Published in 7th European Signal Processing Conference EUSIPCO, 1661-1664, 1994
which should be used for any reference to this work

Study of a VLSI Implementation of a Noise Reduction Algorithm for Digital Hearing Aids

Sara GRASSI, Alexandre HEUBI, Michael ANSORGE, Fausto PELLANDINI

Institute of Microtechnology, University of Neuchâtel, Rue de Tivoli 28, CH-2003 Neuchâtel-Serrières, Switzerland,
Phone: ++41 38 301 479, FAX: ++41 38 301 845, E-Mail: grassi@imt.unine.ch

Abstract. A methodology for meeting the tight constraints in the physical realization of functional blocks for digital hearing aids was applied to the implementation of a noise reduction system based on lattice structures. This methodology fully exploits the flexibility of custom VLSI design through a good interrelation among all the steps of the design. The emphasis was placed in the study of the fixed point quantization effects to find the minimum number of bits and scaling required at every point of the algorithm. Based on these results, an estimation of the power consumption and required silicon area was done in the case of an implementation using a low power VLSI architecture.

1. Introduction

Analog electroacoustic hearing aids are the primary treatment for most hearing impaired people. They contain the basic functions of amplification, frequency shaping, and limiting of the output signal. Digital hearing aids promise many advantages over conventional analog hearing aids, among them the increased precision and programmability of DSP techniques and the possibility of adding new functions such as noise reduction, spectral sharpening and feedback cancellation [1]. On the other hand their physical implementation is characterized by very tight requirements in chip size, voltage supply and power consumption, which are very difficult to fulfil given the complexity and number of functions to be implemented together with the real time requirement and large dynamic range of the input signals [2].

Several algorithms have been proposed to perform the functions of frequency shaping [2], feedback cancellation [3] and noise reduction [4]. However, the ultimate problem remains the feasibility of a physical implementation of these algorithms, in particular for meeting the constraints of chip size and power consumption. This could be achieved by a careful optimization that ranges from algorithm level, through system and circuit architecture to layout and design of the cell library. The key points in this optimization are among others the choice of a fixed point arithmetic unit, the optimization of the algorithm minimizing the number of operations and the number of bits required at every point of the algorithm, and a careful match between algorithms and architecture.

2. The Algorithms

In the algorithms proposed in [4], spectral sharpening is used for noise reduction and compensation of the reduced frequency selectivity encountered among many hearing impaired people. The core of both algorithms contains a Gradient Adaptive Lattice Linear Predictor (GAL) and two, IIR and FIR, modified lattice filters (Synthesis and Analysis filters). These algorithms are particularly suitable for a fixed point VLSI implementation due to the good quantization properties of lattice filters and their modular structure, local interconnections, and rhythmic data flow.

A block diagram of the noise reduction algorithm is provided in Figure 1. The GAL Predictor extracts spectral information from the input signal at every sampling instant. This spectral information is encoded in the Parcor coefficients Ki and used by the Analysis and Synthesis filters to perform a signal dependent filtering of the input.

The first cell of the GAL Predictor and the Analysis and Synthesis filters can be observed in Figures 2, 3 and 4.

To obtain the speech enhancement system, the high pass filter is placed at the input of the GAL and a gain control unit is added at the output of the Synthesis filter [4].

Only the noise reduction algorithm was studied and implemented and all the optimization effort was done in the implementation of the core of this algorithm.

