REAL-TIME PDA-BASED RECURSIVE FOURIER TRANSFORM IMPLEMENTATION FOR COCHLEAR IMPLANT APPLICATIONS

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

Department of Electrical Engineering, University of Texas at Dallas

ABSTRACT

This paper presents a real-time and interactive implementation of the recursive Fourier transform approach on PDA platforms for cochlear implant signal processing applications. PDA platforms provide a cost-effective and portable platform for cochlear implant studies. The computational complexities of the two commonly used signal processing strategies that are being used in commercial cochlear implants are compared. A more computationally efficient approach using recursive Fourier transform is discussed and a real-time implementation of this approach is then accomplished on a PDA platform. Different versions of the implementation are examined and compared in terms of speed and accuracy.

***Index Terms*—**Cochlear implant signal processing, real-time signal processing, PDA platforms, recursive Fourier transform.

1. INTRODUCTION

Cochlear implants are prosthetic devices which restore partial hearing in profoundly deaf people suffering from sensorineural hearing loss. There has been a considerable increase in the number of patients outfitted with cochlear implants since their approval by the FDA. The number of patients having cochlear implants now exceeds 110,000 [1]. The major components involved in cochlear implants include a microphone that captures speech signals, a signal processor that processes the signals and a surgically implanted array of electrodes in the inner ear.

Since the introduction of cochlear implants, there has been a steady attempt at improving their signal processing algorithms to improve speech intelligibility. Several cochlear implant manufacturers, for example Med El and Advanced Bionics Corporation, offer research platforms for cochlear implant studies (e.g., see [2]), but these platforms do not meet the following three key features at the same time: cost-effectiveness, portability, and ease of programming/interactivity. In our previous work [3], we introduced a PDA-based research platform to gain cost-effectiveness and portability as well as a LabVIEW-based hybrid programming approach to gain interactivity and software flexibility. In this paper, we have improved our previous solution in terms of processing speed by considering a recursive Fourier transform approach while maintaining the software flexibility aspect of our solution on any PDA platform.

Section 2 describes the two common signal processing approaches that are used in commercial cochlear implants, one being the filterbank approach and the other being the FFT approach. Section 3 describes how the use of recursive Fourier transform reduces the amount of computation compared to the commonly used methods. Section 4 covers the implementation of the recursive approach. In addition, it includes a comparison of our various implementation versions in terms of speed and accuracy. Finally, the conclusions are stated in section 5.

2. COMMON COCHLEAR IMPLANT SIGNAL PROCESSING STRATEGIES

The function of the signal processor in a cochlear implant primarily consists of dividing an input speech signal into a number of frequency bands (12-22) in order to extract the signal strength in each band for exciting the implanted electrodes accordingly. In other words, the signal processor is programmed to emulate the functioning of the inner ear. The most common strategies used in commercial cochlear implants are Continuous Interleaved Sampling (CIS) and Advanced Combination Encoder (ACE) [4][5]. Both of these strategies can be realized using either a filterbank approach or an FFT approach.

When using the filterbank approach, a set of bandpass filters is used to divide the signal into a number of frequency bands or channels. Then, the filtered output is fullwave rectified and passed through lowpass filters to extract channel envelopes based on which, electrode pulses are generated. Figure 1(a) illustrates the CIS strategy where the input signal is passed to a bank of bandpass filters. The computational complexity of this approach increases as the number of channels and the order of the filters are increased. The same output can be achieved by computing the FFT of the input signal, grouping the frequency bins into different channels, and then summing up the power of adjacent frequency bins falling in a channel to obtain the signal strength in that channel. Figure 1(b) provides an illustration of the FFT approach.

When using the FFT approach, FFTs cannot be computed at the input sampling rate due to the computational demand associated with calculating FFT per sampling time interval. The output rate is often determined based on the rate at which FFTs can be computed in real-time. In the ACE implementation reported in [4], 128-point FFTs are computed every 9^{th} input sample at an input sampling rate of 16kHz, thus giving a 125Hz frequency bin resolution and a channel output rate of about 760Hz, which is much lower than the input sampling rate.

