REAL-TIME LABVIEW IMPLEMENTATION OF COCHLEAR IMPLANT SIGNAL PROCESSING ON PDA PLATFORMS

V. Peddigari, N. Kehtarnavaz, and P. Loizou

Department of Electrical Engineering, University of Texas at Dallas

ABSTRACT

This paper presents the real-time implementation of a cochlear implant signal processing system on PDAs. The PDAs were chosen as they provide portable and cost-effective computation platforms. To gain software flexibility and interactivity, the LabVIEW graphical programming environment is used. The paper discusses the optimization steps which are taken to achieve a real-time throughput. These steps consist of using dynamic link libraries, utilizing efficient memory allocation, and performing fixed-point arithmetic. These steps are general purpose in the sense that the same steps can be deployed for real-time implementation of other clinical or industrial signal processing applications on PDAs.

1. INTRODUCTION

Prosthetic devices such as cochlear implants are used to restore partial hearing in profoundly deaf people or patients suffering from nerve deafness [1]. With the rising demand for implants, the field of cochlear implants has experienced a considerable growth in the last few years. The number of cochlear implant users grew from 12,000 in 1995 [1] to nearly 100,000 during the past decade.

A cochlear implant is a prosthetic device surgically implanted into the inner ear. It delivers an electrical signal to the cochlea through a series of electrodes placed inside the cochlea. It uses a speech processor to convert the sound acquired from a microphone into electrical signals [2].

Current speech processors provided by cochlear implant manufacturers are not portable and software flexible. By software flexibility, we mean the ease with which one may deploy new or modified algorithms. The only commercially available portable research processor is manufactured by Cochlear Corporation in collaboration with CRC/Hearworks [3], however, this processor does not offer software flexibility or ease-of-use as it requires a skilled programmer to develop and test algorithms in assembly language. Hence, it is deemed quite useful to have a new development platform possessing portability, flexibility, and interactivity features in order to accelerate clinical and research applications involving cochlear implants.

In this work, we consider the Personal Digital Assistants (PDAs) to serve as such a platform for implementing a cochlear implant signal processing system. We consider the PDA platform because of its portability, low-cost, and multimedia capabilities. Compared to the PCs the PDAs have limited processing power and memory. The challenge in this work has been to develop flexible cochlear implant signal processing software that would run in real-time on a relatively fast PDA (400-600 MHz clock rate).

Section 2, includes an overview of the cochlear implant signal processing system considered in this work, namely a 16-channel noise-band vocoder. Section 3 covers the implementation steps taken to achieve a real-time throughput on any PDA platform using the National Instruments LabVIEW programming environment. These steps are general purpose in the sense that they can be deployed for real-time implementation of other signal processing systems on PDAs. Section 4 includes the profiling results obtained towards achieving the real-time implementation. Finally, the conclusions are stated in section 5.

2. OVERVIEW OF COCHLEAR IMPLANT SYSTEM

All cochlear implant devices consist of the following three components: a microphone that picks up the sound, a signal processor that converts the sound into electrical signals, a transmission system that transmits the electrical signals to the implanted electrodes, and an electrode array (consisting of multiple electrodes) that is inserted into the cochlea by a surgeon. In multi-channel cochlear implants, an electrode array is inserted in the cochlea so that different auditory nerve fibers can be stimulated at different places in the cochlea depending on the frequency of the signal. Electrodes near the base of the cochlea are stimulated with high frequency signals, while electrodes near the apex are stimulated with low frequency signals. The signal processor is responsible for breaking the input signal into different frequency bands or channels and delivering the filtered signals to the appropriate electrodes.

Various signal processing strategies have been proposed in the literature for converting the acoustic signals to electrical signals [2] [4]. In this work, we implemented the popular Continuous Interleaved Sampling (CIS) strategy used in the commercially available implant devices. The CIS strategy is a vocoder strategy [4]. In order to conduct a qualitative analysis of the electrical stimuli obtained via the CIS strategy, the synthesis stage is also included along with the decomposition stage as illustrated in Fig. 1. The synthesis method implemented here is based on the noise-band vocoder simulation reported in [5].

