MASTER

Compact bitstream conversion

a new algorithm for the digital format conversion between 1-bit signals and PCM signals

Franken, G.

Award date:
1991

Eindhoven University of Technology
Department of Electrical Engineering
Digital Systems Group (EB)

Compact Bitstream Conversion

A new algorithm for the digital format conversion
between 1-bit signals and PCM signals.

Graduation report by Gerwin Franken
EB 288

Supervisors:
Prof. ir. M.P.J. Stevens
Ir. E. Roza

Philips Research Laboratories Eindhoven, the Netherlands, February 1991.

Abstract

For A/D- and D/A-conversion, the combination of a $\Sigma\Delta$-modulator and a digital pre-processor or post-processor is getting more and more popular. Mostly, these digital processors consist of digital filters, designed using frequency considerations. A new algorithm, derived from a time-domain viewpoint has been proposed by E. Roza. The algorithm permits sampling before filtering, which is very interesting for high-frequency applications. Because band-limitation is the only requirement for the internally used filter, some hardware reductions can be made by chosing only single poles at very specific locations.

The algorithm is called COmpact BItstream Conversion (COMBIC) and this report deals with the analysis and evaluation of its performance. For this, a simulation-package has been developed to determine the conversion-quality, varying parameters as the internally used filter and internal calculation-accuracy. During all simulations and analyses, the application as a post-processor for A/D-conversion of video-signals was used as a reference and it showed that COMBIC-converters with quite small dimensions already forfill the required conversion-quality.

I worked on the subject at the Philips Research laboratories from may 1990 until february 1991 as the graduation project in my study in electric engineering at the University of Technology in Eindhoven. I would like to thank ir. E. Roza, leader of the group "digital VLSI" at the Philips Research Laboratories, and prof. ir. M.P.J. Stevens, from the University of Technology Eindhoven, for supervising me during the project.

Philips Research Laboratories, February 1991.

Contents

Chapter 1

Introduction

1.1 Digital Signal Processing

Digital Signal Processing (DSP) is more and more used for several applications like for example the processing of video- and audio-signals. This is caused by the decreasing costs and scale of DSP-circuits, their high reliability, reproducability and capability to perform complex processing. Like audio- and video-signals, most signals vary analog and continue in time. To process these analog and continue signals in a digital way, they have to be converted to and from a digital representation. The conversion from a continue and analog signal to a digital format is called "Analog/Digital-conversion" ("A/D-conversion"). Two different digital formats are presented in section 1.2. The conversion back to an analog signal is called "Digital/Analog-conversion" ("D/A-conversion") and it consists of low-pass filtering these digital signals.

1.2 Two digital signal-formats

Using Pulse Coded Modulation (PCM), a signal is sampled at equidistant moments and the samples are quantized to represent them in a finite number of bits. An example of an analog sine converted in a PCM-signal is given in figure 1.1.

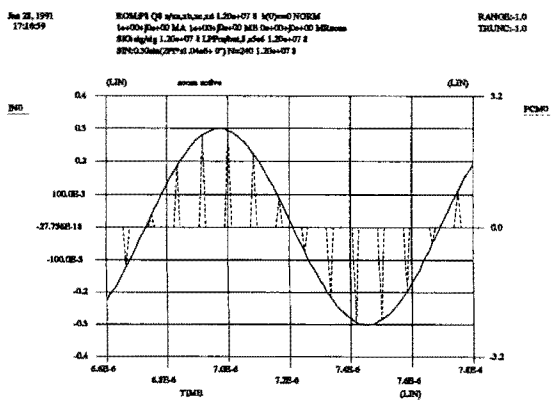

Figure 1.1: A sequence of PCM samples and its original analog and continue sine-wave.

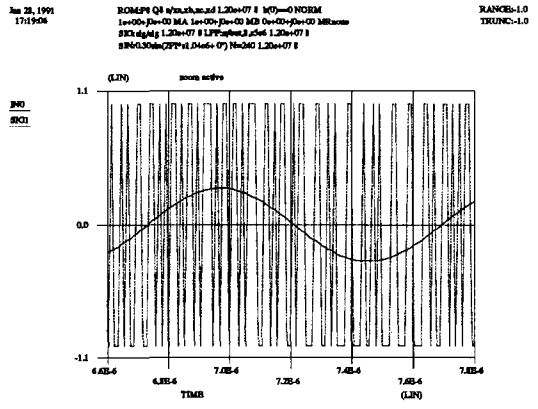

Figure 1.2: A sequence of 1-bit symbols and its original analog and continue sine-wave.

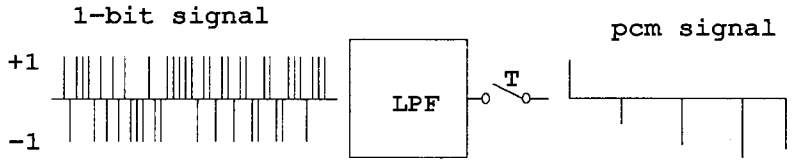

Figure 1.3: Conversion from a 1-bit signal with sampling-period $\frac{T}{q}$ with a "decimating filter" that uses an internal sampling-period $\frac{T}{Q}$ to a PCM signal with sampling-period T .

A 1-bit signal is a sequence of symbols that can be either '1' or '-1', representing the sign of the output of a filter, which is fed with the difference between the orginal signal and the sequence of symbols itself. The converter tracks the input-signal by integrating the difference with its 1-bit output symbols as correcting up/down steps. An example of a sine converted in a 1-bit signal is shown in figure 1.2.

Each format has its advantages and disadvantages. PCM signals are easy to use for signal-processing and 1-bit signals are easier in the conversion to, and from the analog time domain.

For these considerations, the conversion between both of the formats has always been very interesting and implemented in the so-called "decimating filters" [5]. These decimating filters are used for pre- and post-processing, from and to the PCM format, while the 1-bit format is used for the A/D and D/A conversions themselves.

For a comparable quality of the digital signals, the 1-bit symbol-rate (single bits) will be much higher than the PCM symbol-rate (words). Even the total bit-rate is a little higher for the 1-bit signals.

1.3 Compact Bitstream Conversion

The named decimating filters perform calculations at the high sampling-rate of the 1-bit signals. Therefore, an algorithm using only the much lower PCM sampling-rate could lead to an expansion of applications for signals of higher band-widths like video-signals.

With the Compact Bitstream Conversion (COMBIC) algorithm, the 1-bit sampling-rate is only used in parallel to series converters. Because only band-limitation of the internally used filter will be required, some freedom is left for the choice of poles (section 2.3). The use of single poles (section 2.5) gives the opportunity to reduce the fundamental calculation-period from the 1-bit sampling-period to the PCM sampling-period (section 2.6 and section 2.7). By using Lookup-tables (section 2.10) containing all the possible results of multiplications with system-dependent constants, the Compact Bitstream Conversion can be implemented with a much lower number of time and area consuming multipliers. Separation of the format-dependent parts of the filter-sections makes the use of Lookup-tables much easier (section 2.8). The choice of a filter such that the recursive multiplications can be performed with a simple shift operation (section 2.9), should lead to even simpler circuits. The COMBIC-algorithm is symmetrical, and it is possible to implement both the COMBIC-"1-bit to PCM"-converter and the COMBIC-"PCM to 1-bit"-converter with practically the same functional blocks. In this report, the results of simulations will be used to come to the specifications of practical realizations.

1.4 Application

The COMBIC-"1-bit to PCM"-converter can be used as a post-processor of a "1-bit A/D-converter" instead of the so-called decimating filters. This report deals with the application of the COMBIC-algorithm in the video-domain with typical signal-band-widths of 5 MHz. The PCM sampling-rates will therefore be at least 10 MHz. For video-applications, 8 bits PCM samples should be sufficient and with symbol-rates of minimally 120 MHz, the 1-bit signals will be in the same satisfying order of quality.