To achieve higher stimulation rates, the channel output rate has to be increased. In general, depending on the channel output rate and the excitation method used, the electrodes are excited at the rate of 800 to 2500 pulses/sec/channel [5]. The highest rate achievable corresponds to the input sampling rate which means shifting the frame or window one sample at a time. When the window is shifted by one sample, there exists a significant amount of overlap from one window to next. Noting this fact, a recursive computation of DFT is considered in this paper to allow achieving high stimulation rates. The recursive approach for updating DFT is shown to be much more computationally efficient, thus leading to a real-time throughput on PDA platforms.

3. RECURSIVE COMPUTATION OF DISCRETE FOURIER TRANSFORM

As discussed in [6], for a new window or frame shifted from a previous window by one sample position, Fourier transform can be updated in a recursive manner. Such an approach is computationally attractive for cochlear implant applications as it allows one to achieve high stimulation rates. The recursive updating of Fourier transform at frequency ω and time instant n+1 can be updated from the Fourier transform of the previous window as follows [6]:

$$F(\omega, n+1) = e^{-j\omega}F(\omega, n) + x(n+1) - e^{-j\omega N} x(n-N+1) \tag{1}$$

where x(n+1) represents the sampled input at instant n+1, N the window size and F(ω,n+1) the Fourier transform at instant n+1 and frequency ω. In Equation 1, the newest sample gets added while the oldest sample is subtracted after getting phase shifted. To update an N-point DFT at frequencies $\omega_{k} = (2\pi k/N)$, k taking integer values 0, 1,... N-1, Equation 1 can be rewritten as follows:

$$F(\omega_k, n+1) = e^{-j\omega_k}F(\omega_k, n) + x(n+1) - x(n-N+1) \tag{2}$$

As seen from Equation 2, the DFT computation at a single frequency point requires only one complex multiplication.

If a smooth frequency response is desired, a Hanning, a Hamming, a triangular, or an all-pole window may be used as discussed in [7][8]. Note that the use of these windows increases the computational complexity by a factor of 2. For example, as described in [8], a second order all-pole window represented by:

$$W(z) = 1/(1-\beta z)^2 \tag{3}$$

where

$$\beta = \exp\left(-\frac{3+2L}{2+3L+L^2}\right) \tag{4}$$

and L denoting a delay factor, results in the following recursive equation:

$$F(\omega_k, n) = x(n) + 2\beta e^{-j\omega_k}F(\omega_k, n-1) - \beta^2 e^{-j2\omega_k}F(\omega_k, n-2) \tag{5}$$

As an example, Table 1 lists the number of multiplications for 46.4ms (1024 samples) input speech frames at a sampling rate of 22kHz with 22 channels using the filterbank, the 128-point FFT, and the recursive 128-point DFT update methods. For the filterbank method, the bandpass filters are considered to be of 6^{th} order (Butterworth) and the lowpass filters of 4^{th} order.

4. REAL-TIME IMPLEMENTATION ON PDA PLATFORMS

Our implementation was performed within the LabVIEW hybrid programming framework previously reported in [3]. This hybrid framework allows one to incorporate C code within the interactive environment of LabVIEW. As a result, one can easily change various parameters of the signal processing algorithm. Through the interactive Front Panel capability of LabVIEW, users can alter sampling rate, frequency spacing of channels, number of channels, and other parameters, as shown in Figure 2. There is an option to make the channel spacing logarithmic or linear from 300Hz to 1800Hz and logarithmically thereafter. The C code corresponding to the signal processing components was placed in the LabVIEW environment as a Dynamic Link Libraries (DLL). The twiddle factors and the channel bandwidths were computed in the graphical environment of LabVIEW and passed as parameters to the C code. All the optimization steps as discussed in [3] were also performed for this implementation.

In addition, the implementation was done in fixed-point arithmetic with word size of either 16 bit or 32 bit. In general, there is a loss of accuracy when performing fixed-point implementation. This issue was addressed by adopting a 32 bit word size. Furthermore, due to the steady increase in the quantization error after every iteration, the DFT update was terminated every 50ms. This allowed the percentage mean-square quantization error to remain within 1% of the floating-point high-precision implementation for a reset rate of 1024 frames. It should be mentioned that the overhead incurred by computing FFT once every 50ms did not impose a real-time limitation as it only added 1ms to the total processing time. For all the implementation versions, the twiddle factors were kept in the Q15 (15 fractional bits) fixed-point format. Table 2 gives the percentage of time spent by each of the sub-processes and Table 3 lists the real-time timing outcomes on an Intel processor-based PDA with a clock speed of 625MHz.

