FPGA IMPLEMENTATION OF AN ADAPTIVE NOISE CANCELLER FOR ROBUST SPEECH ENHANCEMENT INTERFACES

V. Rodellar, A. Alvarez, C. Cónzalez, and P.Gómez

Facultad de Informática
Universidad Politécnica de Madrid
28660 Madrid - Spain
victoria@pino.datsi.fi.upm.es

E. Martínez

Escuela Universitaria de Informática
Universidad Politécnica de Madrid
Carretera de Valencia Km 7
28031 – Madrid -Spain

ABSTRACT

This paper describes the design and implementation results of an adaptive Noise Canceller useful for the construction of Robust Speech Enhancement Interfaces. The algorithm being used has very good performance for real time applications. Its main disadvantage is the requirement of calculating several operations of division, having a high computational cost. Besides that, the accuracy of the algorithm is critical in fixed-point representation due to the wide range of the upper and lower bounds of the variables implied in the algorithm. To solve this problem, the accuracy is studied and according to the results obtained a specific word-length has been adopted for each variable. The algorithm has been implemented for Altera and Xilinx FPGAs using high level synthesis tools. The results for a fixed format of 40 bits for all the variables and for a specific word-length for each variable are analyzed and discussed.

1. INTRODUCTION

The Adaptive Noise Canceller here described, is devoted to the construction of Robust Speech Enhancement Interfaces to be used in adverse environments with high levels of noise, whose spectral power characteristics are continuously changing. This previous pre-processing step has several applications as Robust Speech Recognition interfaces for their use in Command-Driven systems, such as advanced videogames, virtual reality applications; man-machine communications in computer aided manufacturing, intercommunication systems in factories, aircraft cockpit communications, etc. These environments are characterized by high noise levels (more than 95 dB) produced by costumer's chatting, ambient music, sirens, motors, etc.

There are different techniques to reduce noise levels [1] [2]. Adaptive Noise Cancellation for non-stationary environments in the time domain is an adequate technique offering a competitive performance. This filter computes the input information sample by sample which is very convenient for real time processing but its principal disadvantages are the need of calculating several multiplication and division operations and the wide range in the upper and lower bounds of the variables implied in the algorithm.

This work is funded by grant TIC2006-12887-C02-00 and projects HESPERIA, IBIOL (CCG06-UPM/INF-2) and SIBMATI (FIT-3600000-2007-32).

The implementation of these algorithms has been carried out traditionally with general-purpose DSP microprocessors using floating-point arithmetic. These implementations minimize round-off errors but tend to be limited in processing speed because they have usually available a single or a few processing units. In DSP microprocessors word-length implementation is defined by the hard-wired architecture but in reconfigurable computing the size of each variable may be customized in order to get the best trade-offs 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 over microprocessor implementations for specific applications [3].

The most difficult task in translating an algorithm written in MATLAB, for a general-purpose processor or DSP microprocessor into an algorithm optimized for custom logic, is the floating-point to fixed-point conversion due to accuracy problems [4]. The problem of word-length optimization is NP-hard [5] and different approaches have been adopted and tools developed for its treatment [4][6]. These tools are very helpful in the solution searching space but automatic translation from them can not be applied in a blind manner because the results are not good enough when accuracy is critical, such as the case presented in this paper. For this reason the automatic translation to fixed-point arithmetic has been disregarded in favor of C for fixed-point modeling.

2. NOISE CANCELLATION

The noise canceller under study has interesting characteristics for applications that demand speech enhancement techniques, some of them being mentioned next: No prior knowledge of the characteristics of the signal or noise is needed, it shows a very good behavior in highly non-stationary environments, preserves the quality of the speech, provides an immediate response, makes the further processing of the speech possible, and 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 recording scheme is based on a two-microphone structure as shown in Fig.1. One for noisy speech (primary, x(n)) and the other the noise in itself (reference, $r_a(n)$). The speech source is assumed to be well separated from the reference microphone to avoid crosstalk. The noise is estimated by a lattice filter (which is adapted by its estimation errors), its backward residuals being used to adapt the weights of a ladder filter in combination with the noise estimation generated. Clean speech is then obtained as the error output of the ladder filter.