During the decomposition stage as shown in Fig. 1, the signal is first pre-emphasized and passed through a bank of bandpass filters. The cutoff frequencies for the bandpass filters are obtained by logarithmically dividing the signal bandwidth equally over a given number of channels. The envelopes of the filtered signals are then extracted via half-wave rectification and lowpass filtering with a typical cutoff frequency of 400Hz. During the synthesis stage, the envelopes obtained after the decomposition are excited with white noise and then filtered through the same bank of bandpass filters. The synthesized signal is reconstructed by summing all the filtered signals as indicated in Fig. 1.

3. REAL-TIME OPTIMIZATION

The LabVIEW graphical programming is chosen here to achieve the real-time implementation of the cochlear implant system on PDA platforms since this environment allows one to run the same LabVIEW code on different PDAs equipped with different processors. Furthermore, it provides a higher level of abstraction through graphical system design, thus increasing the software flexibility aspect. LabVIEW graphical programs are named Virtual Instruments (VIs) and consist of two components: Block Diagram (BD) and Front Panel (FP). A BD consists of the interconnected building blocks of a system similar to a flowchart, whereas a FP constitutes its interactive graphical-user-interface incorporating various controls and displays. For more details on LabVIEW programming, the interested reader is referred to [6]. The LabVIEW PDA Module allows one to run the same code on Pocket PC, Windows Mobile, or Palm OS PDAs [7].

The FP of the cochlear implant system depicted in Fig. 2 illustrates the interactivity and ease with which various input parameters can be altered. Figure 3 shows a block-level overview of the real-time implementation process based on the CIS strategy. As shown in Fig. 3, an input frame is acquired from the microphone on a PDA device, and passed through the CIS decomposition and synthesis stages. Synthesized frames are then sent to the speakers or headphones. The filter coefficients and other input parameters are pre-computed during the design phase and are fed back continuously within a real-time loop as noted in Fig. 3.

In this implementation, input frames of 100ms length are acquired at a sampling rate of 22,050 Hz. To achieve a real-time throughput, this requires completing the processing of a frame before the next frame is captured. In other words, the total processing time for a frame should not exceed its length.

The initial LabVIEW implementation generated a processing time of nearly 2 seconds per 100ms frame. The first column in Table 1 lists the time taken by each sub-block or component of the CIS algorithm. This initial implementation is labeled Version A in the table. It should be noted that the timing information was gathered by making use of the flat sequence structure and the timing VI *Tick Count* provided in LabVIEW.

From Table 1, it can be seen that the bandpass and lowpass filtering sub-blocks are the most time-consuming sub-blocks or components. In what follows, we present the real-time optimization steps that were taken in order to bring the processing time below 100ms, thus enabling the system to run in real-time. It is worth emphasizing that these steps are general purpose in the sense that they can be deployed to run other clinical and industrial signal processing algorithms in real-time on PDA platforms.

3.1. Dynamic Link Library (DLL)

Since most PDAs have limited processing power and run fixed-point processors, one needs to minimize overheads. The LabVIEW filtering VIs use floating-point math depending on the selection of the parameters by the user. In other words, there exist overheads associated with these filtering VIs due to floating point emulation. Thus, we used optimized C codes for the bandpass and lowpass filtering sub-blocks and incorporated them as Dynamic Link Libraries (DLLs) within the LabVIEW programming environment [8]. This way we minimized the amount of floating-point overheads.

3.2. Memory Allocation (MA)

For handheld devices such as PDAs, memory management plays an important role in time-critical applications due to their limited memory resources. In general, since dynamic memory allocation requires more processing than static memory allocation, we avoided using dynamic memory allocation within the real-time loops [9]. Instead, a pre-allocated array was initialized with a constant outside of the loops. Furthermore, we de-allocated the memory that had been allocated for temporary variables within the functions of the LabVIEW DLLs. Finally, we passed a pointer to the address of the pre-allocated array instead of copying the data.

3.3. Fixed-Point Arithmetic (FPA)

Since most PDAs lack floating-point arithmetic capabilities, the next real-time optimization step consisted of re-writing the code based on fixed-point arithmetic operations. More specifically, the following modifications were made: (1) conversion of the floating point parameters to so called fixed-point Q-format, (2) computation of the output in Q-format to avoid overflows, and (3) conversion of the Q-format output back to a higher precision integer for getting the original dynamic range.

The floating point coefficients for the bandpass and lowpass filters were converted to Q-20 format during the design phase. This format ensured the stability of the filters by getting all the poles and zeros within the unit circle. Furthermore, it corresponded to the lowest Q-format that avoided overflows, and cause of any noticeable loss in accuracy.