Figure 3 shows the percentage mean-square error of two of the channel envelope outputs (1 and 15) between the fixed-point and floating-point versions for the 16 and 32 bit implementations. Here the channel cutoff frequencies were placed linearly in the lower range and logarithmically for frequencies above 1800 Hz.

An additional optimization step was also carried out for Intel processor based PDAs. This step comprised incorporating Intel Integrated Performance Primitive (IPP) functions within the C code. These functions have been fully optimized for Intel processors. The performance improvement when using the IPP functions is also listed in Table 3.

In all of our implementations, special attention was paid to minimizing the use of dynamic memory allocation and freeing buffers as soon as a function was done. Our PDA implementation also included a synthesized component to play back reconstructed speech signals using the Windows operating system APIs (application programming interfaces). The synthesis was done using the noise-band vocoder strategy described in [5][9].

5. CONCLUSION

In this paper, various real-time implementations of cochlear implant signal processing on PDA platforms were investigated. All the implementations except for the IPP versions are platform independent meaning that they can be run on any PDA platform. It is shown that the most computationally efficient and interactive approach consists of the use of recursive Fourier transform allowing one to obtain Fourier transform at the input sampling rate. Furthermore, by utilizing 32-bit fixed-point arithmetic and a resetting procedure, it is ensured that the quantization error remains very small as compared to the floating-point version.

6. ACKNOWLEDGMENT

This work has been funded by the NIH contract N01-DC-6-0002 from NIDCD. 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 Institute on Deafness and Other Communication Disorders, "Cochlear Implants," National Institutes of Health, Bethesda, MD, 2005 http://www.nidcd.nih.gov/health/hearing/coch.asp.

[2] P. Loizou, G. Stickney, M. Mishra, and P. Assmann, "Comparison of speech processing strategies used in the Clarion implant processor," *Ear and Hearing*, vol. 24, no. 1, pp. 12-19, 2003.

[3] V. Peddigari, N. Kehtarnavaz, and P. Loizou, "Real-time LabVIEW implementation of cochlear implant signal processing on PDA platforms," *Proc. of IEEE Intern. Conf. Acoust. Speech, Signal. Processing,* Hawaii, April 2007.

[4] A. Vandali, L. Whitford, K. Plant and G. Clark, "Speech perception as a function of electrical stimulation rate using the Nucleus 24 cochlear implant system," *Ear Hear.* , vol. 21, no. 6, pp. 608-624, 2000.

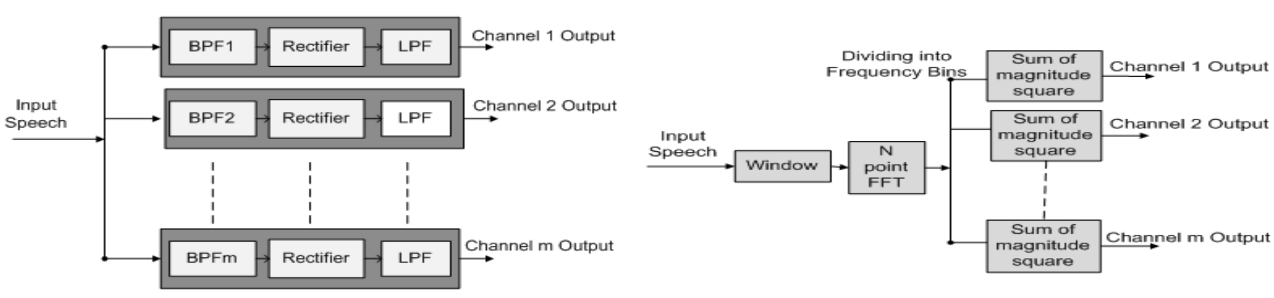
[5] P. Loizou, "Speech processing in vocoder-centric cochlear implants," *Cochlear and Brainstem Implants* (ed. Moller, A.), Adv. Otorhinolaryngol. Basel, Karger, vol. 64, pp. 109–143, 2006.

[6] M. Amin, "A new approach to recursive Fourier transform," *Proc. of IEEE*, vol. 75, pp. 1537-1538, 1987.

[7] E. Sherlock, D. Monro, "Moving discrete Fourier transform," *IEE Proceedings on Radar and Signal Processing*, vol. 139, no. 4, pp. 279-282, 1992.