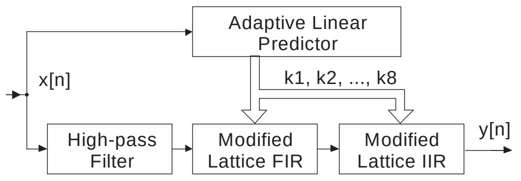

Figure 1. Spectral Sharpening for Noise Reduction

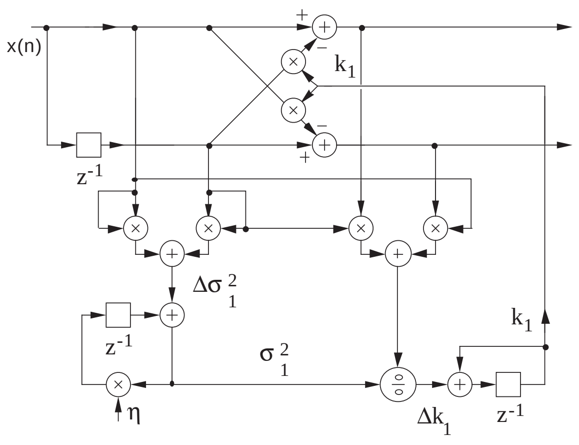

Figure 2. GAL Predictor (first cell)

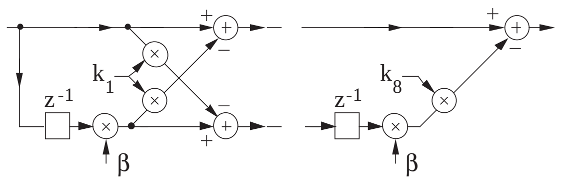

Figure 3. Analysis Filter

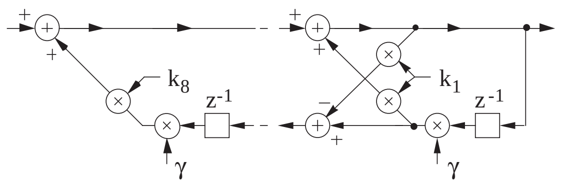

Figure 4. Synthesis Filter

3. High Level Simulation

A high level simulation of the noise reduction algorithm was done first to determine the optimal parameters (γ, β and η) and then to use this implementation as reference system for further simplification and optimization.

All the functional blocks were coded in C language as Matlab functions. Other functions were written to allow high quality DA conversion, loading speech files in a data vector and running programs on a DSP56001 as Matlab functions. Within Matlab, the algorithms can be tried with different parameters and the signals involved can be analyzed, displayed and played to have an immediate feedback of the effect of any change in the system.

A set of 15 files recorded at 8 kHz, 16 bit precision, and the 6300 files from the TIMIT database [5] converted to 8 kHz were used as input. These files were scaled to the range [–1, +1) and their precision reduced to simulate a 12 bit AD converter. The set of 15 files was used to define the first choice of the number of bits of each quantizer. This choice was tuned by processing some (usually 100) files chosen at random from the TIMIT database, and systematic testing on all the files available was done in some cases such as overflow search.

3.1 Measure of Performance

The high level implementation of the algorithm using double precision floating point arithmetic is used as reference system.

To measure the performance of a modified system, its output is compared with the output of the reference system using SNR measures. In this context, the output of the reference system is the "non-noisy" signal and the "noise" is the difference between the output of the modified system and the output of the reference system.

An SNR of 15 dB or more means that the transfer function of the Analysis-synthesis filter of both systems do not differ significantly and the outputs of both systems cannot be distinguished in listening tests.

3.2 Real Time Implementation

A first implementation of the noise reduction algorithm was done on a DSP56001 for checking real time feasibility and identifying which functional blocks are more time-consuming in view of a possible simplification. These observations are summarized in Table 1 for a system of order 8 with a sampling frequency of 8 kHz and clock frequency of 20 MHz. A uniform precision of 24 bit was used throughout the whole system. The serious degradations observed in the performance motivated a more detailed study of the quantization effects.

The computational load of the division was 63% of the time available, therefore a particular effort will be done in the simplification of this operation.

Function	#Cycles	%	Remarks
GAL	682	82	Division:63%
Synthesis	76	9	
Analysis	78	9	
Total	836	100	33% of time available

Table 1. Computational load for first DSP56001 implementation

4. Simplified Division

There is no divide operation in the low power architecture previewed for the final implementation. Also, since the division is a less frequent operation (1 division per 12 multiplications), it is not efficient to implement a full division as a special unit.