Chapter 2

The COMBIC-algorithm

2.1 Introduction

The samples of a PCM signal can be derived from a 1-bit signal, considering that the responses of a band-limited filter to the PCM signal and the 1-bit signal should be the same at the PCM sampling moments. Using this statement, the error introduced by the conversion from 1-bit to PCM signals, depends on the band-limitation of the used filter (See section 3.4). A great advantage of the algorithm is that the filter does not have to have an ideal all-pass character in the signal-band, which gives the opportunity to choose a filter that makes the implementation very simple. In the following, some equations, describing the algorithm that is symmetrical for the conversion in both directions, are derived in a global and short way. In appendix A, a more detailed and analytical derivation is given.

2.2 Signals

The PCM signal $x_{pcm}[jT]$, with a symbol-period T, can be represented by

$$x_{pcm}[jT] = \sum_{k=0}^{\infty} b_k \delta[jT - kT] \tag{2.1}$$

where b_k are the PCM samples and k and j are integers.

The 1-bit signal $x_{1bit}[jT]$, with a symbol period $\frac{T}{q}$, can be represented by

$$x_{1bit}[jT] = \sum_{n=0}^{\infty} a_n \delta[jT - n\frac{T}{q}] \tag{2.2}$$

where $a_n = 1$ or -1, n is an integer, and q is an integer larger than 1.

Considering a causal filter with an impulse response $h[t]$, the sampled response of the filter to the signal $x[t]$ equals

$$y[t] = x[t] * h[t] \tag{2.3}$$

where $*$ denotes convolution.

The sampled response to the PCM signal is

$$\begin{aligned} y_{pcm}[jT] &= x_{pcm}[t] * h[t] \\ &= \sum_{k=0}^{j} b_k \delta[kT] \times h[jT - kT] \end{aligned} \tag{2.4}$$

while the sampled response to the 1-bit signal is

$$\begin{aligned} y_{1bit}[jT] &= x_{1bit}[t] * h[t] \\ &= \sum_{n=0}^{jq} a_n \delta[n\frac{T}{q}] \times h[jT - n\frac{T}{q}] \end{aligned} \tag{2.5}$$

2.3 Difference-equation

Considering that, using a band-limited filter, the sampled response of a 1-bit signal must be equal to the sampled response of a PCM signal at the PCM sampling moments jT, a "difference-equation" can be written:

$$y_{pcm}[(j+1)T] - y_{pcm}[jT] = y_{1bit}[(j+1)T] - y_{1bit}[jT] \tag{2.6}$$

or with use of equation (2.3):

$$\begin{aligned} &x_{pcm}[(j+1)T] * h[(j+1)T] - x_{pcm}[jT] * h[jT] = \\ &\quad x_{1bit}[(j+1)T] * h[(j+1)T] - x_{1bit}[jT] * h[jT] \end{aligned} \tag{2.7}$$

2.4 Recursion

The function $h[t]$ is splitted (figure 2.1) into a finite impulse-response $h_1[t]$ and an infinite impulse-response $h_2[t]$:

$$h[t] = h_1[t] + h_2[t] \tag{2.8}$$

with

$$h_1[t] = h[t] \times \{U[t] - U[t-T]\} \tag{2.9}$$

$$h_2[t] = h[t] \times U[t-T] \tag{2.10}$$

where $U[t]$ is the unit step function.

Replacing $h[t]$ by $h_1[t] + h_2[t]$, equation (2.7) can be rewritten as

$$\begin{aligned} &x_{pcm}[(j+1)T] * h_1[(j+1)T] \\ &= x_{1bit}[(j+1)T] * h_1[(j+1)T] \\ &\quad + \{x_{1bit}[(j+1)T] * h_2[(j+1)T] - x_{1bit}[jT] * h[jT]\} \\ &\quad - \{x_{pcm}[(j+1)T] * h_2[(j+1)T] - x_{pcm}[jT] * h[jT]\} \end{aligned} \tag{2.11}$$

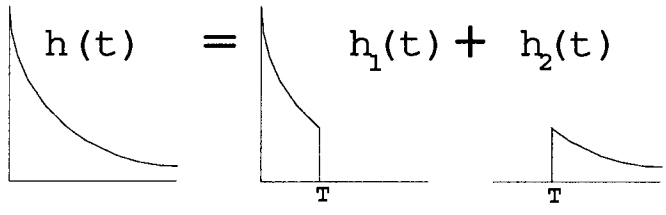

Figure 2.1: Splitting $h[t] = h_1[t] + h_2[t]$.

Because $h_2[t] = 0$ for $t < T$, the last two terms of equation (2.11) can be rewritten using:

$$x[(j+1)T] * h_2[(j+1)T] \;=\; x[jT] * h[(j+1)T] \tag{2.12}$$

and the difference-equation becomes

$$\begin{aligned} & x_{pcm}[(j+1)T] * h_1[(j+1)T] \\ &= x_{1bit}[(j+1)T] * h_1[(j+1)T] \\ &\quad + \{x_{1bit}[jT] * h[(j+1)T] - x_{1bit}[jT] * h[jT]\} \\ &\quad - \{x_{pcm}[jT] * h[(j+1)T] - x_{pcm}[jT] * h[jT]\} \end{aligned} \tag{2.13}$$

By introducing the causal function $g[t]$

$$g[t] \;=\; \{h[t+T] - h[t]\} \times U[t] \tag{2.14}$$

the difference-equation becomes

$$\begin{aligned} & x_{pcm}[(j+1)T] * h_1[(j+1)T] \\ &= x_{1bit}[(j+1)T] * h_1[(j+1)T] \\ &\quad + \{x_{1bit}[jT] * g[jT]\} \\ &\quad - \{x_{pcm}[jT] * g[jT]\} \end{aligned} \tag{2.15}$$

When the inverse function $h_1^{-1}[t]$ is defined such that:

$$h_1[t] * h_1^{-1}[t] \;=\; \delta[t] \tag{2.16}$$

and it is possible to construct a functional block performing $h_1^{-1}[t]$, then equation (2.15) describes a recursive "digital format converter" as displayed in figure 2.2. All blocks can be implemented as transversal or recursive filters except for the block $h_1^{-1}[t]$. This block determines which finite set of signal-samples during a period T should have caused the passed response to $h_1[t]$. Producing PCM signals, only one single sample occurs during such an interval T, and the convolution with $h_1[t]$ is reduced to a single multiplication. The inverse function $h_1^{-1}[t]$ is performed by a division with the same constant.

When generating 1-bit signals, a vector of q 1-bit symbols which response to $h_1[t]$ comes the closest to the given input-sample should be found. The determination of this vector of 1-bit symbols can be performed by a search in a sorted list. This list is generated by calculating the responses of all the 2^q vectors and sorting them according the calculated values. This sorted list will be called "MAP-table" (See figure 2.11).

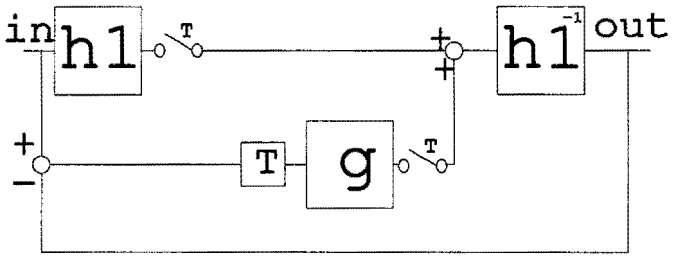

Figure 2.2: Block-diagram of the "digital format converter".