4. REAL-TIME OUTCOME

The real-time implementation of the cochlear implant system incorporates all the optimization steps discussed above. To analyze the impact of the optimization steps on the timing performance, each optimization step was added to the initial Version A implementation in an incremental fashion and the corresponding profiling results are tabulated in Table 1. The timing numbers listed correspond to the Dell PDA PXA270 model running at a clock rate of 624MHz and using the Windows Mobile 5.0 operating system. Similar results were obtained with other PDA models.

It can be observed that Version D with all the three optimizations (Version A + DLL + MA + FPA) met the real-time processing throughput of below 100ms processing time for frame lengths of 100ms. As can be seen from Table 1, Version D took about 77ms to obtain a synthesized frame for a 100ms length input frame. Note that although Version C (Version A + DLL + MA) did not provide much improvement in terms of speed, it ensured the system stability by avoiding an inefficient memory usage. It should be noted that an actual cochlear implant system only includes the decomposition stage, which takes 52ms on the PDA platform for 100ms frames. Although the real-time constraint is met, additional steps can be taken to further lower the processing time. For example, the use of IPPs for PDAs equipped with Intel processors has been considered to further speed up the processing time.

In brief, the hybrid programming approach (LabVIEW + C DLLs) together with the fixed-point and memory optimization steps led to the real-time implementation of the cochlear implant system on the PDA platform. The optimization steps mentioned in this paper can be used to speed up any other signal processing system on PDAs.

5. CONCLUSIONS

In this paper, we discussed the need for the deployment of a PDA platform for cochlear implants to gain portability and cost-effectiveness. We demonstrated the software flexibility and interactivity offered by the PDA platform when utilizing the LabVIEW graphical programming environment. We presented three optimization steps that can be used to optimize any signal processing algorithm for real-time execution on PDAs.

6. ACKNOWLEDGEMENTS

This work has been funded by contract N01-DC-6-0002 from NIDCD/NIH. The authors wish to thank Mr. Jim Cahow, University Program Manager at National Instruments, and Dr. Arthur Lobo for their comments. Interested readers may contact Dr. Philip Loizou (contract PI) at loizou@utdallas.edu for a copy of the LabVIEW PDA code.

7. REFERENCES

[1] National Institutes of Health, *Cochlear Implants in Adults and Children*, NIH Consensus Statement Online, 1995.

[2] P. Loizou, “Mimicking the human ear,” *IEEE Signal Processing Magazine*, vol. 15, pp. 101-130, 1998.

[3] Vandali, A., Harrison, J., Van Hoesel, R., McDermott, H. and Cowan, R., “A new speech processor for cochlear implant/hearing aid research,” *Abstracts of 8th Symposium on Cochlear Implants in Children*, 2001.

[4] P. Loizou, "Speech processing in vocoder-centric cochlear implants," *Cochlear and Brainstem Implants* (ed. Moller, A.), Adv. Otorhinolaryngol. Basel, Karger, vol. 64, pp. 109–143, 2006.

[5] Shannon, R., Zeng, F-G., Kamath, V., Wygonski, J. and Ekelid, M., “Speech recognition with primarily temporal cues,” *Science*, vol. 270, pp. 303-304, 1995.

[6] N. Kehtarnavaz and N. Kim, *Digital Signal Processing System-Level Design*, Elsevier/Newnes, 2005.

[7] National Instruments, LabVIEW PDA 8.0 PDA Module, http://www.ni.com/labview/whatsnew_in_lv8_pda_module.htm.

[8] National Instruments, *How to Call External Code in LabVIEW PDA for Pocket PC*, http://digital.ni.com/public.nsf/allkb/517300B49212795986256DDD00623FEE.

[9] National Instruments, *Optimizing LabVIEW Embedded Applications*, http://zone.ni.com/devzone/conceptd.nsf/webmain.

