Word-length optimization of an Adaptive Noise Canceller

V. Rodellar, A. Muñoz, A. Álvarez , E. Martinez, C. González, and P. Gómez
Facultad de Informática
Universidad Politecnica de Madrid
Campus de Montegancedo – Boadilla de Monte (28660 – Madrid)
SPAIN

Abstract: - In this paper the determination of the optimal word-length of the variables implicated in a noise adaptive canceller based on a gradient lattice-ladder algorithm is presented. Upper and lower bounds from the variables are determined from a set of spoken words.

Key-Words: Noise cancellation, speech enhancement, integer arithmetic, data format optimization.

1 Introduction

There are several applications as advanced video games, virtual reality, automotive systems, man-machine communications, aircraft control, mobile telephony etc. that may operate in adverse environments with high levels of noise and whose spectral and power characteristics are continuously changing. There are several techniques to reduce noise levels [1] [2]. Adaptive Noise Cancellation for non-stationary environments in the time domain is an adequate technique presenting a competitive performance. This filter computes the input information sample by sample which is very convenient for real time processing but its principal disadvantage are the need of calculating several multiplication and division operations and the wide range in the upper and lower bounds of the data. The implementation of these algorithms has been done traditionally with general-purpose DSP microprocessors using floating-point arithmetic [3]. These implementations minimize the round-off errors but tend to be limited in processing speed because they have usually available a single processing unit. In microprocessors implementation the word-length is defined by the hard-wired architecture but in reconfigurable computing the size of each variable maybe customized in order to get the best tradeoffs in numerical precision, speed, size and power consumption. It is shown that reconfigurable computing designs are capable of achieving up to 500 times speed up and 70% energy savings over microprocessor implementations for specific applications [4]. The problem of word-length optimization is NP-hard [5] and different approaches have been adopted and tools have been developed to its treatment [4][6].

In this paper we will present the word-length optimization problem of an Adaptive Noise Canceller under the point of view of the accuracy performance of the algorithm as preliminary stage to a hardware implementation with reconfigurable logic. The accuracy will be studied by observing the outputs of the system as a function of the word-lengths used to represent all the intermediate variables in the algorithm. The problem approach has been done by means of simulation using as input stimulus a proprietary data base of spoken commands.

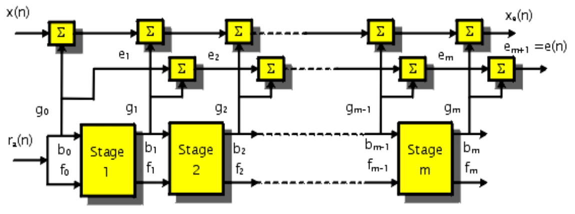
2 The adaptive noise canceller

The noise canceller under study has interesting characteristics for applications that demands speech enhancement techniques, which will be presented next [7]:

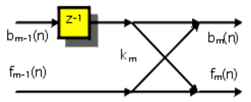
- No a priory knowledge of the characteristics of the signal or noise is needed.
- Shows a very good behavior in highly non-stationary environments.
- Preserves the quality of the speech.
- Provides an immediate response.
- Makes possible further processing of the speech.
- Removes interfering sources arriving in similar conditions to both channels.
- No band suppression or artificial tone introduction was appreciated in direct hearings, the quality of the resulting speech being noticeable good.

2.1 Filter structure

The basic scheme is based on a two microphone structure. One for noisy speech (primary, x(n)) and the other the noise itself (reference, $r_a(n)$). The primary and reference microphones must be separated a given distance in order to avoid crosstalk. In our case, a cancellation average from 6 to 12 dB is obtained for a sampling rate of 11.025 Hz, microphone separation of 20 cm and 14 processing stages. As can be seen in Fig.1, the structure of the Gradient Lattice-Ladder filter has two main parts: the stages, which are lattice filters and the ladder which is removing the noise from the signal.