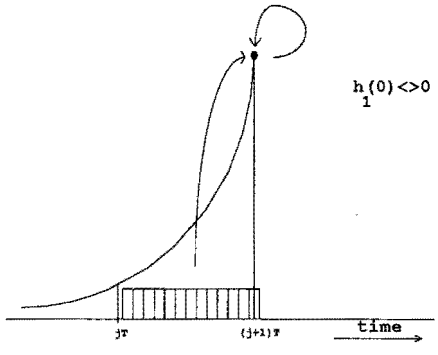

Figure 2.3: Convolution of the vector of q signal-samples between sampling-moments jT and $(j+1)T$ with $h_1[t]$ to obtain their response at the PCM sampling-moment $(j+1)T$. The samples in the vector are weighted according the curve $h_1[-t]$ above them.

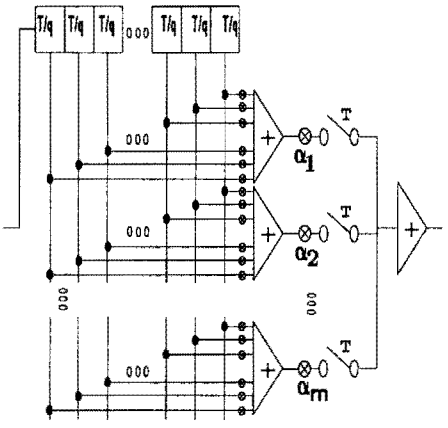

Figure 2.4: Block-diagram of $h_1[jT]$ implemented as an array of weighting-factors. The separate samples in the shift-register are weighted with the factors $e^{s_m i\frac{T}{q}}$ and the contributions of the sections are weighted with the factors α_m.

2.5 Simplification of the internal filter

In order to simplify the converter, it is assumed that $h(t)$ is the impulse response of a causal filter with M single poles. Then $h[t]$ can be expressed as a sequence of exponential functions:

$$h[t] = U[t] \sum_{m=0}^{M-1} \alpha_m e^{s_m t} \tag{2.17}$$

where $U(t)$ is the unit step function and $\mathrm{Re}[s_m] < 0$ to assure stability.

The earlier introduced function $g[t]$ (equation (2.14)) becomes by this:

$$g[t] = U[t] \sum_{m=0}^{M-1} \alpha_m \{e^{s_m T} - 1\} \times e^{s_m t} \tag{2.18}$$

Now, $h_1[jT]$ can be implemented as in figure 2.4. With a sequence of q 1-bit symbols (or a single PCM sample with $(q-1)$ following zeros) as the input of the shift-register, the output equals the output of $h_1[t]$ at the PCM sampling-moments jT.

2.6 "Equi-Shifting" samples

A filter-section performing $e^{s_m t}$ has the property that the time-shifted response equals the non-shifted response multiplied with a constant:

$$\begin{aligned} e^{s_m[t+\tau]} &= e^{s_m \tau} \times e^{s_m t} \\ &= C \times e^{s_m t} \end{aligned} \tag{2.19}$$

When a 1-bit symbol is shifted backward or forward in time over a period τ, and it is multiplied with the proper constant $e^{s_m \tau}$, it still causes the same response to the section

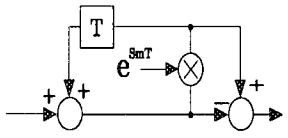

Figure 2.5: Recursive configuration of a section with impulse-response $g_m[jT] = \{exp^{s_m T} - 1\} \times e^{s_m jT}$.

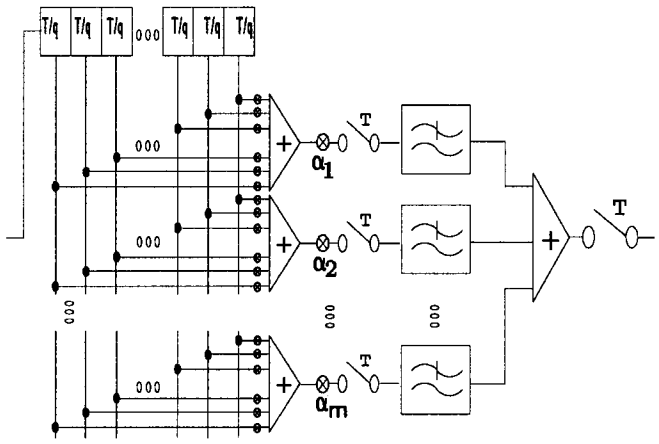

Figure 2.6: Block-diagram of $g[jT]$.

$e^{s_m t}$ after both the sampling-moments T and $T + \tau$. This operation will be called "equi-shifting".

Therefore, a set of q 1-bit symbols, each equi-shifted to the next or previous PCM sampling-moment jT, can be added to one single sample while their merged response remains the same at the following PCM sampling-moments. The filter-sections of $h_1[t]$ as implemented in figure 2.4 perform such an operation.

2.7 Recursive calculations at the PCM sampling-rate

Because the response to $g[t]$ in figure 2.2 is only of interest at the PCM sampling moments jT, it can be replaced by a block similar to figure 2.4 where each equi-shifting section is followed by a recursive block performing $\{exp^{s_m T} - 1\} \times e^{s_m jT}$ at the PCM sampling-rate (See figure 2.5). This results in a configuration as shown in figure 2.6. The arrays of weighting-factors 'equi-shift' the input-symbols to a PCM sampling-moment jT while the recursive sections generate the "tail" of the impulse-response at the following PCM sampling-moments. Using the implementation of $g[jT]$ as given by figure 2.6 the converter becomes as shown in figure 2.7.

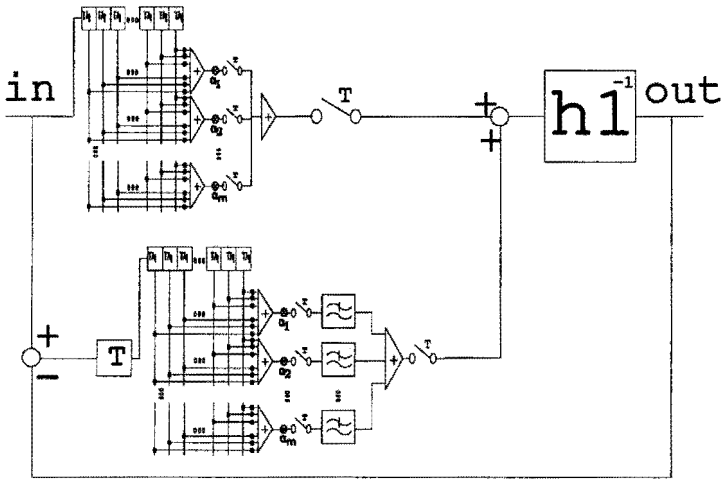

Figure 2.7: Block-diagram of the "digital format converter" using the implementations of $h_1[jT]$ and $g[jT]$.

2.8 Separate arrays for both the signal-formats

Looking at the structure of the converter in figure 2.7, it is obvious that the arrays of weighting-factors can be splitted into two arrays such that both are excitated by only PCM signals or only 1-bit signals. This finally results in a configuration as shown in figure 2.8 in which both the arrays of weighting-factors can be implemented as Lookup-tables and $h_1^{-1}[t]$ can be implemented with a multiplier for "1-bit to PCM conversion" and with a sorted list for the "PCM to 1-bit conversion".

2.9 Placement of the poles

The only restriction for the internal filter is its band-limitation, while there is no ideal all-pass character required in the PCM-band. So there are still quite a lot of possibilities left for the placement of poles and optional zeros. A last reduction of complexity is obtained by using negative powers of two for the constants $e^{s_m T}$, which are used in the recursive parts of $g[jT]$ to determine the locations of the poles s_m. Multiplications with the constant 2^{-n} can be performed by a bit-wise shift-operation of n positions.

