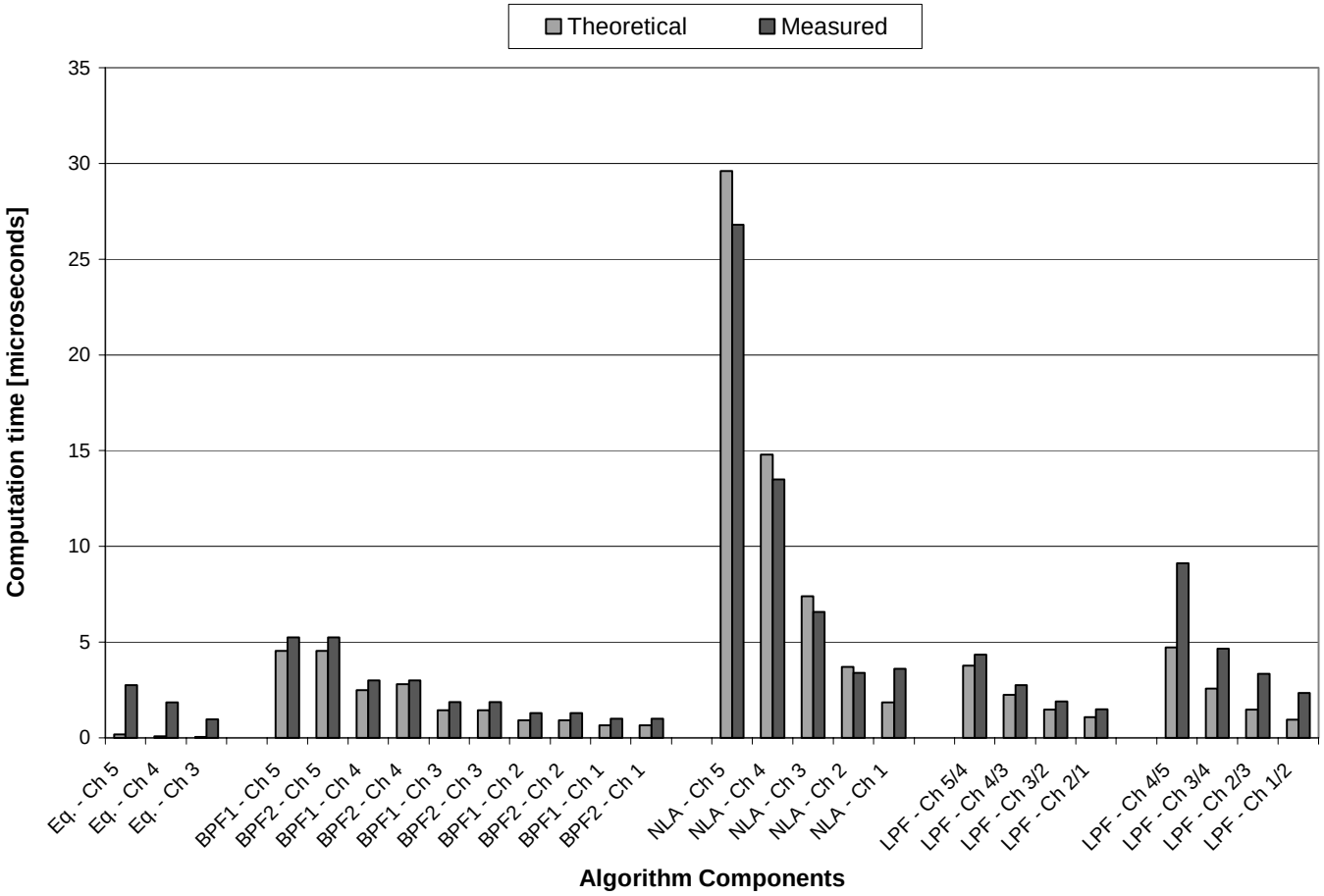

Fig. 5. Execution time for hearing aid algorithm components.

To assess the timing of the signal processing computations, we compare theoretical and measured timing for a 16-sample block of audio input (initially monaural, extended below to stereo). Blocks of 16 samples (1 ms window) are accumulated in an input buffer (managed by the input interrupt service routine), processed as described in Fig. 1, and then delivered from an output buffer to the codec. The blocking adds only 2 ms group delay to the overall system.

