REALIZATION OF AN ACOUSTIC ECHO CANCELLER ON A SINGLE DSP

Gerard Egelmeers, Piet Sommen and Jacob de Boer
Eindhoven University of Technology (TUE)
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
Tel: +31 40 2473634; fax: +31 40 2455674
e-mail: p.c.w.sommen@ele.tue.nl

ABSTRACT

An Acoustic Echo Canceller (AEC) based on the Decoupled Partitioned Block Frequency Domain Adaptive Filter (DPBFDAF) [3, 4] is implemented on a single Digital Signal Processor (DSP), the TMS320C30. This flexible setup makes it possible to choose the sample frequency (f_s), the number of coefficients (N) of the adaptive filter and the processing delay independent of one another (only limited by the total complexity). Two implementation examples are given: one with $N = 2016$ and f_s=7 kHz with a processing delay of 1.6 msec., the other one with $N = 2560$ and f_s=13kHz with a processing delay of 6.5 msec. It is shown that the setup works both for a white noise input signal and a real speech signal.

1 Acoustic echo cancellation

In fig.1 the AEC scheme is depicted. The filter part of the AEC performs the convolution of the input signal $x[k]$ and the adaptive weight vector $\underline{w}[k]$, while the update of the weight vector is done in the update part. The goal of the AEC is to produce an estimate $\hat{e}[k]$ of the echo $e[k]$ and subtract it from $\tilde{e}[k]$ such that, in average, only the local speech $s[k]$ remains in the residual signal $r[k]$.

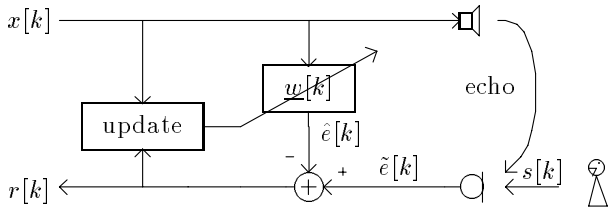

Figure 1: AEC scheme.

2 Towards a single DSP solution of AEC

A well known adaptive filter algorithm is the (Normalized) Least Mean Square algorithm. However it has a too high computational complexity for real time implementation on a single DSP. Besides that its convergence behaviour is strongly affected by the input signal charateristics. By using Fast Fourier Transforms (FFTs) and block processing techniques, adaptive filters can be implemented with much lower computational complexity and much better convergence behaviour. This results in the Block Frequency Domain Adaptive Filter (BFDAF) [1]. However an efficient realization of this algorithm for large adaptive filters results in a large processing delay. A combination of moderate complexity and a smaller processing delay can be achieved by partitioning the filter and update part of that algorithm, resulting in the Partitioned BFDAF (PBFDAF) [1].

In [2] an *unconstrained* version of this algorithm is used to realise an AEC with $N = 2048$ coefficients at a sample frequency of 16 kHz on a single DSP. The processing delay of this algorithm equals *32* msec. This setup has two disadvantages: The first one is the processing delay that is still large. This is a direct consequence of the need to use large blocks to obtain an efficient realization of the frequency domain approach. Secondly in [2], for complexity reasons, the unconstrained approach is choosen for every separate partition. This however degrades the convergence properties a lot. In this unconstrained approach all FFTs (used for the windowing in each separate partition) are left out. However these window FFTs are needed to perform an exact linear convolution using cyclic ones. In [1] it is shown that these windows can be left out in the BFDAF (not partitioned) resulting in a small increasement of the final misadjustment. For the BFDAF case this is allowed because the adaptive filter length is choosen in such a way that the unknown impulse response is rougly zero outsite this range. This statement does not hold any more for each separate partition in the PBFDAF case. The result is that leaving out all FFTs in each separate partition degrades considerably the convergence properties of the adaptive filters.