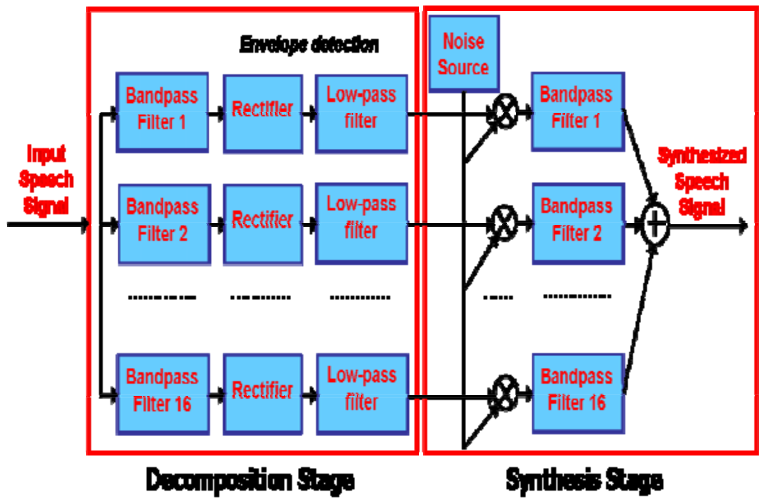
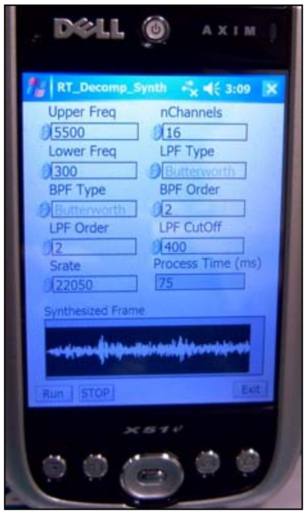


Fig. 1 Noise-band vocoder simulation of cochlear implants.



Upper Freq 5500
nChannels 16
Lower Freq 300
LPF Type Butterworth
BPF Type Butterworth
BPF Order 2
LPF Order 2
LPF CutOff 400
Srate 22050
Process Time (ms) 75

Fig. 2 Real-time LabVIEW PDA Front Panel.

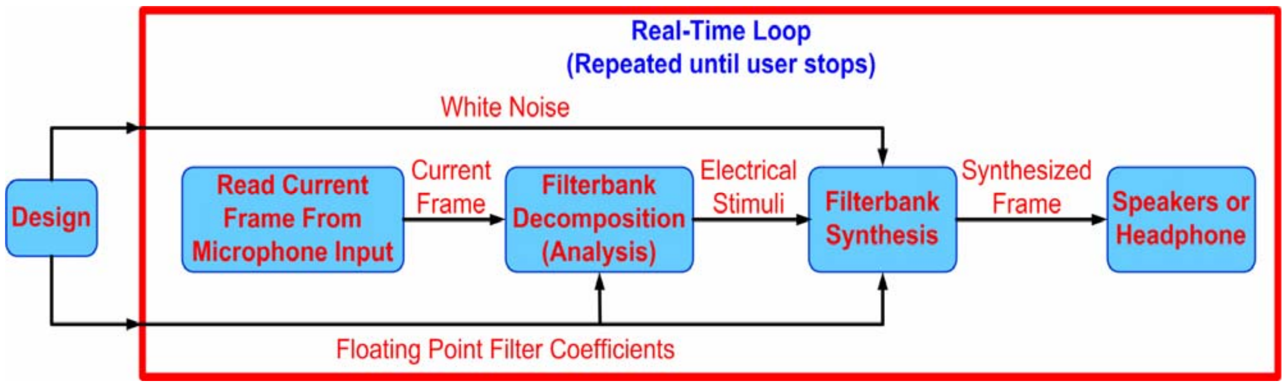

Real-Time Loop
(Repeated until user stops)
White Noise
Design
Read Current Frame From Microphone Input
Current Frame
Filterbank Decomposition (Analysis)
Electrical Stimuli
Filterbank Synthesis
Synthesized Frame
Speakers or Headphone
Floating Point Filter Coefficients

Fig. 3 Real-time implementation process of the noise-band vocoder algorithm shown in Fig. 1.

Sub-block or Component	**Processing time for 100ms length frames (ms)**			
	Version A	**Version B (A + DLL)**	**Version C (B + MA)**	**Version D (C + FPA)**
DC Offset Removal	7	5	3	1
Pre-Emphasis	11	8	5	3
Bandpass Filtering (Decomposition & Synthesis)	1450	705	412	34
Full-Wave Rectification	40	23	15	7
Lowpass Filtering	260	135	76	24
White-Noise Excitation	60	32	19	8
Total Processing Time	**1828**	**908**	**530**	**77**

Table 1 Timing outcome corresponding to the optimization steps.