Examining the required processing in Fig. 1, and making several simplifying assumptions, the theoretical (predicted) computation time required (using the expressions in [11]) and the measured computation time required (from the DHP-100) for each of the components are shown in Fig. 5.

Generally, the correspondence between predicted and measured timing results is quite good. For the band-pass filters and down-sampling low-pass filters, the measured execution times are slightly larger than the predicted times. The opposite is true for the non-linear amplifiers, where the measured execution times are generally slightly lower than the predicted times. The greatest discrepancies occur for the equalization functions and the up-sampling low-pass filters. In the case of the equalization functions, the time is dominated by C loop overheads rather than the traditional numerical computation considered in the theoretical analysis, so it is not surprising that the analysis is in error here, and in the case of the up-sampling low-pass filters, the deployed application does not take advantage of the fact that every other sample is zero, while the theoretical analysis assumes that optimization is exploited. The resulting 2:1 discrepancy between predicted and measured execution times is therefore to be expected.

Given that the above execution times are for a 16 sample block, the hard real-time computation requirement is that the processing be complete in less than 1 ms. The predicted execution time for a complete block is 192 μs, while the measured execution time is 236 μs, for an overall relative error of only 18%. Given the uncertainties present in the analysis, this is a fairly significant alignment between predicted and measured execution times.

V. Power Measurements

The instantaneous power consumption of the DHP-100 is potentially a function of the computational complexity of the algorithm executing on the DSP chip and the output signal levels driving the speakers. In our experience, the former had negligible impact on the power consumption, while the latter's impact was significant. When powered via a 5 VDC source, the input current was monitored with different output signal levels and varying loads. These results are summarized in Table II.

TABLE II
Power Consumption under Varying Load Conditions

	POWER CONSUMPTION		
Input Signal	Low load (R_L = 1 MΩ)	Mid load (R_L = 100 Ω)	High load (R_L = 10 Ω)
50 mVp-p sinusoid	300 mW	325 mW	370 mW
100 mVp-p sinusoid	300 mW	360 mW	420 mW
speech	300 mW	330 mW	370 mW

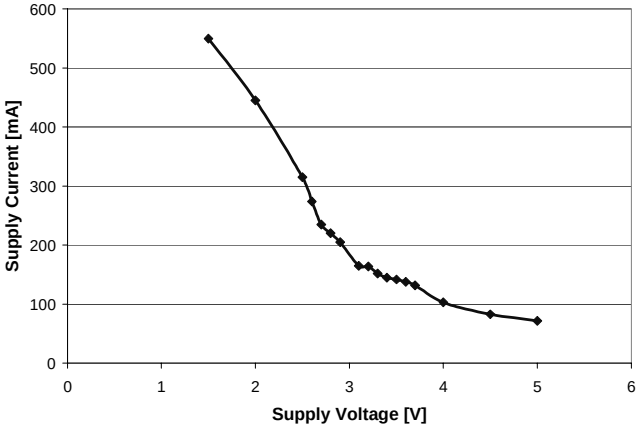

Fig. 6. Supply current vs. supply voltage.

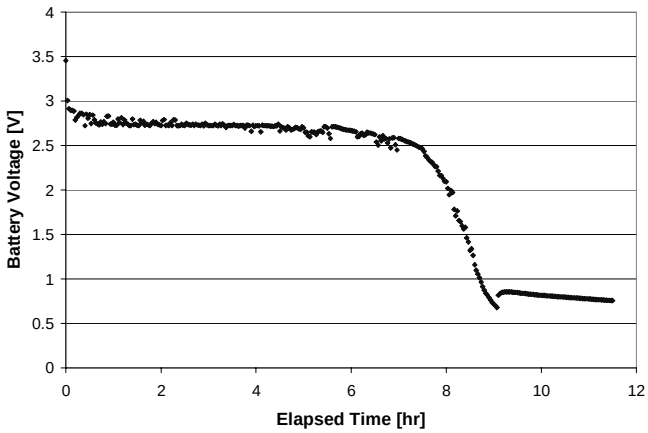

Fig. 7. Battery life for a single pair of AA batteries.