Structure of the Gradient Lattice-Ladder Equalizer



Structure of a given Lattice stage

Fig. 1. Filter block diagram

The noise estimated by the lattice filter, (which is adapted by its estimation errors) and its backwards residuals ($b_m(n)$) are used to adapt the weights of the ladder filter. Estimated noise $x_e(n)$ and clean speech e(n) are obtained in the last stage of the ladder part of the filter.

2.2 Computational requirements

The computational steps of the algorithm are shown next. They are basically three: initialization, lattice filter and ladder filter calculations. In the initialization stage, the adaptive parameter for samples and stages are set to one. The initial values of the residual backward and forward errors are ε = 5.10 ^{8} and the adaptation step w = 0,9999. In the lattice stage, the Parcor and reflection coefficients are updated, the backward, forward and residual error is calculated and the adaptive parameter updated. And finally, the ladder filter calculates a gain factor, estimates the noise and gives as a final result the clean signal.

The convergence rate of this algorithm is good and has a computational complexity of N. The algorithm requires 6 divisions per stage and $2N_{stages}$ +1 division per sample. One of the advantages of the algorithm is that the amount of memory required is very small. The processing of the speech samples can be done as soon as they are available therefore only a small array for speech input data is needed instead of large buffers. The algorithm steps are described next:

$\boxed{\textbf{Initialization}}$: Being n the number of sample, and m the number of stage

Samples:

$\alpha_m(-1)=1;$	Adjust parameter
$k_m(-1)=0;$	Parcor coefficient
$r_m^b(-1)=r_m^f(0)=\varepsilon>0;$	Residual backward and forward errors
$d_m(-1)=0;$	Escalar to compact the ladder part result

Stages:

$\alpha_0(n)=1;$	Adjust parameter
$e_0(n+1)=x(n+1);$	Input signal (primary)

Reference signal, forward and backward errors

$$f_0(n+1)=b_0(n+1)=r_a(n+1);$$

Residual forward and backward errors

$$r_0^f(n+1)=r_0^b(n+1)=wr_0^f(t)+\left|ra(n+1)\right|^2;$$

$\boxed{\textbf{Lattice Filter}}$: It begins with n=0 and it computes the updates for m=0,1,…N-2; with N=14.

$$k_{m+1}(n)=w_2k_{m+1}(n-1)+\alpha_m(n-1)f_m(n)b_m(n-1);\text{ Parcor coefficients update}$$

$$\psi_{m+1}^f(n)=-\frac{k_{m+1}(n)}{r_m^b(n-1)}\quad \psi_{m+1}^b(n)=-\frac{k_{m+1}(n)}{r_m^f(n-1)};\text{ Reflection coefficients update}$$

$$f_{m+1}(n+1)=f_m(n+1)+\psi_{m+1}^f(n)b_m()$$

$$b_{m+1}(n+1)=b_m(n)+\psi_{m+1}^b(n)f_m(n+1)$$

$$r_{m+1}^f(n)=r_m^f(n)-\frac{\left|k_{m+1}(n)\right|^2}{r_m^b(n-1)}$$

$$r_{m+1}^b(n)=r_m^b(n-1)-\frac{\left|k_{m+1}(n)\right|^2}{r_m^f(n)}$$

$$\alpha_{m+1}(n)=\alpha_m(n)-\frac{\alpha_m^2(n)\left|b_m(n)\right|^2}{r_m^b(n)}$$

$\boxed{\textbf{Ladder Filter}}$: It begins with n=0 and it computes the updates for m=0,1,…N-1; being N=14

$$d_m(n)=wd_m(n-1)+\alpha_m(n)b_m(n)e_m(n)$$

$g_m(n)=-\dfrac{d_m(n)}{r_m^b(n)};$	Gain factor
$x_{e_m}(n)=x_{e_{m-1}}(n)+b_m(n)g_m(n);$	Estimate noise
$e_{m+1}(n+1)=e_m(n+1)+g_m(n)b_m(n+1);$	Clean signal