2.10 Lookup-tables

Filling in equation (2.15) with equation (2.4) and equation (2.5), while assuming $h[0] \neq 0$, the COMBIC-algorithm can be described by:

$$\sum_{m=0}^{M-1} B_{m(j+1)} = \sum_{m=0}^{M-1} A_{m(j)} + \sum_{m=0}^{M-1} \Gamma_{m(j)}[1 - e^{s_m T}] \tag{2.20}$$

where

$$B_{m(j+1)} = \alpha_m b_{j+1}$$

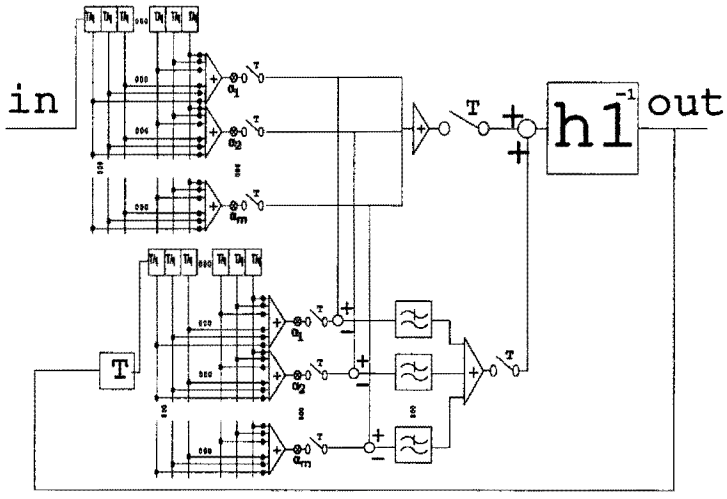

Figure 2.8: Block-diagram of the COMBIC-converter, with separate arrays of weighting-factors for both the PCM and 1-bit signals.

$$A_{m(j)} = \alpha_m \times \sum_{n=1}^{q} a_{qj+n} e^{s_m(T-n\frac{T}{q})}$$
$$\Gamma_{m(j)} = \Gamma_{m(j-1)} \times e^{s_m T} - A_{m(j-1)} + B_{m(j)}$$

In appendix A the equations for $h[0] = 0$ and for the COMBIC-"PCM to 1-bit"-converter are given too.

In an implementation for video-applications Lookup-tables can be used to replace the arrays of weighting-factors and avoid the time-consuming multiplications with implementation-dependent constants. The so-called "$B_{m(j)}$-tables " contain the responses to the COMBIC-filter of all possible PCM-inputs and the so-called "$A_{m(j)}$-tables" contain the filter-responses of all the possible 1-bit vectors of length q. In figure 2.9, figure 2.11 and figure 2.10, the use of the Lookup-tables is illustrated.

The $A_{m(j)}$-tables will typically be the largest. Dividing these $A_{m(j)}$-tables, while the outputs of the separate parts will be added as the filter-responses of the two halves of the 1-bit vector, will reduce their size from $2^q \times M$ entries to $2 \times 2^{\frac{q}{2}} \times M$ entries. In appendix D some considerations are made about a possible reduction of the so-called "MAP-table" performing $h_1^{-1}[t]$ in the COMBIC-"PCM to 1-bit"-converter.

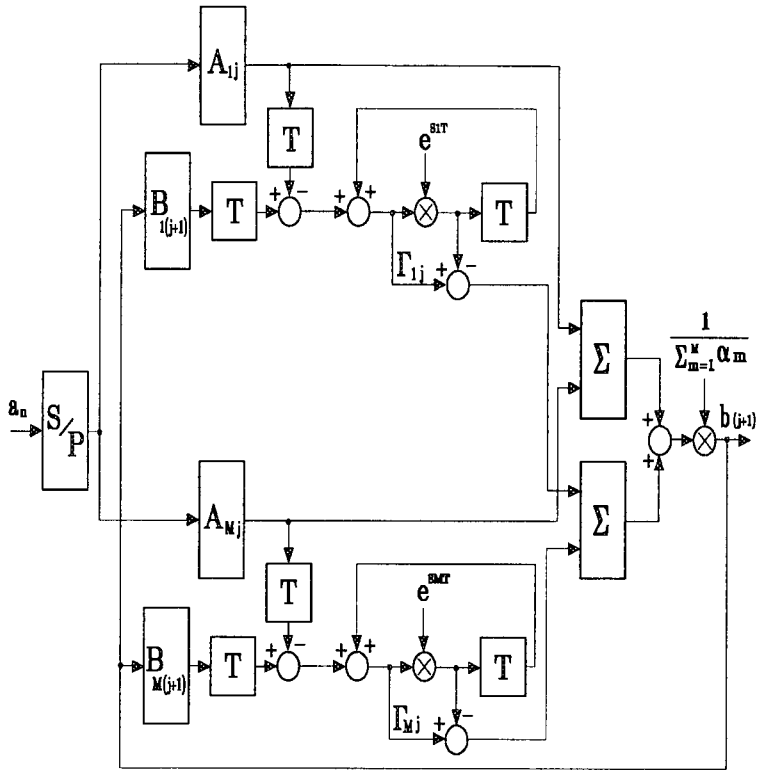

Figure 2.9: Block-diagram of the COMBIC-"1-bit to PCM"-converter for $h[0] \neq 0$ using Lookup-tables.

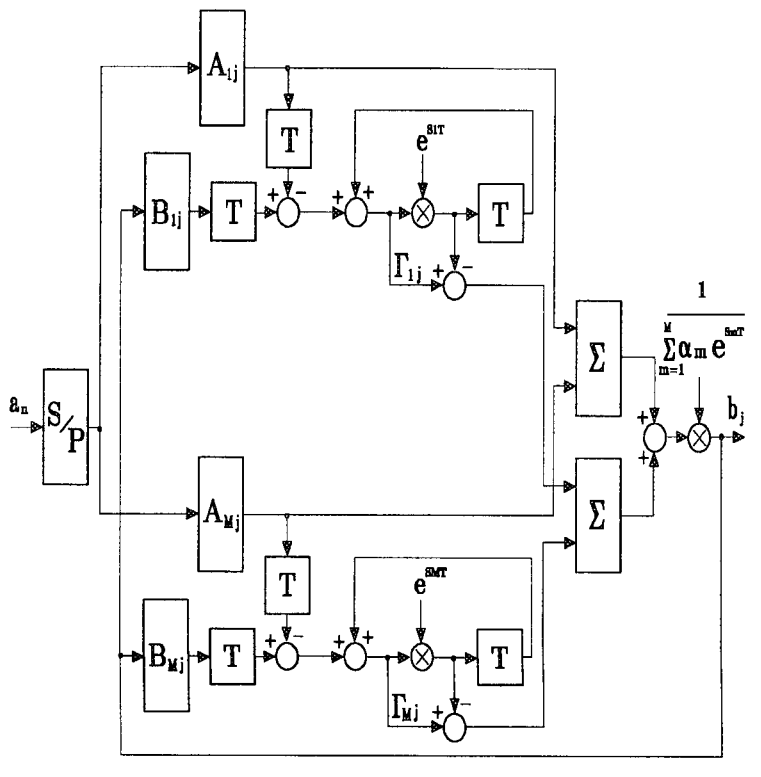

Figure 2.10: Block-diagram of the COMBIC-"1-bit to PCM"-converter for $h[0] = 0$ using Lookup-tables.