GAL algorithms where the division is replaced with a multiplication by a small constant do not yield the fast convergence independently of the input signal level required for good speech enhancement results. A reasonable compromise is the approximation of the divisor by a power of two.

The measured SNR between the output of a system with simplified division (but otherwise no other change with respect to the reference system) and the reference system was more than 20 dB. This result shows that the simplified division is feasible.

5. Quantization Effects

Before a VLSI implementation is actually made, a simulation of the system in operating conditions is mandatory, especially realising that the needed word-length determines the size and power consumption of the final implementation.

The effects of a two's complement fixed point arithmetic were included in the high level simulations by placing quantizer operators at different points of the algorithms. Each quantizer is completely defined by its rounding policy and its number of integer bits (NI) and number of fractional bits (NF). An example of the C code for a quantizer with sign-magnitude truncation and its placing in the Analysis filter is given in the Appendix. The place of the quantizers as well as their rounding policy is determined at the moment of compilation, whereas the number of bits of each quantizer is chosen at run time together with other parameters of the algorithms and the input signal. The value of an overflow counter for each quantizer is returned at the end of the simulation.

5.1 Parameters of the System

The interesting values of γ and β are in the ranges (0.90, 0.99) and (0.10, 0.50) respectively. These parameters were quantized to 12 bit. The effect of this quantization is negligible and was included in the reference system for the remaining of the study. All the simulations were done using values in these ranges and the results obtained hold under these conditions. The system was more sensitive to modifications when $\gamma = 0.99$.

When η is in the range (0.98, 0.985) the GAL algorithm performed well for all input signals. A value of 0.9805 yields to an efficient implementation as explained in section 5.3.

5.2 The Optimized System

The target architecture for the final VLSI implementation is a low power architecture described in [6,7]. The placing of the quantizers and their rounding policy was influenced by the characteristics of the arithmetic unit available in the target VLSI architecture. The place of the quantizers can be observed in Figures 5, 6 and 7. The chosen rounding policy is sign-magnitude truncation.

The number of bits of each quantizer is also influenced to some extent by the final target architecture: the word-length of the multiplier must be a multiple of four and the word-length of the accumulator is twice the word-length of the multiplier.

The chosen values for the number of bits of the quantizers is given in Table 2. The measured SNR between the output of the quantized system and the reference system was more than 20 dB.

A system with both quantizers and simplified division gives more than 15 dB of SNR when compared to the reference system.

# bits	q1	q2	q3	q4	q5
NI	2	2	7	1	7
NF	14	25	25	15	9

Table 2. Number of bits for each quantizer

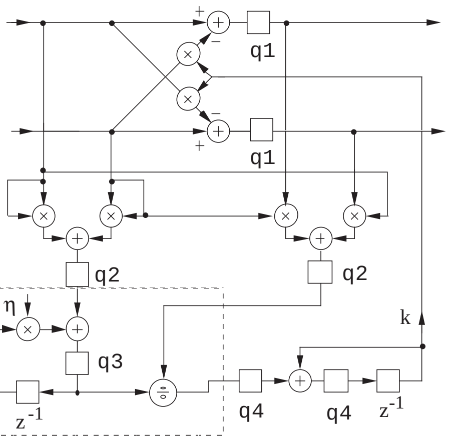

Figure 5. Quantizers in the GAL Predictor

5.3 Implications for the VLSI Implementation

From the point of view of the word-length requirements we could divide the circuit into two sections. These two sections will be implemented using two different hardware units.

The first section corresponds to the dashed box shown in Figure 5 and contains the power estimation recursion and the division. This section requires higher dynamic range (although not necessarily higher precision). It contains the two most critical operations of the system which are the long-word (32 bit) multiplication by η and the long-word computational expensive division. Using a fixed choice of $\eta = 0.9805 = 1 - 2^{-6} - 2^{-8}$ and the simplified division yields to an efficient implementation on a dedicated unit that contains a 32 bit adder, and the logic for the approximation of the power estimation by a power of two. The multiply accumulate is substituted by three additions and two hard-wired shifts.

