

DESIGN AND IMPLEMENTATION OF A LINEAR-PHASE EQUALIZER IN DIGITAL AUDIO SIGNAL PROCESSING

C.H. Slump, C.G.M. van Asma, J.K.P. Barel, W.J.A. Brunink,
F.B. Drenth, J.V. Pol, D. Schouten, M.M. Samsom, O.E. Herrmann

Department of Electrical Engineering
University of Twente, P.O. Box 217
7500 AE Enschede, the Netherlands

Abstract - This contribution presents the four phases of a project aiming at the realization in VLSI of a digital audio equalizer with a linear phase characteristic. The first step includes the identification of the system requirements, based on experience and (psycho-acoustical) literature. Secondly, the signal processing algorithms constituting the global design of the equalizer are computer simulated. The third step includes the realization of the equalizer design using one or more programmable DSP's. In order to minimize the number of DSP chips necessary for the realization, this step requires the optimization of the structure and mapping of the algorithm on the resources of the DSP. The number of processor cycles is crucial in this optimization. The purpose of the resulting prototype is to test and to validate in a digital audio environment the specification generated in the first step. The programmability of the DSP's allows for specification changes at this stage of the project. The fourth step is the VLSI implementation of the validated algorithm of the previous phase. For this purpose the structure of the algorithm is optimized in order to take full advantage of the silicon resources. Speed and required area are the crucial parameters in this optimization. The final step includes the testing of the completed chips together with a parallel designed and realized PCB in a digital audio environment. The presentation will emphasize the algorithmic and design considerations together with the results.

INTRODUCTION

In the field of audio recording or reproduction, signal analyzers are widely used *e.g.* to overcome shortcomings of analog natural sources or to modify the sound frequency characteristic in a more preferred shape. Equalizers, therefore, do consist of a set of bandpass filters. In each band the gain can be set by the user. The bandpass filters in many analog equalizers are only of second order. In general, analog equalizers have the disadvantages of the

dependent gain control between adjacent bands, due to the relative large overlap in frequency band and small stopband attenuation, and a large ripple in the passband. Furthermore analog equalizers inevitably add phase distortion especially in the enhanced resp. attenuated parts of the spectrum. For monaural sources this distortion is of no importance. However, in a multi-point acoustic field, the stereo image can be severely distorted due to the phase nonlinearities.

The location of a source in a soundfield by the human auditory system is, at least for the lower frequency range, guided by the phase difference perceived between the two ears. The topic of the perceptivity of phase distortion, however, continues to generate much controversy and we will stay away from it.

These considerations lead us to the idea to apply digital FIR bandpass filters with a strict linear phase characteristic for the equalizer realization. In order to test the performance and the user preference, a project started to design and realize such a system. As linear phase FIR filters have symmetrical impulse responses, we anticipate in this approach the pre-echos of impulsive sound dynamics. Whether this is worse than phase distortion is an interesting question in the area of psycho - acoustics but outside our scope. This paper emphasizes the design and realization of the digital linear-phase equalizer.

The final step of this project includes the testing of the completed chips together with a parallel designed and realized PCB in a digital audio environment. However, this point has not been reached yet and is outside the scope of this paper.

SYSTEM SPECIFICATION

First we specify the input signal to the equalizer system. If we adopt the CD digital audio quality standard we have an audio signal sampled at a rate of 44.1 kHz and digitized in 16 bits. The output of the equalizer follows this standard as well. The equalizer can be realized by a set of N parallel bandpass filters with a gain factor for each band which can be set independently by the user. The system should have a linear overall phase characteristic and a flat amplitude characteristic if the gain factors of all bands are set equal.