Since the block length of the filter part determines the processing delay and the block length in the update part controls the complexity, we can obtain both low complexity and a small processing delay by using different partition factors and block lengths in update and filter part. This results in the Decoupled PBFDAF (DPBFDAF) [3]. Besdides that, we can use a large nor-

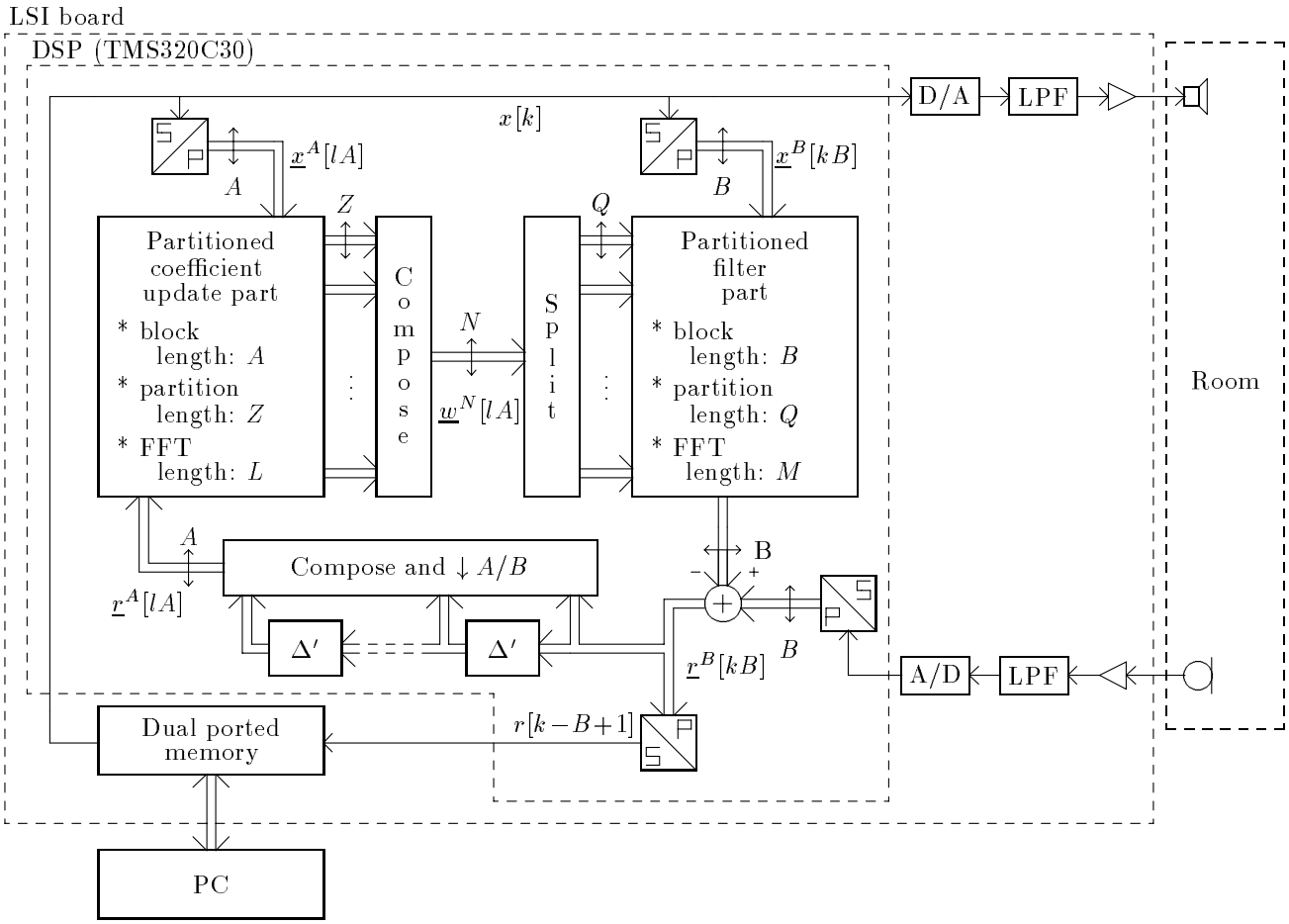

Figure 2: Implementation scheme.