[8] W. Chen, N. Kehtarnavaz, T. Spencer, "An efficient recursive algorithm for time-varying Fourier transform," *IEEE Transactions on Signal Processing*, vol. 41, no 7, pp. 2488-2490, 1993.

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



(a) (b)

Fig.1 - Two common signal processing strategies for cochlear implants based on: (a) Filterbank approach, and (b) FFT approach.

Number of real multiplications for 1024 sample frames, 128-point FFT, 22 channels	Filterbank	FFT (N-point)	Recursive DFT update	
			Rectangular window	2^{nd} order all-pole window
	No. of channels * (3 stage cascaded 2^{nd} order BPF + 2 stage cascaded 2^{nd} order LPF)	$2N(\log_2 N) + (N/2)*2$	$(N/2)*4 + (N/2)*2$	$(N/2)*8 + (N/2)*2$
	≈560K	≈2M	≈400K	≈660K

Table 1 - Computational complexity of different implementations of cochlear implant signal processing strategies.

Percentage time of sub-processes	Read speech input and write output	Recursive DFT update	DFT magnitude square computation	Analysis channel output computation
	5%	45%	30%	20%

Table 2 – Percentage time spent by sub-processes.

Specifications: 16 channels Filterbank: 6^{th} & 4^{th} order BPF & LPF FFT: 128-point	Filterbank		FFT						
			Non recursive	Recursive					
	Non interactive	Interactive	Non interactive	Interactive					
	Using C + IPP+ Assembly	Using LabVIEW + C	Using C + IPP	Rectangular window - Using LabVIEW + C		2^{nd} order all-pole window - Using LabVIEW + C		Rectangular window - Using LabVIEW + C + IPP	
				16bit	32bit	16bit	32bit	16bit	32bit
	6	19	248>46.4 (non real-time)	11	13	15	16	8	10

Table 3 - Timing outcomes corresponding to different implementations on a PDA platform. Numbers indicate time (in ms) required to process 46.4 ms duration frames.

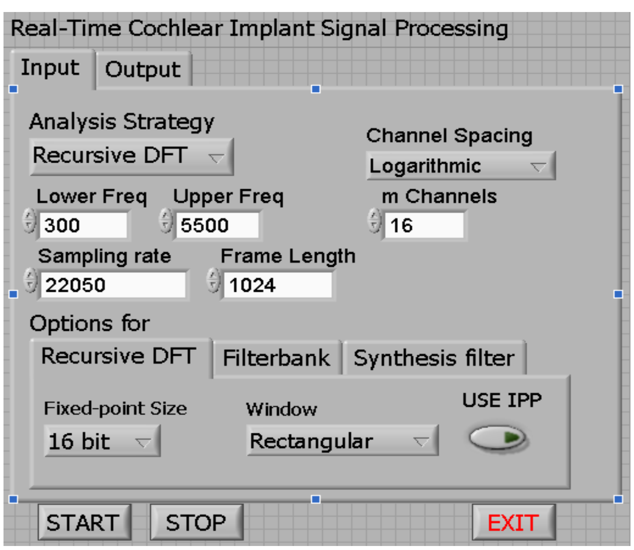


Fig.2 – Front Panel of real-time PDA implementation.

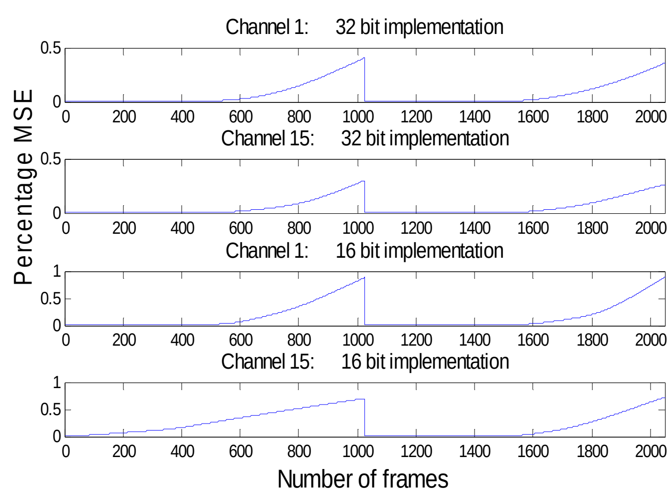


Fig. 3 – Percentage mean-square error (MSE) between envelope outputs obtained using fixed-point and floating-point.