The ability of the human audio perceptive system to discriminate between frequencies is diminishing proportional with absolute frequency. Therefore, for a perceptive uniform audio resolution, the filter bandwidths can be proportional with frequency too. A number of $N = 10$ octave bands covers the range of audible frequencies 30 Hz - 20 kHz adequately, and is therefore a good starting point for the design of the equalizer. Consecutive octave bands double in bandwidth. Also the transition width of the bandpass filters can be increased proportional to frequency. The octave bands are defined in Table 1. The Table shows that the overlapping adjacent slopes of two consecutive filters have to be realized in such a way that with equal gain factors the overall amplitude characteristic becomes flat. The dynamic range of 16 bits corresponds to an attenuation of 90 dB in the stopband. Psycho - acoustical

| # | f_{sl} (Hz) | f_{pl} (Hz) | f_{pu} (Hz) | f_{su} (Hz) | Δf (Hz) |
|----|---------------|---------------|---------------|---------------|-----------------|
| 1 | 0 | 10 | 30 | 40 | 20 |
| 2 | 30 | 40 | 80 | 100 | 40 |
| 3 | 80 | 100 | 180 | 220 | 80 |
| 4 | 180 | 220 | 380 | 460 | 160 |
| 5 | 380 | 460 | 780 | 940 | 320 |
| 6 | 780 | 940 | 1580 | 1900 | 640 |
| 7 | 1580 | 1900 | 3180 | 3820 | 1280 |
| 8 | 3180 | 3820 | 6380 | 7660 | 2560 |
| 9 | 6380 | 7660 | 12780 | 15340 | 5120 |
| 10 | 12780 | 15340 | - | - | - |

Table 1: The specification of the ten octave bands, note that band 10 is a highpass filter!

literature reports audible effects 40 dB below the main signal. Therefore we set the requirement on the overall passband ripple to ± 0.1 dB. The system requirements of the equalizer system are hereby specified.

EQUALIZER ALGORITHM

In principle, the requested bandpass filters can be realized using the direct form realization of Finite Impulse Response (FIR) filters [1]. We have *e.g.* for the i^{th} band:

$$y_i(k) = \sum_{l=0}^{L_i-1} h_i(l) u(k-l) \quad k = 0, 1, 2, \dots \quad (1)$$

The impulse response of the FIR filter is given by the set coefficients $h_i(l)$, $l = 0, 1, \dots, L_i$. A symmetric impulse response ensures a linear phase. In order to obtain some insight in the task we have set ourselves we designed the 10 bandpass filters of Table 1 as equiripple filters with the (worstcase) specification of 90 dB stopband attenuation and ± 0.1 dB passband ripple for each bandpass filter. For this design we have used the filter design software from the Signal Processing Workstation (SPW) [2]. The number of coefficients L_i necessary for the realization in this "brute force" approach is summarized in Table 2.

For this realization the total number of filtercoefficients is 37277. In order to be able to test this realization against the system specification, we therefore need a processing power of 1.65 Giga Operations per Second (GOPS) and at least 76 DSP chips, based upon the assumption, that a multiplication and accumulate instruction is performed in two clockcycles and with a clockcycle of 44.1 Mhz.

| filterband i | filterlength L_i |
|----------------|--------------------|
| 1 | 14928 |
| 2 | 11196 |
| 3 | 5600 |
| 4 | 2800 |
| 5 | 1400 |
| 6 | 700 |
| 7 | 350 |
| 8 | 175 |
| 9 | 88 |
| 10 | 44 |

Table 2: The obtained filterlength L_i with the "brute force" approach.

Inspired by [3] we have chosen the multi - rate signal processing structure of Fig. 1, which appeared promising for further reduction of computations [4]. By applying decimation techniques the number of multiplications per second can be reduced considerably. Of course the original frequency contents must be restored by interpolation. In order to reduce the sample rate with a factor M we first have to filter the signal to remove the frequency components above half the reduced sample rate. Alias distortion is prevented in this way. Of every M samples only one sample is processed, the others are discarded. Because of the less stringent requirements to these anti-alias filters, it is often advantageous to perform the decimation (interpolation) in more than one stage. The interpolation process inserts $M - 1$ zeros between consecutive samples. The interpolation filter restores the signal.