malization vector to decorrelate the input signal with high resolution in the update part. In this paper the DPBFDAF algorithm, with low complexity and small processing delay, is used to realize an AEC on one single DSP. The implementation scheme is depicted in fig.2. In this figure it is shown that both filter and update part are partitioned separately [1]. The filter part is partitioned in length Q partitions while the FFTs used are of dimension M. Furthermore by choosing a small block length B for the filter part this scheme results in a small processing delay. To decrease complexity the block length A of the update part is choosen large. Here the length of each partition equals Z and the FFT dimension is L. Because of the different partition factors for the update- and filter-part the weight vectors that results from the update part have to be composed and split to weight vectors that are needed for the filter part. Furthermore the residual signal vector $\underline{r}^B[kB]$ (length B) that results from the filter part has to be delayed (delay elements Δ') and composed to a residual signal vector, $\underline{r}^A[kA]$ (length A) that is needed for the update part. Finally the figure shows that the input signals samples are converted by a Serial to Parallel converters (S/P) to signal vectors. The output signal vector is converted back to signal samples $r[k-B+1]$ by the Parallel to Serial converter (P/S).

3 Implementation

3.1 Hardware

The AEC implementation is realized on a single DSP, the Texas Imstruments TMS320C30. It has a capacity of 33.3 MFLOPS (Million FLoating point Operations Per Second) and is mounted on a Loughborough Sound Images (LSI) prototyping board, inserted in a PC-slot. The hardware configuration is also depicted in fig.2.

The input signal $x[k]$ is taken from PC memory, and fed to a speaker through a Digital to Analog (D/A) converter, Low-Pass Filter (LPF) and amplifier, controlled by the DSP. In the same room the microphone is situated, whose output is directed towards an Analog to Digital (A/D) converter via an amplifier and a low-pass filter, also controlled by the DSP. The residual signal $r[k]$ is returned to the PC memory (from which it can eventually be sent towards a loudspeaker). Communication between PC and DSP takes place via the Dual Ported memory, that can be accessed by the PC without

[1] For a detailed description of the partitioning concept we refer to [1, 4]

halting the DSP.

3.2 Software

The PC controls the AEC software, running on the DSP. This AEC implementation consists of several modules:
Main part: Initialization of the AEC.
Interrupt Service Routine (ISR): Because of the very small block length in the filter part and large block length in the update part of the adaptive filter, many very small time frames are available to compute the update part. Therefore a complex interrupt scheme is needed to control the spread of the calculation over several time-frames. Interrupts are also needed for the A/D en D/A converters.
Filter part: Calling (I)FFTs and performing elementwise multiplications. To obtain an efficient elementwise multiplication routine, all data must reside in the DSP's on chip internal memory. Because of the limited amount of internal memory, the DMA (Direct Memory Access) has to run in parallel to transport data from the external (on board) memory to the internal memory (and back).
Update and coupling part: Calling (I)FFTs and performing elementwise multiplications.
Stepsize Control And Normalization (SCAN): The stepsize influences the misadjustment and the speed of convergence. Initially, we prefer a fast convergence, implying a large stepsize parameter. When the residual signal decays, the stepsize parameter is decreased, to obtain a smaller misadjustment. When "double talk" occurs, the large residual signal implies a huge coefficient misadjustment or even instabillity. Adaption of the coefficients must be inhibited during "double talk". This implies that we need a "double talk" detector.
(I)FFT: The (Real) (I)FFT routines available for the TMS320C30 processor were not efficient enough, so new routines have been developed, that are about 30% faster than previously available routines.

3.3 Two examples

In principle the variables of the DPBFDAF algorithm can be chosen freely, although certain maxima exist to make the implementation less complicated. In table 1 these bounds are given, together with two possible variable sets as example. In table 2 the computational load of the two examples is given. The load of each seperate part is defined as a percentage of the DSP's total available number of instructions. The overhead parts (Rest) contain some data-rearranging and initialization of loops.

4 Tests

4.1 Test Conditions

The test room has dimensions $4.90 \times 3.75 \times 3.20$ m^3. The distance between the loudspeaker and microphone is 0.40 m. As we can see in figure 2, the AEC does not only have to cancel the actual echo path, but also the (non ideal) impulse responses of the D/A and A/D converters, the LPF's, the amplifiers, the loudspeaker and the microphone. As input signal we first use white noise, to show that the AEC indeed works. After that, the sentence "entering the forest without moving the grass" is taken as input signal. This is depicted in figure 3. The non-stationarity of the input signal implies that we can not look at the residual signal only to investigate the performance, but we have to normalize it to obtain the Echo Return Loss Enhancement, defined by $-10 \cdot \log_{10}(\mathcal{E}\{r^2[k]\}/\mathcal{E}\{\hat{e}^2[k]\})$.