As can be seen in the first data column, under low load conditions the power consumption is independent of the signal level. As the load increases, however, the power consumption grows by as much as 40%.

With no output load, we also measured the input current while varying the power supply voltage. The results of this experiment are shown in Fig. 6. As is expected, the supply current increases with decreasing supply voltage. The efficiency of the power conditioning circuitry also decreases with decreasing supply voltage, resulting in greater power consumption when battery operated. The battery voltage vs. time for a pair of series connected AA batteries (Eveready lithium L91) is plotted in Fig. 7. As can be seen, the system runs for approximately 8 hours in this scenario.

The battery life is currently being compromised by a significant factor that can be readily corrected. In the current implementation, when the processor has completed the signal processing computations associated with an individual block of 16 samples, it waits in a spin-lock for further input from the codec. Clearly, a more power-efficient approach would be to either put the processor to sleep (i.e., enter a low-power mode) until the next input interrupt, or reduce the clock rate. Given that the DSP is currently approximately 25% utilized, a battery lifetime improvement of as much as 4× might be possible (assuming most of the power consumption is on the part of the processor).

VI. Conclusions

Given our desired use for the DHP-100 platform (the capability to perform field testing of novel hearing aid signal processing algorithms), it performs admirably. There is clearly sufficient computational capability to execute our algorithms and meet the real-time constraints. While a battery lifetime of approximately 8 hours is fairly short for a production hearing aid, it is definitely sufficient for clinical testing purposes. Recent progress toward power efficiencies that will support production hearing aids is described in [12].

In our experience, the only significant difficulty we encountered was the need to manually account for the numerical representation mismatch between the DSP and the high-level language. A compiler that understands non-integer fixed-point numerical formats would be a significant improvement over the current state of affairs.

References

[1] J.L. Goldstein, Hearing aids based on models of cochlear compression using adaptive compression thresholds, U.S. Patent Appl. #20020057808, filed August 23, 2001.

[2] T. Stetzler, N. Magotra, P. Gelabert, P. Kasthuri, and S. Bangalore, Low-power real-time programmable DSP development platform for digital hearing aids, Application Report SPRA657, Texas Instruments, April 2000.

[3] N. Magotra et al., Development of a low power digital hearing processor, Int'l Hearing Aid Research Conf., August 2000.

[4] H. Dillon, Compression? Yes, but for low or high frequencies, for low or high intensities, and for what response times?, *Ear & Hear.*, **17**:287-307, 1996.

[5] N.Y.S. Kiang et al., Single unit clues to cochlear mechanisms, *Hear. Res.*, **22**:171-182, 1986.

[6] L. Robles et al., Basilar membrane mechanics at the base of the chinchilla cochlea. I. Input-output functions, tuning curves, and response phases, *J. Acoust. Soc. Am.*, **80**:1364-1374, 1986.

[7] J.L. Goldstein, Modeling rapid compression on the basilar membrane as multiple-bandpass nonlinearity filtering, *Hear. Res.*, **49**:39-60, 1990.

[8] J.L Goldstein, M. Oz, P. Gilchrist, and M. Valente, Alternative compressive hearing aid algorithms derived from loudness psychophysics and cochlear models, Int'l Hearing Aid Research Conf., August 2002.

[9] J.L Goldstein, M. Oz, P. Gilchrist, and M. Valente, Intelligibility benefit of waveform compression in an essentially nonlinear hearing aid, NCRAR Auditory Rehabilitation Conf., Oct. 2003.

[10] J.L. Goldstein, M. Oz, P. Gilchrist, and M. Valente, Signal processing strategies and clinical outcomes for gain and waveform compression in hearing aids, in *Proc. of 37th Annual Asilomar Conf. on Signals, Systems, and Computers*, November 2003.

[11] Texas Instruments, TMS320C54x DSP Library Programmer's Reference, Literature Number SPRU518, April 2001.

[12] N. Magotra, B. Siravara, P. Gelabert, G. Gata, and J. Hochschild, Programmable ultra-low power digital signal processing (DSP) systems solution, in *Proc. of 37th Annual Asilomar Conf. on Signals, Systems, and Computers*, November 2003.