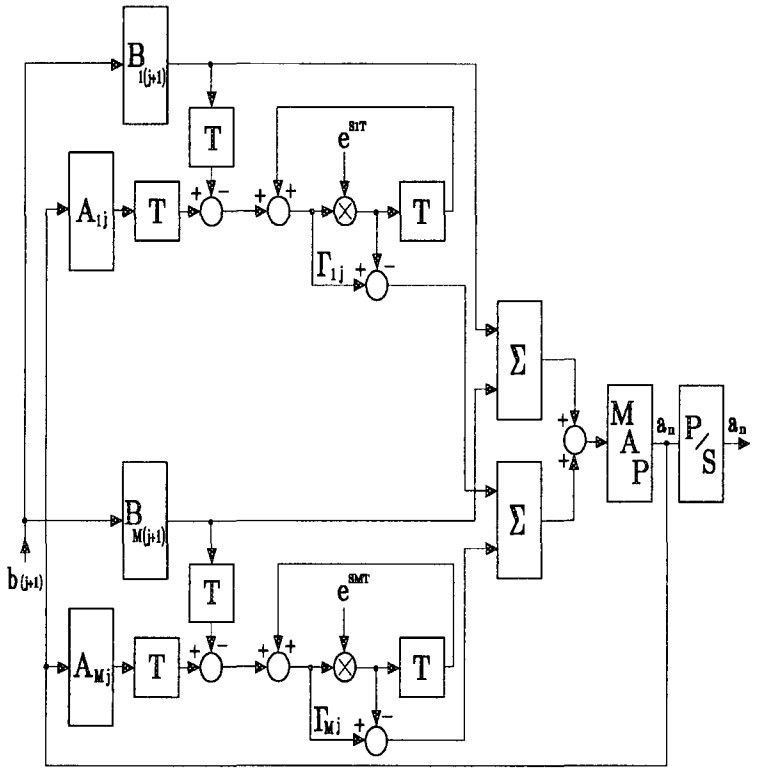

Figure 2.11: Block-diagram of the COMBIC-"PCM to 1-bit"-converter using Lookup-tables.

Chapter 3

Noise and conversion-errors

3.1 Introduction

When considering the quality of COMBIC-converted signals, a differentation has to be made between the sources of errors. When converting from 1-bit signals to PCM signals, the quality of these signals themselves should be kept in mind while formulating specifications for the conversion-precision. Varying the order of the COMBIC-filter and the placement of poles and zeros will influence the error introduced by the COMBIC-algorithm as described in section 3.4. Besides the errors caused by the COMBIC-algorithm itself, the internal use of a finite number of bits introduces errors too, which can be divided into "quantization-noise" and so-called "limit-cycles". There are two options for the type of rounding as it is used for quantization. The first one is called "rounding-off" (or just "rounding") which uses the closest possible value as the approximation. But the so-called "magnitude truncation" (or just "truncation") simply truncates values and ignores the least significant bits or decimals. In the developed COMBIC-simulation package, the type of rounding can be chosen for PCM-signals and internal calculations.

3.2 Quantization-noise

Due to the finite calculation precision for internal calculations in the COMBIC-converters, quantization-noise is introduced. If the input-signal varies sufficiently, the quantization of intermediate results has about the same effect as adding noise to the useful signal. This gives an undesirable degradation of the Signal/Noise-ratio. To analyse these effects, it is assumed that each quantizer can be considered as an additional noise-source with a flat spectrum.

Quantization by rounding-off with a quantization-step q is accompagnied by a rounding-off error between $-\frac{q}{2}$ and $\frac{q}{2}$; the corresponding noise-source has a total nois-power of $\frac{q^2}{12}$. Magnitude-truncation with the same quantization-step gives a quantization error between $-q$ and 0 for positive signals and between 0 and q for negative signal values. The power of the corresponding noise-source can be taken to be $\frac{q^2}{3}$. As quantizers are located inside the filter-sections, the noise no longer has a flat spectrum at the output of the section in which it arises. When calculating with such quantization-errors, it should be avoided to subtract large values which are close to eachother. The resulting difference will be close

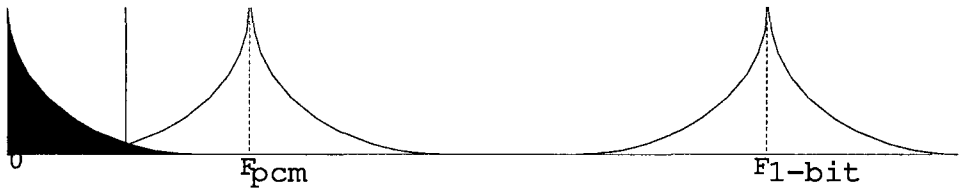

Figure 3.1: Spectral folding around $\frac{1}{2}f_{pcm}$ of error-components which lay outside the PCM-band when the filtered 1-bit signal is sub-sampled to the PCM sampling-rate.

to the quantization-steps, which leads to large relative errors. When using a filter which is a sequence of first-order sections:

$$h(t) = U(t) \sum_{m=0}^{M-1} \alpha_m e^{s_m t}$$

the residues α_m will reflect the possible occurance of such errors. When their values are large and both positive and negative the sum of them could become relatively very small. This showed to be of increasing importance when using higher order filters.

3.3 Limit-cycles

The so-called "limit-cycles" can be seen as a constant or periodic error in the output-signal which is introduced by rounding-off intermediate results in recursive filters. It occurs when rounding-off quantizers cause an up and down rounding-effect in the recursive sections of a filter (See [5]). When simulating with sine-waves as input-signals, there is a good chance for "limit-cycles" to appear. This can be avoided by using magnitude-truncation but unfortunately, this type of rounding causes more quantization-noise (See section 3.2).

3.4 COMBIC-error

The derivation of the COMBIC-algorithm starts with the assumption that the response of both of the digital signals to the same filter is band-limited. Usually, the filtered 1-bit signal is not completely band-limited, which introduces an error when converting to the PCM format. The conversion from PCM to 1-bit signals, is a conversion to a lower sampling-rate which is called "sub-sampling". This sub-sampling can be seen as the folding of the 1-bit signal-spectrum around the lower PCM sampling-frequency. In figure 3.1 it can be seen that the part of the signal-spectrum that lays outside the PCM-band ($\frac{1}{2}f_{pcm}$) will introduce an error, because it is folded back into the PCM-band. The average noise-power N can be calculated by summing the spectral components of the filtered 1-bit signal outside the PCM-band:

$$N = \frac{1}{\pi T} \int_{w_{pcm}}^{\infty} S_{1bit}(w)|H(w)|^2 dw$$

where $w_{pcm} = \frac{\pi}{T}$ and $S_{1bit}(w)$ is the power spectral density of the 1-bit signal before filtering.

Table 3.1: Theoretic Signal/Noise-ratios for the COMBIC-algorithm using ideal integrators considering the folding-error.

Order of integrator	SN(ideal low-pass)	SN(20% oversampling)
1	10 dB	11.5 dB
2	25 dB	28.0 dB
3	30 dB	34.5 dB
4	33 dB	39.0 dB

For a flat PCM spectrum we have

$$S_{pcm}(w) = R(0)$$

where $R(0)$ is the autocorrelation funtion of the samples b. After filtering the signal value at sampling instants is equal to $b_j \times h[T]$. Hence the RMS Signal/Noise-ratio SN of the conversion with a flat spectrum is therefore equal to:

$$SN = \pi h(T)^2 T / \int_{w_{pcm}}^{\infty} |H(w)|^2 dw$$

Usually, Signal/Noise-ratios in A/D and D/A converters are not based on the RMS signal power but on peak power. For a sine, the difference between the peak-to-peak signal-power and the RMS signal-power is 9 dB.

Using ideal integrators of several orders for the internal COMBIC-filter table 3.1 shows the Signal/Noise-ratio based on RMS signal-power. It can be seen that over-sampling can be used to increase the performance, also because the addition an extra section is more effective when over-sampling.

Chapter 4

Signal and parameter formats

4.1 Introduction

For simulations with the COMBIC-algorithm, several parameter and signal representations in different formats are required. The formats to describe parameters and signals are introduced in section 4.2.

Principally, continue signals should be simulated with an infinite sampling-rate and analog signals should be represented with an infinite calculation precision. To keep the consumed computer-time as short as possible and due to the finite machine-accuracy, both these infinite parameters will be approximated with finite values. From now on, quoting the terms 'infinite', 'continue' and 'analog' will remind about the approximation.