Variable			Max.	Ex.1	Ex.2
Filter len.		N	8192	2016	2560
Filter Part	Block len.	B	256	8	64
	Part. len.	Q	256	56	192
	FFT size	M	256	64	256
Update Part	Block len.	A	512	504	512
	Part. len.	Z	1024	504	512
	FFT size	L	1024	1024	1024
Sample frequency		f_s		7 kHz	13 kHz
Echo path length				288 ms	192 ms
Processing delay				1.6 ms	6.5 ms

Table 1: Maxima and examples for variable sets.

Part	Rate	Oper.	#	Ex.1	Ex.2
Filter part	f_s/B	FFT_M	1	5.0%	6.3%
		IFFT_M	1	5.9%	7.4%
		$\otimes$	N/Q	58.3%	16.7%
		Rest	1	1.3%	0.8%
Update part	f_s/A	FFT_L	2	4.6%	8.3%
		IFFT_L	N/Z	10.3%	20.9%
		$\otimes$	N/Z	1.5%	2.7%
		Rest	1	4.0%	3.1%
Coupling	f_s/A	FFT_M	N/Q	3.3%	10.3%
		Rest	1	0.2%	0.9%
SCAN	f_s/A		1	1.2%	2.3%
ISR	f_s		1	2.8%	5.0%
Total				98.4%	84.7%

Table 2: Computational load of algorithm.

4.2 Test Results

In the first test, both examples have a white noise input signal, with the adaptive filter weights initialized to zero. The results are given in figure 4.

The second experiment involves the introduction of a non-stationarity in the echo path. This is done by placing a hand between microphone and loudspeaker. In fig.5 the effect of this sudden change in the echo path impulse response is shown. Since this experiment has been performed manually it can be seen in the figure that the non-stationarity does not take place at exactly the same time in the examples. From these two experiments it follows that the AEC works well when a white noise signal is used as an input signal. The third experiment involves a real speech signal. The sentence "entering the forest without moving the grass" is ap-

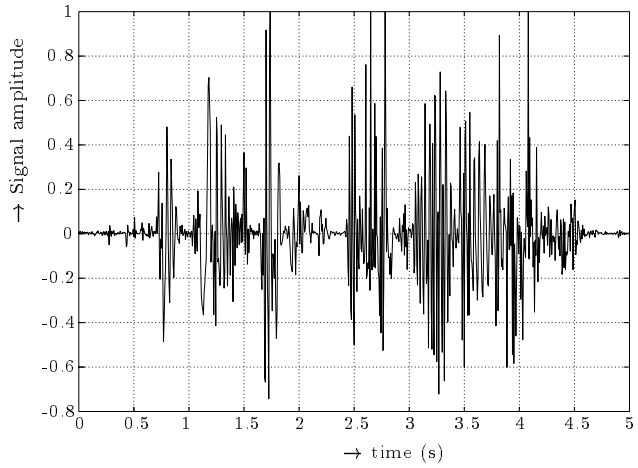

Figure 3: Speech signal: "entering the forest without moving the grass".

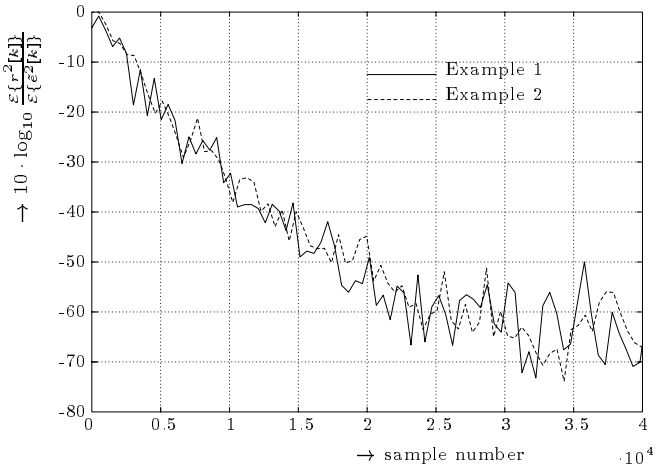

Figure 4: Convergence behaviour, $x[k]$ white noise.