2.3 Upper and lower bounds of the variables in floating-point representation.

The algorithm was written in ANSI-C and tested with a set of commands in English and Spanish. The command set was recorded from 32 speakers from both sexes (equally distributed) in an age from 20 to 45. The records were done under strong environmental noisy conditions (95-100 dB). The English commands set being used was: double, down, eight, end, five, four, go, hit, jump, last, left, next, nine, no, off, on, right, seven, six, split, start, stop, ten, turn, two, up, yes and zero. The set of Spanish commands used was: aceptar, adelante, adentras, cancelar, cero, cinco, cuatro, dos, enviar, establecer, fax, información, internet, marcar, mensaje, menú, nueve, ocho, recibir, repetir, seis, servicio, siete, teléfono, texto, tres, and uno.

The bounds for worst case results for floating point arithmetic are shown in Table 1.

	UPPER BOUND	LOWER BOUND
Reference canal sample $r_a(n)^2$	12.411.529,00	0,00
Inicial residual fordward and backward errors $r_0^f(n+1) = r_0^b(n+1)$	1.663.767.296,00	76.082.344,00
Parcor coefficient $k_{m+1}(n)$	1.318.524.032,00	-314.949.312,00
Reflection Coeficientes $\Psi_{m+1}^f(n)$ $\Psi_{m+1}^b(n)$	0,56 0,56	-0,80 -0,80
Fordward error $f_{m+1}(n)$	2.497,71	-2.864,51
Backward error $b_{m+1}(n)$	2.281,11	-2.788,99
Residual forward error $r^f_{m+1}(n)$	728.626.176,00	76.068.048,00
Residual backward error $r^b_{m+1}(n)$	728.131.264,00	76.067.752,00
Adaptive parameter $\alpha_{m+1}(n)$	1,00	0,96
Auxiliary variable for gain factor $d_m(n)$	2.598.985.472,00	-144.405.008,00
Gain factor $g_m(n)$	0,43	-2,13
Estimate noise $x_{em}(n)$	5.038,00	-5.142,00
Clean signal $e_{m+1}(n)$	5.414,00	-4.532,00

Table 1. Upper and lower bounds of the variables

As the final implementation of the algorithm is going to be done with reconfigurable logic by using high level synthesis methodologies, then the limitation of the synthesis tools to integer data type must be specially taken into consideration, due to the implications in the algorithm computation accuracy.

3 Word-length adjust

By observing Table 1 a big dispersion on the variables bounds can be noticed. The reflection coefficients, the adaptive parameter, and the gain factor take values below one and they can not be represented as integer numbers. In the other hand, the auxiliary variable to calculate the gain factor takes a value that exceeds the range of representation of integer numbers (-2.147.483.648, + 2.147.483.647). A first approximation to the problem was to work with integer numbers and to scale the conflictive variables. This was done in a heuristic way by multiplying the variables with small values to make them significant and by dividing the variables close to the upper bound of integer's representation, undoing those changes later on. This approximation didn't give good results mainly because it was very difficult to adjust the scale parameters taking into consideration the algorithm recursively. A second approach was working with float-point arithmetic but considering the results as integer numbers. In this case, the upper bound didn't present any problem; the problem was in the variables values below one. The most critical variable in this case is the adaptive parameter $\alpha_{m+1}(n)$ due to its significance in the algorithm feedback adaptation. The evolution of these values for a typical case is shown in Table 2. It can be observed that the three more significant figures remain unchanged. And changes may be appreciating in the last significant figure in the position 10^{-6}. Thus to consider the influence of this last significant figure the adaptive parameter must be scaled by 10^6 or 2^{20} having in mind the hardware implementation of this scale factor.

…
0,999682
0,999785
0,999738
0,999726
0,999715
0,999578
0,999575
0,999666
0,99956
0,999708
…

Table 2. Adaptive variable values evolution

The reflection coefficients and the adaptation step were scaled in the same proportion than the adaptive parameter. The gain factor evolution requires to be scaled by 10^4 or 2^{14} doing the same analysis than in the case of the adaptive parameter.