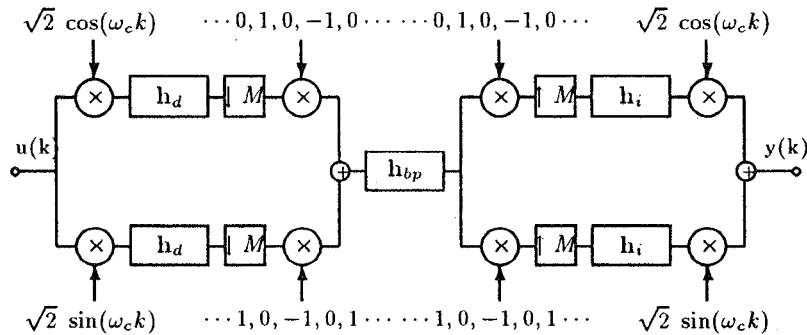


Figure 1: Block diagram bandpass filter based on single sideband modulation, ω_c is the center frequency of the passband.

DSP IMPLEMENTATION

The implementation on the DSP can be divided into two parts. After a discussion of the implementation of a single band we turn to the multi-band case. For a single band we choose for an input buffer size of twice the decimation factor M of the first stage. Once M samples have been read, M new output samples can be computed. The algorithm includes modulation, filtering, demodulation etc. as indicated in the block diagram in Fig. 1. The modulations/demodulations at the input and at the output of the block diagram are performed with cosine and sine tables because the DSP cannot execute a simple $\cos(x)$ instruction. The length of such a table depends on the modulation frequency (*i.e.* the ratio of the sampling frequency and the center frequency of the pertinent bandpass filter). Applying octave bands we need for lower frequencies more coefficients in order to obtain a cyclic table. The modulation/demodulation in the middle of the diagram are simple frequency shifts of $\frac{1}{2}\pi$, so these tables become very short having the coefficients 0,1,0,-1.

For the filter operations we apply tables with filter coefficients using a pointer to the current coefficient. The in - and output of such a filter will also be stored in two tables using pointers, one table for the real components and one table for the imaginary components. Handling these pointers in the proper way the samples are multiplied with the pertinent coefficients and accumulated. Using this method each filter will be calculated. The decimation is performed by removing (not using) $M - 1$ of the M samples with decimation factor M . The interpolation output must be multiplied with the interpolation factor to adjust the energy of the signal. The single band implementation is illustrated in Fig. 2.

If we want to expand this single-band system to a multi-band system we have to handle some problems. For the input and output we should take a buffer with a size of at least twice the maximum decimation factor. However in particular cases this size might prove too short and samples might be processed at the moment they have been overwritten. (For example if the two biggest decimation factors don't differ much). For this reason an input/output buffer should be taken with a size of twice the maximum decimation factor plus the second largest decimation factor. A pointer giving the current position in the main input/output buffer should be retained for each band. Each band will be processed according to a scheduling table containing a list that points to the band to be processed.

The necessary causality of the filters introduces a signal delay when processing the individual filters. The total delay of each bandfilter is the sum of the delays in each single filter. Since each band has different filters and different decimation factors, the total delays of each band will be different. For this reason all filters should be delayed in such a way that the impulse response of each individual filter becomes symmetrical around the same point in time. The practical delay is realized by means of a memory buffer, hence the implementation of a large delay will take a large amount of memory. A better way of realizing the same delay is by making advantage of the lower