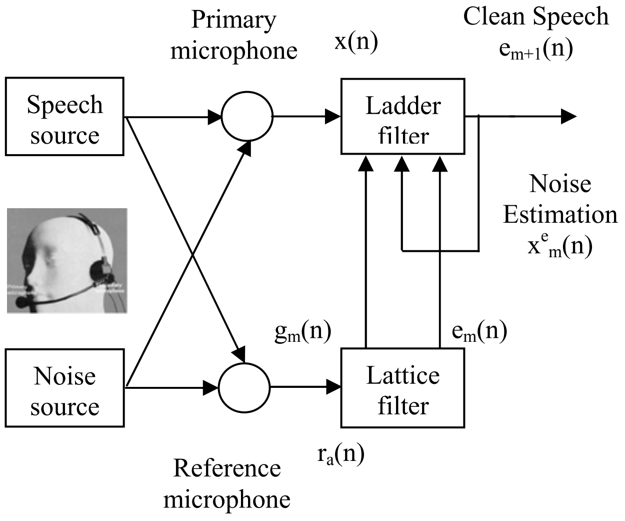


Fig. 1. General framework for noise removal

2.2. Computational analysis

The aspects which have more influence in the computational complexity of the algorithm are the sampling frequency (11025 Hz, enough for speech) and the number of stages of the lattice filter (14, required to support two microphones separated 20 cm), for these characteristics a cancellation average from 6 to 12 dB is obtained. The filter is recursive in the order of the lattice filter (m) and in time (n). The convergence rate of the algorithm is good and has a computational complexity of N. The stages for the calculation of the algorithm are three: initialization, lattice and ladder.

Variable initialization is shown in Table 1. The adaptive parameter α for samples and stages is set to one, and the adaptation step ω=0,9999. The initial values of the residual backward and forward errors for sample treatment are fixed to $\varepsilon = 5.10^8$, this value ensuring a high lock-up performance under acceptable stability conditions.

Table 1. Variable initialization

Adjust parameter	$\alpha_m(-1) = 1$	Samples (n)
Parcor coefficient	$k_m(-1) = 0$	
Residual errors	$r^b_m(-1) = r^f_m(0) = \varepsilon >> 0$	
Gain factor aux.	$d_m(-1) = 0$	
Adjust parameter	$\alpha_0(n) = 1$	Stages (m)
Estimated noise /clean signal $x^e_0(n+1) = e_0(n+1)$		
Forward and backward errors $f_0(n+1) = b_0(n+1) = r_a(n+1)$		
Forward and backward residual errors $r^f_0(n+1) = r^b_0(n+1) = \omega\, r^f_0(n) + \lvert r_a(n+1)\rvert^2$		

The lattice filter computation is the most expensive part of the noise removal algorithm, as show in Table 2. It begins with n = 0 and computes the updates for m = 0, 1, … N-2; with N = 14. The first step is to update the Parcor coefficient for the next stage and from it to calculate the reflection coefficients. The forward and backward errors, and the forward and backward residual errors are evaluated next. Finally, the adaptive parameter is estimated.

Table 2. Lattice calculation

Parcor coefficients $k_{m+1}(n) = \omega\, k_{m+1}(n-1) + \alpha_m(n-1)\, f_m(n)\, b_m(n-1)$	(1)
Reflection coefficients $\psi^f_{m+1} = -\dfrac{k_{m+1}(n)}{r^b_m(n-1)} \qquad \psi^b_{m+1} = -\dfrac{k_{m+1}(n)}{r^f_m(n-1)}$	(2)
Forward and backward errors $f_{m+1}(n+1) = f_m(n+1) + \psi^f_{m+1}(n)\, b_n(n)$ $b_{m+1}(n+1) = b_m(n) + \psi^b_{m+1}(n)\, f_n(n+1)$	(3)
Forward and backward residual errors $r^f_{m+1}(n) = r^f_m(n) - \dfrac{\lvert k_{m+1}(n)\rvert^2}{r^b_m(n-1)}$ $r^b_{m+1}(n) = r^b_m(n-1) - \dfrac{\lvert k_{m+1}(n)\rvert^2}{r^f_m(n)}$	(4)
Adjust parameter $\alpha_{m+1}(n) = \alpha_m(n) - \dfrac{\alpha^2_m(n)\lvert b_m(n)\rvert^2}{r^b_m(n)}$	(5)

The ladder filter calculates the gain factor, estimates the noise and produces the clean signal as a final result (see

Table 3). It begins with n = 0 and computes the updates for m = 0, 1 … N-1; with N = 14.