The second section corresponds to the rest of the GAL algorithm together with the Analysis and Synthesis filters. The word-length requirements of this section are met by 16 bit multipliers with 32 bit adder-accumulators.

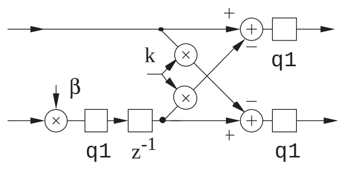

Figure 6. Quantizers in the Analysis filter

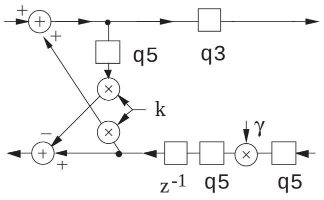

Figure 7. Quantizers in the Synthesis filter

5.4 Implications for the DSP56001 Implementation

In Table 2 is observed that, at some points of the algorithm, the minimum word-length required exceeds the uniform 24 bit word-length used in the first DSP56001 implementation. Seven steps of normalization and denormalization were added at these points to extend the dynamic range of the temporal registers of the DSP56001. The computational load of the GAL was almost doubled as observed in Table 3, but the real time constraint is still met. The Analysis and Synthesis blocks were left unchanged but a 6-bit scaling is included at the input of the Synthesis filter to compensate the amplification introduced by this block. Each functional block was coded separately on DSP56001 assembler code and simulated in C language (including the effects of the arithmetic used in the DSP56001). Both implementations gave exactly the same results allowing the verification of the accurateness of the simulations.

The output of this second DSP56001 implementation is virtually equal to the output of the reference system, with measured SNR of more than 80 dB. Systematic search on the TIMIT database showed no overflows. From a practical point of view, the DSP implementation was a good trial of the results obtained in the study of the quantization effects.

Function	#Cycles	%	Remarks
Decorrelator	1530	91	Division:31%
Synthesis	76	4.5	
Analysis	78	4.5	
Total	1684	100	67% of time available

Table 3. Computational load for second DSP56001 implementation

6. VLSI Implementation

An implementation of the optimized system was done using the target low power VLSI architecture which is described in [6]. The architecture takes advantage of the regularity of the algorithm to simplify the scheduling and the hardware implementation. The processor architecture and modules are organized in a way to limit the overall data transfer to the strict minimum, local data traffic being preferred versus global traffic.

Larger memories are split into a set of smaller memories where a single one is activated at a time. A sequential dynamic memory is used for storing the Parcor coefficients, another for storing the state variables of the Analysis and Synthesis filters, and two remaining ones for the variables of the Linear Predictor.

The arithmetic unit is a serial-parallel unit optimized for performing scalar products. The number of relevant partial products occurring in the multiplications is reduced at least by a factor of two using Booth's recoding scheme. Two arithmetic units of this kind are used to achieve a sufficient computational throughput. This for the implementation of the Analysis, Synthesis and the portion of the GAL outside the dashed box in Figure 5.

The dedicated unit for the implementation of the portion of the GAL inside the dashed box in Figure 5 was realized separately using a low power standard cell library.

The scheduling was hierarchically organized to limit the processing rate of each module to the strict minimum using an adapted version of the "TABU search" optimization technique, which is particularly suitable for the scheduling of DSP algorithms.

The details of this implementation are given in a companion paper [7]. The resulting silicon area was approximately 4 mm^2 using VLSI Technology's CMN12 1.2 µm CMOS process. The estimated power consumption was 0.65 mW at 2V.

7. Conclusions and Future Work

A methodology for meeting the tight constraints in the physical realization of functional blocks for digital hearing aids was presented. This was applied to the particular case of a noise reduction algorithm based on lattice filters. The emphasis was placed in the study of the quantization effects.