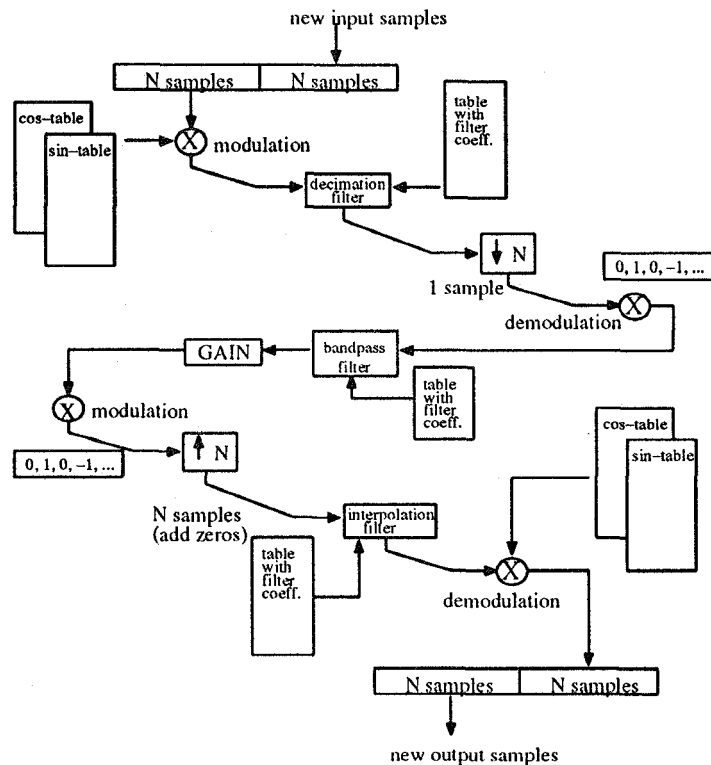


Figure 2: Single band DSP data flow structure

sampling frequency in the system, since a delay of one sample at the lower rate corresponds to M delays at the higher frequency.

At this point the DSP can be programmed. The main problem of the programming is of course situated in the available processing time of the DSP for a sample. Using the Motorola DSP 56001 the number of available clock cycles between two consecutive input samples will be ± 460 . In realizing the entire system this does not seem much, however, by using efficient (pre)decimation techniques this amount of cycles can be raised to M times the original number of cycles depending on which (pre)decimation factor has been used.

VLSI IMPLEMENTATION

Besides the DSP implementation of the developed algorithm we wanted to survey the possibility of realizing the algorithm in dedicated hardware. The (V)LSI implementation of an algorithm in hardware can give some profit in terms of speed and area compared to the use of DSP's if one admits less flexibility. One can optimize towards speed and/or area while preserving

the necessary functionality. In contrast with the DSP realization, where the number of cycles is the main cost factor (because of the sequential behaviour of the processor), one can make fully use of parallelism and pipelining when designing dedicated hardware. In this part shortly two prototype designs are discussed of which one makes use of extended pipelining, in this way minimizing the number of functional blocks, whereas the second design makes use of several functional parts which work in parallel, each executing part of the algorithm.

To assure a correct design, the algorithm is firstly described and simulated using the Modeling and Design Language (MoDL) [5]. Starting from the behavioral description, the design can be worked out via an initial implementation towards a more detailed hardware description. Using this method, every design step can be verified. From this hardware description the datapath is entered in the Mentor Graphics' Idea Station (schematic entry) [6] using a standard cell library. The controller part of the design is generated using a logic synthesis package [7] with interface to the Mentor Graphics' Idea Station. The layout is generated using Mentor Graphics' Cell Station (Placement & routing). Processing is done via Eurochip using the chosen standard cell technology (ES2 1.5 μm CMOS or MIETEC 2.4 μm CMOS). After fabrication the design will be tested with an ASIC tester using the determined test patterns, as well as an in-circuit test in an audio test environment.

Looking at the algorithm we can distinguish several functions that have to be performed. One needs 14 convolutions, which have to do as many multiplications as possible (in this way determining the maximal filter lengths) within one sample period. Also 9 slower multiplications for modulation and attenuation, each performing just one multiplication during each sample period are required. Finally some additions and control will be needed. If the number of filter coefficients of the several filters and the sample period is known one can compute the maximum time allowed for a multiplication. This knowledge is needed to adjust the hardware to it's function.