Table 3. Ladder calculation

Gain factor aux. $d_m(n) = \omega\, d_m(n-1) + \alpha_m(n)\, b_m(n)\, e_m(n)$	(6)
Gain factor $g_m(n) = -\frac{d_m(n)}{r_m^b(n)}$	(7)
Estimate noise $x^e_m(n) = x^e_{m-1}(n) + b_m(n)\, g_m(n)$	(8)
Clean signal $e_{m+1}(n+1) = e_m(n+1) + g_m(n)\, b_m(n+1)$	(9)

The algorithm demands 9 additions, 15 multiplications and 6 divisions per stage. The previous figures have to be multiplied by the number of stages to evaluate the number of operations required per sample. To have an idea of the computational cost in a real platform, we had implemented the algorithm in two different DSP microprocessors. The first implementation was done in a TSM320C1-50. The C31 is a 50 MHz, 32-bit floating point, with a computational pick power of 50 MFLOPS and 25 MIPS. These figures can be achieved executing two instructions in parallel in a 40 ns cycle basis. In this case, the algorithm demanded the 96% of the total available DSP computational power. The second implementation was done in a more modern and powerful microprocessor, the ADSP-21261-150. The ADSP is a 32-40 bit floating point microprocessor which features an instruction cycle time of 6,67 ns at 150 MHz. With its SIMS computational hardware it reaches 900 MFLOPS. The microprocessor overload was 5,65 in this case. The main reason for this costly computational load is due to the fact that both DSP microprocessors lack the division operation implemented in hardware, which is a common practice in most DSP microprocessors. If the computational cost of the algorithm is an obvious disadvantage, it presents the advantage of the very small amount of memory required, because the processing of the speech samples can be done as soon as they become available, therefore only a small array for speech input data is needed instead of large buffers.

3. WORD-LENGTH ESTIMATION

The algorithm was coded in ANSI-C and tested with a set of commands in English and Spanish. The command set was recorded from 32 speakers of both sexes (equally distributed) in an age from 20 to 45. The records were acquired under strong environmental noisy conditions (95-100 dB). The English command 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, detrás, 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 the worst case results in floating-point arithmetic are shown in Table 4. As the final implementation of the algorithm is to be carried out using reconfigurable logic by high level synthesis methodologies, the limitation of the synthesis tools in using integer data types must be specially taken into consideration, due to the implications in the algorithm computation accuracy [7].

Table 4. Upper and lower bounds of variables

	UPPER BOUND	**LOWER BOUND**
Reference channel sample $r_a(n)^2$	12.411.529,00	0,00
Initial residual forward 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 Coefficients $\Psi_{m+1}^f(n)$ $\Psi_{m+1}^b(n)$	0,56 0,56	-0,80 -0,80
Forward 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 $e_m(n)$	5.414,00	-4.532,00

In Table 4 a large dispersion on variable bounds can be observed. The reflection coefficients, the adaptive parameter, and the gain factor take values below one and they can not be represented as integer numbers. On the other hand, the auxiliary variable to calculate the gain factor takes a value that exceeds the range of representation for 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 carried out heuristically by multiplying the variables with small values to make them significant and by dividing the variables close to the upper bound of the integer representation, undoing those changes later on. This approximation didn't give good results mainly because to adjust the scale parameters was very difficult taking into consideration the recursive algorithm. A second approach was to work with floating-point arithmetic but considering the results as integer numbers. In this case, the upper bound didn't present any problem; the problem resided in variables taking 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 5. It can be observed that the three more significant figures remain unchanged. And changes may be appreciated in the last significant figure from 10^{-4} to 10^{-6} positions. 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. The reflection coefficients and the adaptation step were scaled in the same proportion than the adaptive parameter. The gain factor requires to be scaled by 10^4 or $2^{14}$from the same analysis than in the case of the adaptive parameter.