NB	Upper bound	Lower bound	T1	T2	T3	T4	T5
15	16.383	-16.384		$g_m(n)$	$g_m(n)$		$g_m(n)$
16	32.767	-32.768				$g_m(n)$	
17	65.535	-65.536	$g_m(n)$				
20	524.287	-524.288					
21	1.048.575	-1.048.576	$\Psi^f{}_m(n)$ $\Psi^b{}_m(n)$	$\Psi^f{}_m(n)$ $\Psi^b{}_m(n)$	$\Psi^f{}_m(n)$ $\Psi^b{}_m(n)$	$\Psi^f{}_m(n)$ $\Psi^b{}_m(n)$	$\Psi^f{}_m(n)$ $\Psi^b{}_m(n)$
22	2.097.151	-2.097.152	$\alpha_m(n)$	$\alpha_m(n)$	$\alpha_m(n)$	$\alpha_m(n)$	$\alpha_m(n)$
23	4.194.303	-4.194.304					
24	8.388.607	-8.388.608					
25	16.777.215	-16.777.216					
26	33.554.431	-33.554.432					
27	67.108.863	-67.108.864					
28	134.217.727	-134.217.728					
29	268.435.455	-268.435.456					
30	536.870.911	-536.870.912					
31	1.073.741.823	-1.073.741.824	$r^f{}_m(n)$ $r^b{}_m(n)$			$r^f{}_m(n)$ $r^b{}_m(n)$	
32	2.147.483.647	-2.147.483.648	$r_0{}^b(n)$ $k_m(n)$				
33	4.294.967.295	-4.294.967.296	$d_m(n)$			$r_0{}^b(n)$ $k_m(n)$	
34	8.589.934.591	-8.589.934.592					
35	17.179.869.183	-17.179.869.184					
36	34.359.738.367	-34.359.738.368			$r^f{}_m(n)$ $r^b{}_m(n)$		
37	68.719.476.735	-68.719.476.736		$r^f{}_m(n)$ $r^b{}_m(n)$	$d_m(n)$		
38	137.438.953.471	-137.438.953.472			$r_0{}^b(n)$ $k_m(n)$		$r^f{}_m(n)$ $r^b{}_m(n)$
39	274.877.906.943	-274.877.906.944		$k_m(n)$ $d_m(n)$			$k_m(n)$ $d_m(n)$
40	549.755.813.887	-549.755.813.888		$r_0{}^b(n)$			$r_0{}^b(n)$
41	1.099.511.627.775	-1.099.511.627.776					
42	2.199.023.255.551	-2.199.023.255.552					

Table 3. Variables word-length demand for test files

Taking into consideration the values of the scale factor mentioned before, an exhaustive simulation study has been done in order to adjust the number of bits for each variable (NB). They has been adjusted according to the values of the lower and upper bounds obtained during the computation of the algorithm for all the commands enclosed in the proprietary data base. The commands all were grouped in five files. T1, T2 and T3 include the English command set and T4 and T5 the Spanish one. The criteria to validate the results consisted in the observation of the error between of the clean signal obtained in floating point and in floating point considering it as integer numbers including the scaling factor. Also the waveform of the clean signal result has been heard to subjectively evaluate the quality of the intelligibility of the commands. The results of the variables for the test files are shown in Table 3. Table 4 summarizes the optimal word-length for each variable and its associated scale factor.

	N. BITS	**Scale factor**
$r_a(n)^2$	31	NO
$r_0^f(n) = r_0^b(n)$	40	NO
$k_{m+1}(n)$	39	NO
$\Psi^f_m(n)$	21	* 2 ^ 20
$\Psi^b_m(n)$	21	* 2 ^ 20
$f_{m+1}(n)$	16	NO
$b_{m+1}(n)$	16	NO
$r^f_{m+1}(n)$	38	NO
$r^b_{m+1}(n)$	38	NO
$\alpha_m(n)$	22	* 2 ^ 20
$d_m(n)$	39	NO
$g_m(n)$	17	* 2 ^ 14
$x_{em}(n)$	16	NO
$e_{m+1}(n)$	16	NO

Table 4. Final word-lenght adjust

To have a visual idea about the results quality, the words down and eight corrupted by noise are shown in Figure 2a). The clean signal obtained after floating point computation is shown in Fig. 2b) and finally the clean signal obtained using the word-length and parameters from Table 4 are presented in Fig 2c).