The first prototype design developed uses a parallel multiplier. In the given 1.5 μm CMOS technology this multiplier is fast enough to compute the convolution of a filter with 90 coefficients within a single sample period. If one uses 8 stage pipelining it is possible to compute the 14 required convolutions together with some of the necessary modulation and attenuation computations with just two parallel multipliers and a large amount of pipeline registers. Looking at the first design we see that no use is made of the sample rate reduction introduced by the decimation factors in the algorithm. This sample rate reduction allows much more time for the computation of the intermediate filterconvolutions of the algorithm.

This first design prototype results in a two chip solution which requires approximately 64 mm² in the ES2 1.5 μm CMOS Standard Cell technology for each chip. Hereto the algorithm is divided in two parts (an upper part and a lower part) each performed by a single chip. This can be easily done if the intermediate and final results are combined on each single chip. In this way both chips will be exactly equal. The chips will be running at 33 MHz.,

will contain 64 pins of which 52 are used to address and read the two 40 ns 1K RAM banks of coefficient and sample memory [8]. See Table 3.

For the second design we put as aim the to use the sample rate reduction for subsequent filters within the algorithm, meanwhile avoiding the use of pipelining. This at the cost of more but smaller multipliers. Because of the use of several multipliers instead of one or two one can trim the filterlengths and multiplier speed so that large filters are computed at the lower sample rates. This results in a design consisting of 4 serial-parallel multipliers which make optimal use of the possibly lower intermediate sample rate.

The algorithm is divided over four multipliers. see Fig. 3. The first multi-

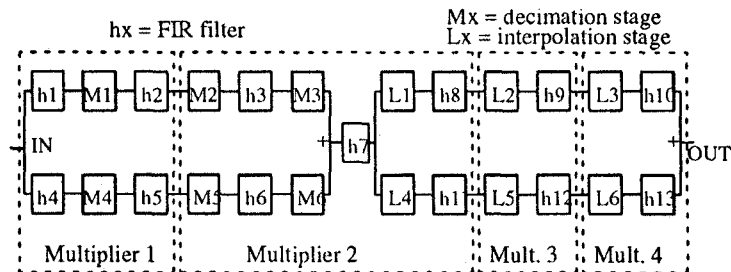


Figure 3: Division of the algorithm over different multipliers

plier performs the first 4 filter convolutions (h1, h2, h4, h5) and decimations (m1 and m4) at the initial (and highest) sample rate. The second multiplier (second stage) handles filters h3, h6, h7, h8 and h11, as well as decimation m2, m3, m5, m6 and interpolation l1 and l4. This is possible because the sample rate is m1 times lower. The third multiplier is used to compute filters h9 and h12 together with interpolation l2 and l5. The last multiplier (again working at the highest sample rate) computes filter h10 and h13. By optimizing the filters for this design, this means making filters h1, h4, h10 and h13 small while allowing the other filters to get relatively large, one makes more efficient use of the available computing power without the need for parallel multipliers or pipelining techniques [9], [10].

Also in this second prototype the characteristic of equal filter coefficients around the center coefficient of FIR filters is turned into a profit. Of a filter of length N coefficients 1 and N, 2 and N-1 etc. are equal so first samples 1 and N, 2 and N-1 etc. can be added before the multiplication with the filtercoefficient is performed, in this way reducing the number of multiplications required and the size of coefficients memory about two times. This makes the computation of more filters with one multiplier or the use of larger filters possible. See Fig. 4.

This second design prototype results in an estimated chipsize of 35 mm² in the ES2 1.5 μm CMOS Standard Cell technology. The chip will be running at 25 MHz and will have the same pincount of 64 pins. The required amount of off-chip memory will be around 2kB RAM. The access time of the used RAM's must be less than 20 ns. See Table 3.