Table 5. Adaptive variable values evolution

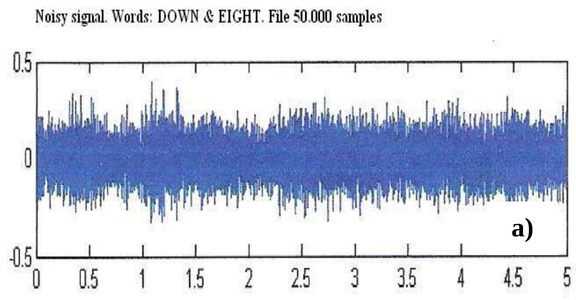
…
0,999**682**
0,999**785**
0,999**738**
0,999**726**
0,999**715**
0,999**578**
0,999**575**
0,999**666**
0,999**560**
0,999**708**
……

Taking into consideration the values of the scale factor mentioned before, an exhaustive simulation study has been carried out in order to adjust the number of bits for each variable (NB). This factor 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 mentioned before. The criterion to validate results consisted in estimating the errors between the clean signals obtained in floating point format considering them as integer numbers including the scaling factor. The clean waveform result has also been listened to subjectively evaluate the quality of command intelligibility. Table 6 summarizes the optimal word-length for each variable and its associated scale factor.

Table 6. Final word-length adjust

	NB	**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

To give an idea about the quality of results, 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 3 are presented in Fig 2c). When comparing the clean trace obtained with float point arithmetic and with optimally adjusted word length it can be concluded that the results are interchangeable.



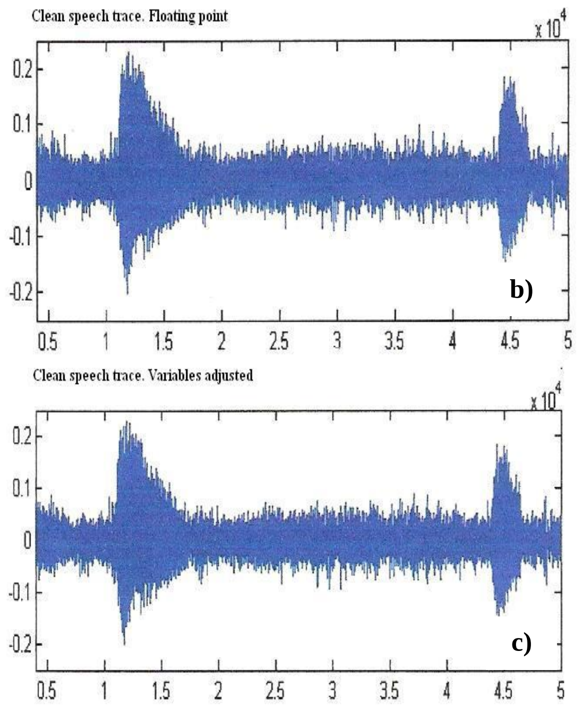


Fig. 2. Results for floating point arithmetic and optimized word sizes

4. IMPLEMENTATION RESULTS

The algorithm description has been carried out in ANSI C and automatically translated into VHDL by means of the CATAPULT-C tool from Mentor Graphics [8]. Later on, the VHDL resulting code was synthesized by the Quartus II from Altera and ISE from Xilinx tools. The results presented next correspond to two word-length cases. The first case, considers a fixed 40 bit word-data format for all the variables implied in the algorithm because it is the longest data format needed after optimization. The second one uses the word-length adjusted *ad hoc* after optimization for each variable according to Table 6. Table 7 shows the Altera results for the device EP2S15F484C3 from the Stratix family. Table 8 shows the Xilinx results for the device 4vsx25ff668 from Virtex II family. The first column of both tables indicates the parameter to evaluate: physical resources, frequency and power dissipation. The available amount of physical resources is indicated between parentheses. The results are indicated by the absolute total amount of physical resources used and its percentage from the total resources available in the device. The power estimation is obtained for an 80% of test coverage. Finally, the last column indicates the performance improvements in the word-length adjusted case.

The Altera results from Table 7 show that there is a reduction of the 37 % in the ALUTs, 41,5 % in registers and 32,3 in the bits of memory. But these reduction rates seem to imply a 68,7 % increment in the DSP blocks needed. The frequency increments a 5,1 %. And the significant number of a 30,8 % of reduction is achieved in the dynamic power.