Starting with 'continue' and 'analog' signals, PCM and 1-bit signals can be generated. Because low-pass filtering of the two digital signals should deliver nearly the original 'continue' and 'analog' signal, it would be comfortable to use an object in which all the formats can be put but which gives basicly the same output when these objects are used as the input of a filter. These objects will be defined in section 4.3 but first some types of values and parameters are introduced.

4.2 Formats of parameters and values

The following types are used as arguments for the programs while the type *complex* will be used for signals too:

- *integer* values are numbers without decimals and can be positive and negative.
- *double* values, which can be positive and negative, can be passed in the floating-point notation or the scientific notation where the character 'e' is used as a separator of the mantissa and the exponential part. For example 1000 equals 1e3 and 0.0025 equals $2.5e-3$.
- *complex* numbers are given in the form $RE+jIM$ or $RE-jIM$ where RE can be a positive or negative *double*, but IM can only be a positive *double*. The *characters* 'a', 'b', 'c', 'd' and 'e' can be used to replace a *double* value, representing the values $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ and $\frac{1}{32}$.

When the parameters *low_frequency* and Q are passed arguments too, prefixes can be used to define the *complex* numbers in several ways:

Prefix 'l' causes the value to be taken as relative to the sampling-rate *low_frequency*.

Prefix 'h' is used when the value is given relative to $Q \times low\_frequency$.

Prefix 'x' makes the value to be taken as the term $E = e^{ST}$ with $T = \frac{1}{low\_frequency}$. So xE will be replaced by $S = \frac{1}{T} \times ln[E]$.

Without prefix the value is taken as an absolute value.

- *character* is a single character which is taken case-insensitive by the programs.
- *name* is string of characters, taken case-sensitive, mostly used for file-names.

4.3 General signal vectors

The mentioned signal-object has been implemented as a vector of a specific number of samples following eachother in time , within a specific *range* and possibly rounded to a specific number of *bits*. Here, the *range* is defined as the maximal positive value which is binary represented by a sequence of only 1's. Because at least one PCM sample has to be in such a signal-vector and the COMBIC-algorithm converts one PCM sample into q 1-bit symbols, each vector will represent exactly one PCM sampling-period and therefore contain exactly one PCM sample or q 1-bit symbols. To save processing-time the 'analog' and 'continue' signals are approximated by q samples per vector in 'infinite' precision. This sampling-rate for 'continue' signals will be exactly high enough, when filtering the 1-bit signals. For generalisation, signal-samples are of type *complex* and in infinite precision, while they can be set to specific values (1-bit signals) or rounded according a given number of *bits* within a certain *range* (PCM signals). The PCM sampling-rate will be called *low_frequency* and the 1-bit sampling-rate *high_frequency*. The *integer* Q is defined by their relation:

$$high\_frequency = low\_frequency \times Q \tag{4.1}$$

4.4 Conversion between representations

Since there are building blocks with different input and output formats and these blocks will be chained later, some conversions (See figure 4.1.) have to be defined. When defining a simulation configuration, care should be taken about which signals are the input and output of the building blocks. The converters expect their input to be of a specific format and output-signals will be invalid when this is ignored.

The conversion from a 'continue' and 'analog' signal to the PCM format, consists of sampling and quantizing. The sampling can be done by taking the first element of a vector. The quantizing will be performed by a rounding function which uses the following parameters:

- The *integer* number of *bits*.

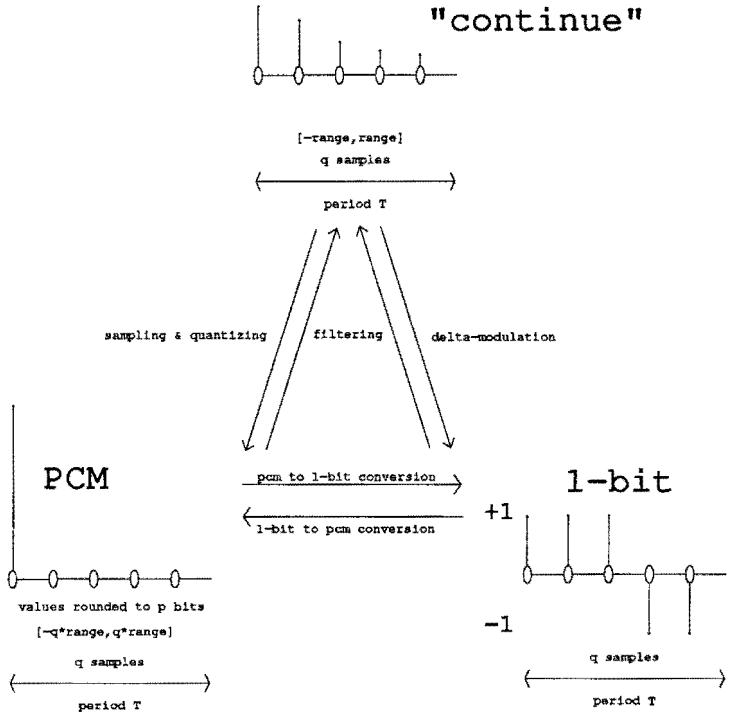

Figure 4.1: Conversions between the 'continue' and 'analog' signals, PCM signals and 1-bit signals.

- A *double* value determining the maximum amplitude of values, called *range*.
- The *type* of rounding which can be "rounding" or "truncating" denoted by the *characters* 'r' or 't' (See also section 3.2).

The conversion from a 'continue' and 'analog' signal to the 1-bit format, can be performed with a $\Sigma\Delta$-modulator which is described in section 5.3.

With the COMBIC-converters, it will be possible to transform signals from the PCM format to the 1-bit format and vice versa.

Because all the signals are put in signal-vectors of the same format and they're scaled for constant signal-energy (section 4.5 and figure 4.1), they can be converted to 'continue' and 'analog' signals, by filltering them with the same low-pass filter.

4.5 Scaling

When considering the continue signal $x_c[t]$, the sampled signal $x_s[jT_s]$ can be written as:

$$x_s[jT_s] = \sum_{n=0}^{j} x_c[nT_s] \times \delta[T_s(j-n)]$$

and will have the same energy as the original signal $x_c[t]$ when $\delta(t)$ is scaled according to T_s.

This means that, with scaling depending on T_s, a signal always gives a filter-response with the same energy, independent of which format is used. Through scaling of a PCM-signal, by multiplying it with q, the response of filters to PCM, 1-bit and 'continue' signals are in the same range.

4.6 Signal-files

All the used signal-files are in the so-called "agap-format". Signals are therefore always graphicly viewable with "agap" and power-spectrums can be calculated with "ugap". To generate sine-wave signal-input-files to be used for simulations a program called "sin" is available and should be used as follows:

```
sin duration amplitude frequency phase low_frequency Q output
```

where:

- *duration* is a positive *double* defining the duration of the sine-wave in seconds.
- *amplitude* is a *double* value defining the maximum value of the sine.
- *frequency* is a real and positive number of type *complex*, defining the frequency of the sine-wave. Prefixes can be used as described in section 4.2.
- *phase* is a *double* value defining the phase-shift of the sine-wave in degrees (360° = 2π).
- *low_frequency* is a *double*, defining the PCM sampling-rate in Hz.
- Q is a positive *integer*, defining the relative rate of the 1-bit symbols such that $high\_frequency = low\_frequency \times Q$.
- *output* is the *name* of the output-file (See figure 4.2) in agap-format and contains the values for a x-variable $TIME$ and a complex y-variable $IN0$ in the RIM notation which means that samples are given by their real and imaginary part (See subsection 5.2.1).