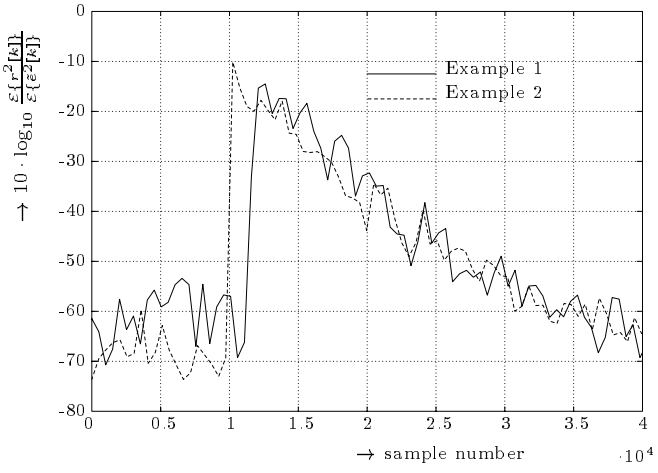

Figure 5: Convergence behaviour with non-stationarity, $x[k]$ white noise.

plied to the AEC (using the system settings of example 2), and the result is depicted in fig.6. The initial state of this experiment was choosen in such a way that the adaptive filter was fully adapted when the experiment starts. Comparing this figure with fig.3 shows that when the input signal amplitude is high, good suppression of the echo is obtained.

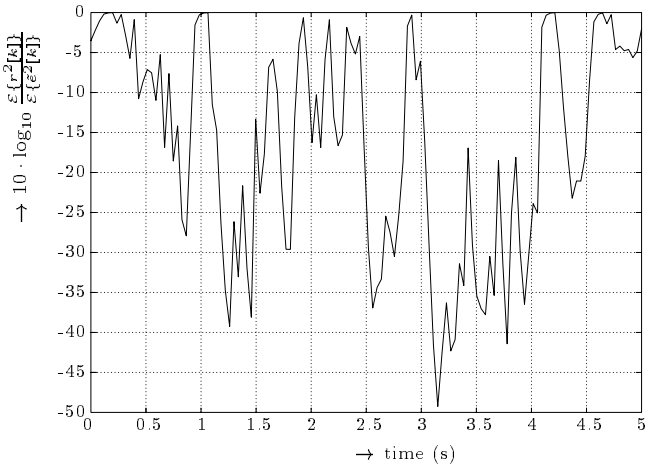

Figure 6: Convergence behaviour, $x[k]$ speech at 13 kHz.

5 Conclusions

A real-time implementation of a low delay acoustic echo-canceller is realized on one Digital Signal Processor (the TMS320C30) using the Decoupled PBFDAF (DPBFDAF) algorithm. Two implementation examples are given: one with $N = 2016$ and f_s=7 kHz with a processing delay of 1.6 msec., the other one with $N = 2560$ and f_s=13kHz with a processing delay of 6.5 msec. It is shown that the setup works both for a white noise input signal and a real speech signal.

References

[1] Sommen P.C.W., *Adaptive Filtering Methods.* Eindhoven (The Netherlands): Thesis Eindhoven University of Technology, June 1992, ISBN 90-9005143-0.

[2] Páez Borallo J.M. and Garcia Otero M., "On the implementation of a partitioned block frequency domain adaptive filter (PBFDAF) for long acoustic echo cancellation," *Signal Processing*, vol. 27, no. 3, June 1992, pp. 301-315.

[3] Egelmeers G.P.M., "Decoupling of partition factors in Partitioned Block FDAF", in *Proc. ECCTD 1993* (Davos, Switserland), Aug. 1993, pp. 323-328.

[4] Egelmeers G.P.M., *Real Time Realization Concepts of Large Adaptive Filters.* Eindhoven (The Netherlands): Thesis Eindhoven University of Technology, Nov. 1995, ISBN 90-386-0456-4.