4 Conclusions

We have presented a study for the word-length optimization of a noise canceller filter that can be used for speech enhancement. The optimization has been done taking as basis a set of spoken commands from a private data base. Initially, the upper and lower bounds of the variables implicated in the algorithm were determined in float point calculation. These initial results evidence that the variable more critical is the responsible for adapting the filter, $\alpha_m(n)$. This value serves as guide to scale the rest of the variables. To properly optimize the length of each individual variable an exhaustive simulation with all the spoken commands has been done. When comparing the clean trace obtained with float point arithmetic and with optimally adjusted word length we can conclude that the results are almost the same.

5. Acknowledgements

This research is being carried out under the project TIC2003-08756 of Programa Nacional de las Tecnologías de la Información y de las Comunicaciones from the Ministry of Science and Technology of Spain.

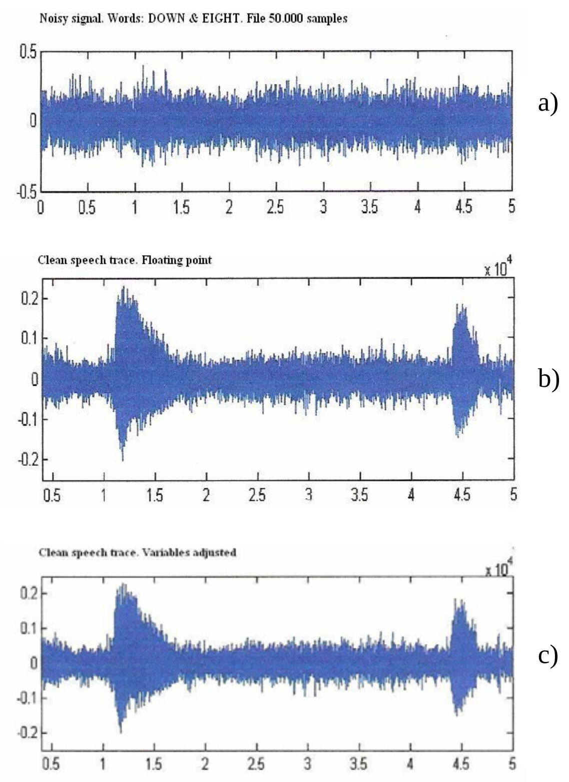


Fig. 2. Results for floating point arithmetic and optimized word sizes

References:

[1] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, 1996.

[2] S. Proakis, *Digital Communications*, McGraw Hill, 1989.

[3] R. Martinez, A. Alvarez, V. Nieto, V. Rodellar and P. Gomez, Implementation of an Adaptive Noise Canceller on the TSM320C31-50, *Proceedings of the 13th International Conference on Digital Signal Processing*, Vol. 1, 19XX, pp. 49-52.

[4] T. J. Todman, G. A. Constatinides, S. J. E. Wilton, O. Mencer, W. Luk and P. Y. K. Cheung. Reconfigurable computing: architectures and design methods. *IEE Proc-Comput Digit Tecn.* Vol. 152. No. 2 March 2005, pp. 193-207.

[5] G. A. Constantinides and G. J. Woeginger, The complexity of multiple word length assignment, *Appl. Math. Letters.* 2002. 15. (2), pp. 137 -140.

[6] Mark L. Chang and S. Hauck, Précis: A Usercentric Word-Length Optimization Tool, *IEEE Design & Test of Computers,* July-August 2005, pp. 349-361.

[7]http://tamarisco.datsi.fi.upm.es-Grupo-GTC-Grupo_Tecn_Comp_Esp.html