Table 7. Synthesis results for Altera

Parameter	40 bits	Optimal	Gain
ALUTs (12.480)	10.846 (87 %)	6.831 (55 %)	+ 37 %
Registers (14.410)	7.670 (53 %)	4.487 (31 %)	+ 41,5 %
Memory bits (419.328)	10.240 (2,44 %)	6.928 (1,65 %)	+ 32,3 %
DSP blocks 9 bits (96)	32 (33 %)	54 (56 %)	- 68,7 %
Frequency Max	114,8 MHz.	120.7 MHz.	+ 5,1 %
Dynamical Power	201.71 mW	139,62 mW	+ 30,8 %
Statical Power	359,88 mW	345,79 mW	+ 4 %

Concerning the results for Xilinx shown in Table 8, a similar saving percentage is found for function generators, CLB slices and Dff, this being a 28,1 % for RAM blocks. The DSP blocks show the same tendency than the Altera case increasing a 100 %. Not significant differences for frequency and power dissipation were observed.

Table 8. Synthesis results from Xilinx

Parameter	40 bits	Optimal	Gain
Function generators (20.480)	7.640 (37 %)	4.846 (24 %)	+ 36,6 %
CLB slices (10.240)	3.937 (38 %)	2.529 (25 %)	+ 35,8 %
Dff or latches (21.440)	7.873 (37 %)	5.058 (24 %)	+ 35,8 %
RAM blocks (128)	32 (25 %)	23 (18 %)	+ 28,1 %
DSP blocks 48 bits (128)	12 (9 %)	24 (18 %)	- 100 %
Max Frequency	91, 02 MHz.	91,30 MHz.	+ 0,3 %
Dynamical Power	160,48 mW	159,8 mW	+ 0,12 %
Statical Power	280,1 mW	279,3 mW	+ 0,28 %

The Altera and Xilinx results can't be strictly compared because the FPGAs being used in the implementations have different characteristics and the synthesis, optimization and mapping modules of the tools may not use the same strategies. Physical resource demand in the optimal case shows the same tendency for both tools, a reduction for the generation of combinational and memory parts and an increment in the number of DSP blocks. This increment is natural as the number of bits decreases because the tool can map functionality more easily to DSP units according to its number of bits, 9 for Altera and 48 for Xilinx. The maximum clock frequency and dynamical power show a better behaviour for Altera than for Xilinx.

5. CONCLUSION

A study on word-length optimization of a speech enhancement noise-cancelling filter has been presented. The optimization has been carried out taking a set of spoken commands from a data base as a reference. 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 most critical variable is the filter adaptation step $\alpha_m(n)$. The procedure used in the case of this variable serves as a model 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 carried out. When comparing the clean trace produced with float-point arithmetic using an optimally adjusted word length it can be concluded that the results are comparable. Finally, the longest data-format after optimization was implemented for all the variables and contrasted with the data format optimized for each one of them. The quality of the results shows a high dependency on the tools and implementation devices when design methodologies based on high level synthesis are used.

6. REFERENCES

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

[2] S. Proakis, “Digital Commnunications,” *McGraw Hill*, 1989.

[3] T. J. Todman, G. A. Constantinides, 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, pp. 193–207, March 2005.

[4] Thomas Hill, “AccelDSP Synthesis Tool Floating-Point to Fixed-Point Conversion of MATLAB Algorithm Targeting FPGAs,” *Xilinx Whitepaper (WP239-V1.0)*, April. 2006.

[5] G. A. Constantinides and G. J. Woeginger, “The Complexity of multiple word length assignement,” *Applied Mathematics Letters*, vol. 15, no. 2, pp. 137-140, 2002.

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

[7] K. Kung and W. Sung, “Combined word-length optimization and high-level synthesis of digital processing systems,” *IEEE Trans. on Computer Aided Design*, vol. 20, no. 8, pp. 931-930, 2001

[8] S. McCloud, “Catapult-C, Synthesis-based design flow: speeding implementation and increasing flexibility,” *White paper. Mentor Graphics*, 2004.