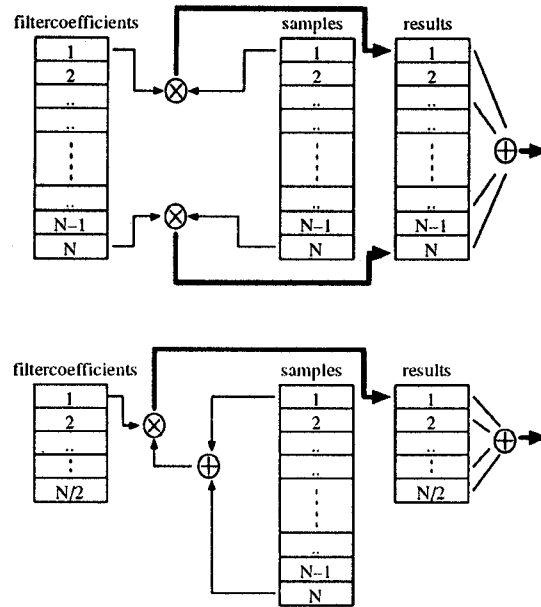


Figure 4: Reduction of multiplications through symmetric filter characteristics

In the current stage of the VLSI implementation, external memory will be used to store the filter coefficients and data samples. Embedded memory may replace this data storage in future. It will be obvious that the number of memory accesses is limited by the read/write access time of the applied memory devices. There are two reasons for minimizing the number of memory accesses of a particular filter(bank) implementation. Reducing the number of accesses will reduce the cost of the memory environment since the prizes of the memories are proportional to the speed of the devices. The size of the memories should also be taken into account for the same reason. Secondly, minimizing the number of memory accesses makes it possible to implement filters with more stringent filter specifications.

ACKNOWLEDGMENTS

We gratefully acknowledge the funding and continuing support by the Dutch PICO and European EUROCHIP design actions. This support enables us to use up-to-date design software and have the designs processed at chip foundries. We also acknowledge the Polytechnic Enschede for allowing some of their students to work on our project. We specially want to thank Hans Snijders for his ever lasting software and system support. Furthermore we want to thank all other members of our group who have been a help in this

| | <i>1st design</i> | <i>2nd design</i> |
|------------|------------------------------|------------------------------|
| no. chips | 2 | 1 |
| area | 2*64 mm ² | 35 mm ² |
| frequency | 33 MHz | 25 MHz |
| pins | 64 | 64 |
| technology | 1.5 μ m | 1.5 μ m |
| memory | 2 k | 2 k |

Table 3: Comparison of two design prototypes

project.

References

- [1] A.V. Oppenheim, R.W. Schaffer, *Discrete - Time Signal Processing*, Prentice-Hall, Englewood Cliffs N.J., 1989.
- [2] Signal Processing Workstation (SPW), users guide, Comdisco, 1991.
- [3] R.E. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing*, Prentice-Hall, Englewood Cliffs N.J., 1983.
- [4] C.G.M. van Asma, Design of a phase linear audio equalizer, internal report EL-BSC-91N127, University of Twente, Enschede, the Netherlands, 1991
- [5] O.E. Herrmann, J. Smit et al., *Modeling & Design Language, Hardware Description Manual*, University of Twente, 1992.
- [6] Mentor Graphics, IDEA Series Schematic Capture and Cell Station User's Manual, 1989/1990.
- [7] H.F.G.M. Huijnen, M.R.C.M. Berkelaar, J.F.M. Theeuwen, PICO logische synthese gebruikershandleiding.
- [8] D. Schouten & W.J.A. Brunink, Design & implementation of a digital linear phase audio equalizer (in Dutch), internal report EL-BSC-91N155, University of Twente, Enschede, the Netherlands, 1991.
- [9] J.V. Pol, VLSI-implementation of a digital linear phase audio equalizer (in Dutch), internal report EL-BSC-92N068, University of Twente, Enschede, the Netherlands, 1992.
- [10] F.B. Drenth, Realization of a digital linear-phase audio equalizer (in Dutch), internal report EL-BSC-92N117, University of Twente, Enschede, the Netherlands, 1992.