The interrelation between the target VLSI architecture and the algorithmic level plays an essential role in the optimization process. The resources available in the architecture influenced some choices at the algorithmic level, whereas some constraints and particular needs of the algorithm forced some choices in the VLSI implementation.

We proposed a simplification of the Gradient Adaptive Lattice algorithm which improves the efficiency in the implementation while keeping good convergence properties. Further study should be done in this topic.

From a practical point of view, we consider that the approach using real input signals is the more appropriate for the characterization of the systems in final operating conditions. The implementation on a commercial fixed point DSP is an important intermediate step which allows real time evaluation and gives further information on the behaviour of the final implementation.

Given that Lattice filters and Lattice Linear Predictors are used in many areas of speech processing, the results obtained can be used in other applications where the needs of reduced size and power consumption plays and important role such as portable devices for telecommunications.

Acknowledgements

This work was supported by the Swiss Found for Research on Microtechnique (FSRM) under the project FSRM 91/07 in collaboration with CSEM SA, Neuchâtel, Switzerland.

References

[1] **Working-group on Communication and Aids for the Hearing-impaired people (1991):** "Speech-perception aids for hearing-impaired people: Current status and needed research", J. Acoust. Soc. Am., 90/2, pp. 637-685.

[2] **Lunner T. and Hellgren J. (1991):** "A Digital Filterbank Hearing Aid Design, Implementation and Evaluation", Proc. ICASSP'91, Toronto, Canada, May 1991, pp. 3661-3664.

[3] **Kates J. M. (1990):** "Feedback Cancellation in Hearing Aids,", Proc. ICASSP '90, Albuquerque, New Mexico, USA, April 1990, pp. 1125-1128.

[4] **Schaub A. and Straub P. (1991):** "Spectral Sharpening for Speech Enhancement / Noise Reduction", Proc. ICASSP '91, Toronto, Canada, May 1991, pp. 993-996.

[5] **J. Garofolo et al. (1993):** Darpa TIMIT, Acoustic-Phonetic Continuous Speech Corpus CD-ROM, National Institute of Standards and Technology, NISTIR 493.

[6] **A. Heubi, M. Ansorge, F. Pellandini (1993):** "Low power VLSI Architecture for Digital Signal Processing", Proc. GRETSI'93, Juan-les-Pins, France, Sept. 1993, pp. 3661-3664, (in French).

[7] **A. Heubi, S. Grassi, M. Ansorge, F. Pellandini (1994):** "A Low Power VLSI Architecture with an Application to Adaptive Algorithms for Digital Hearing Aids", EUSIPCO'94, Edinburgh, UK, Sept. 13-16, 1994.

[8] **Rutter M. J. (1985):** "An Adaptive Lattice Filter" in *VLSI signal processing: a bit-serial approach*, Denyer P. and Renshaw D. (Eds.), Addison-Wesley, 1985.

Appendix

Example of macros and functions used to simulate the finite precision quantizers and portion of C code that implement the Analysis filter with the quantizers as shown in Figure 6:

```
/*define rounding policies */
#define SMTRNK(a) (a)<0 ? ceil(a) : floor(a)
double qnt( double a, int prec)
{
if (a>maxv[prec])
   {a=maxv[prec];ovv[prec]++;}
else if (a<minv[prec])
   {a=minv[prec];ovv[prec]++;}
else a=(SMTRNK(conv[prec] * a))/conv[prec];
return a;
}

double ana_lat(double   x,double   *k,unsigned
order, double *del, double bb)
{  double f[50], b[50];
   int i,
   f[0] = x; b[0] = x;
   for (i=1;i<=order;i++) {
      f[i] = f[i-1] - k[i-1]*del[i-1];
      b[i] = del[i-1] - k[i-1]*f[i-1];
      f[i] = qnt(f[i],1);
      b[i] = qnt(b[i],1);
      del[i] = bb*b[i-1];
      del[i] = qnt(del[i],1);}
   return f[order];
}
```