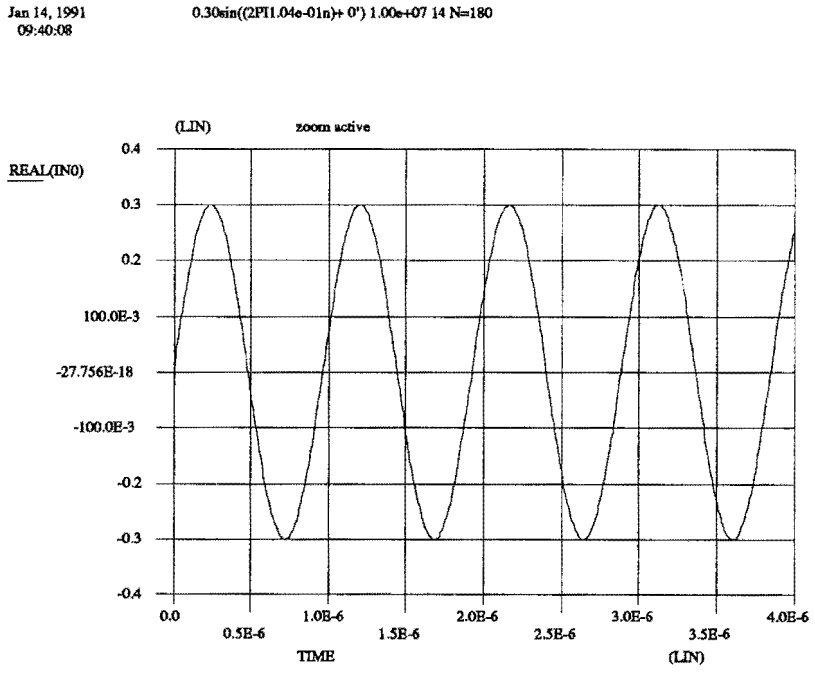

Figure 4.2: An input-sine, generated with "sin", displayed with "agap".

Chapter 5

Building blocks for simulation purposes

5.1 Introduction

The definition of building blocks with compatible standard input and output signal-formats makes the simulation flexible and convenient. In the following several building blocks are described. Building blocks need special files describing their parameters. The program "mlpf" generates filter-description-files and the program "table" generates COMBIC-description-files. Together with an agap-file containing an input-sine these description-files can be use to perform simulations (See figure 5.1). All arguments of the used programs are given in the formats described in section 4.2.

5.2 Filters

5.2.1 Generation of filters

With the program "mlpf" several types of filters can be generated. The passed arguments are as follows:

```
mlpf N type pole_list zero_list low_frequency Q output
```

Where:

- N is an positive *integer*, defining the number of output samples used for an impulse-response or a frequency-response. When the *type* argument is set to R, N is not used but has to be given anyway.
- *type* determines the output-type. The generated output can be one of the following:

  A filter-description-file, containing the parametric description of the filter, with poles, zeros and sampling frequencies, used by the recursive filter-blocks. The chosen *type* is R.

  An impulse-response-file in agap-format, which can be in the RIM-format (real and imaginary part), the POL-format (modulus and argument) and the DB format

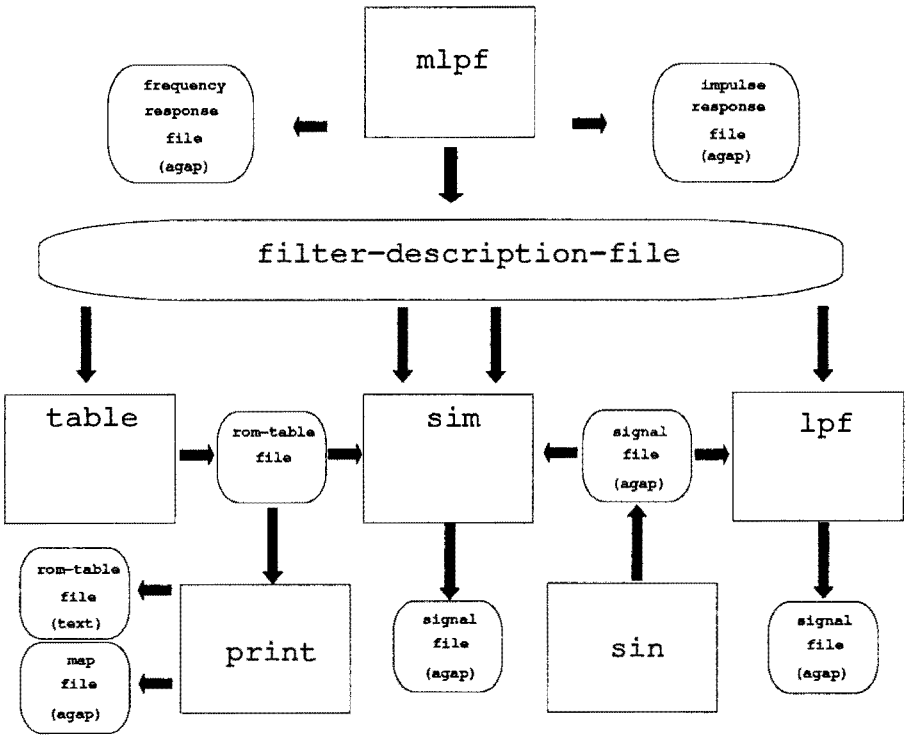

Figure 5.1: Configuration of the COMBIC-simulation package with programs to create and use description-files as they are needed by the building blocks.

(logarithm of the modulus and the argument). These are selected by $TRIM$, $TPOL$ and TDB.

A frequency-response-file in agap-format, which can be in the RIM-format, the POL-format and the DBA-format. Here $FRIM$, $FPOL$, FDB, $WRIM$, $WPOL$ and WDB are the options for *type*. The F and W are used to choose for a regular or a radial frequency on the x-axis.

- *pole_list* is a list of *complex* numbers separated by commas or can be a code-*string* to define Butterworth-filters or $\Sigma\Delta$-modulator-filters (See subsection 5.2.3 and subsection 5.2.4). Prefixes can be used as described for *complex* numbers in section 4.2.

- *zero_list* is a list of *complex* numbers separated by commas in the same format as the poles are given.

- *low_frequency* is a *double*, defining the PCM sampling-rate in Hz.

- Q is a positive *integer*, defining the relative rate of the 1-bit symbols such that $high\_frequency = low\_frequency * Q$.

- *output* is the *name* of the file containing the output which can be a description-file or an agap-file containing a frequency- or an impulse-response.

Mostly three different sorts of filters will be used. These are: internal filters for the COMBIC-converters, Butterworth-filters, and $\Sigma\Delta$-modulator-filters. The generation of these filters is described in the following subsections.

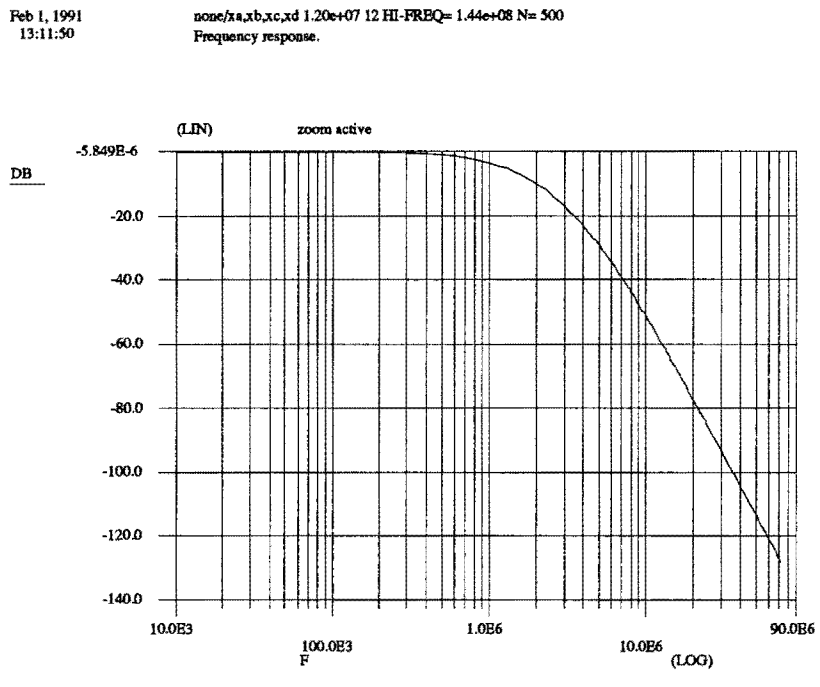

Figure 5.2: Frequency response of a 4^{th} order COMBIC-filter for a PCM sampling-frequency of 12 MHz without zeros and the poles determined by the terms $e^{s_m T}$ equaling $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ and $\frac{1}{16}$.

5.2.2 COMBIC-filters

This type consists of a filter which has to be defined to create a ROM-table-file (COMBIC-description-file) with the program "table" (See section 5.5). The impulse-response of the filter can be written as:

$$h[jT] = \sum_{j=0}^{\infty} \sum_{m=0}^{M} \alpha_m \times e^{j s_m T}$$

where the terms $e^{s_m T}$ should be negative powers of two to be able to replace the multiplications with these terms, in the recursive part of the COMBIC-converters, by a simple bit-wise shift-operation. A COMBIC-description-file of a typical COMBIC-filter is generated with

```
mlpf 1 R xa,xb,xc,xd 1j.5,1-j.5 10e6 14 sys
```

which has it poles determined by the terms $e^{s_m T}$ equal to respectively $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ and $\frac{1}{16}$, a pair of imaginary zeros at half the 10 MHz PCM sampling-rate and a 1-bit symbol-rate of 140 MHz. The frequency-response of an example without zeros is shown in figure 5.2.

5.2.3 Butterworth-filters

A Butterworth-filter has to be defined by replacing the *pole_list* by the *string* "but", followed the order and the normalized frequency, separated by commas. Poles are placed

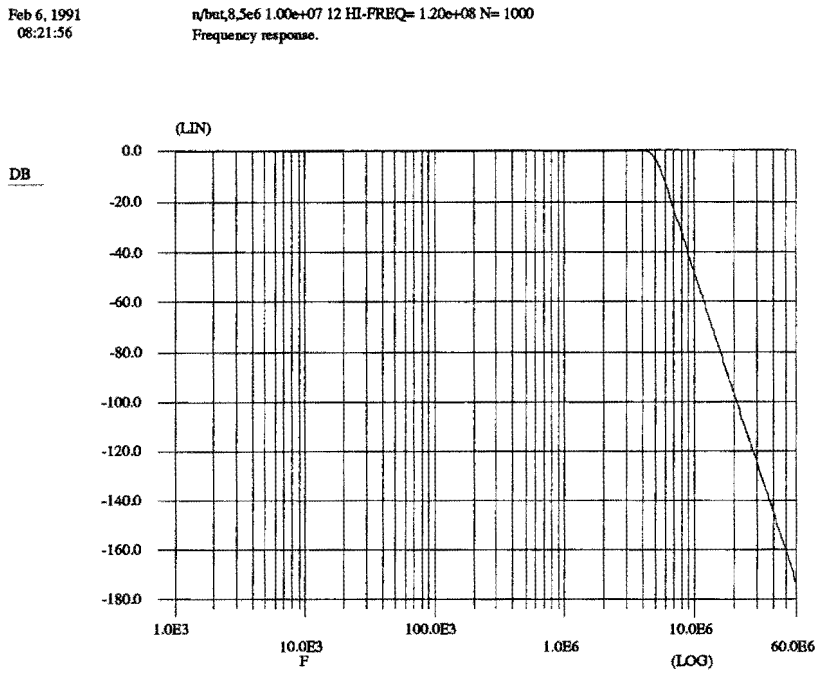

Figure 5.3: Frequency response of a 8^{th} order Butterworth-filter with the normalized frequency at 5 MHz.

on a circle according the Butterworth-definition and normally there are no zeros added, by using "none" for the *zero_list*. The frequency response of a Butterworth-filter is given in figure 5.3. The filter-description-file of a Butterworth-filter is created by:

```
mlpf N type but,order,shift none low_frequency Q output
```

where all arguments are given in the regular way (subsection 5.2.1) except:

- *but* denotes that the *pole_list* is of the Butterworth-type.
- *order* is a positive *integer* defining the number of poles.
- *shift* is a positive real *complex* number, determining the position of the normal frequency of the filter.

5.2.4 ΣΔ-modulator-filters

The low-pass filter in the ΣΔ-modulator (See section 5.3.) has to have predescribed positions of zeros and poles. The used placement of poles and zeros has been taken from a practical realization of a ΣΔ-modulator at a 1-bit sampling frequency of 140 MHz. The poles are located at -2.32 MHz, -2.80 MHz and -4.63 MHz and the zeros at -13 MHz, -14.6 MHz and -40 MHz. The frequency-response of such a filter is shown in figure 5.4. The filter-description-file can be produced with the command:

```
mlpf 1 R h-.01657,h-.02,h-.03371 h-.09285,h-.10428,h-.28571 10e6 14 sigma
```

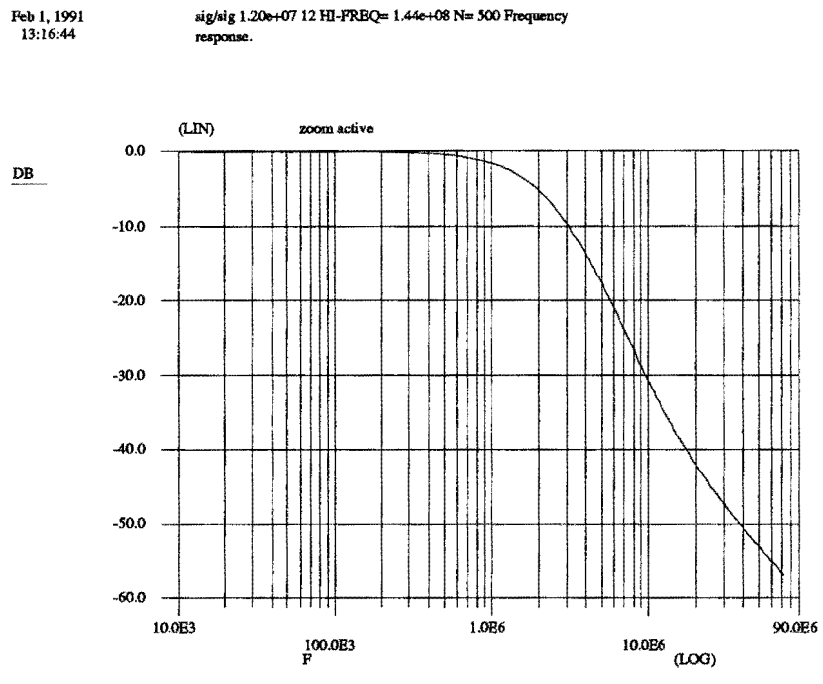

Figure 5.4: Frequency response of the filter for the $\Sigma\Delta$-modulator at a sampling frequency of 144 MHz.

or shorter:

```
mlpf 1 R sig sig 10e6 14 sigma
```

When using the code-*string* "*sig*", the poles and zeros will be automaticly scaled for 1-bit sampling-rates which are not equal to 140 MHz.

5.3 The $\Sigma\Delta$-modulator

With this block 1-bit signal vectors can be generated according to an 'analog' and 'continue' input signal which is passed in vectors. A special low-pass filter will be used. The location of the poles and zeros of this filter, will be taken as they are described in subsection 5.2.4 which results in a frequency-response as showed in figure 5.4.

5.4 The COMBIC-"1-bit to PCM"-converter

The input vectors are assumed to be in the 1-bit format. One PCM sample per 1-bit vector is returned corresponding to the Lookup-tables, the input vector and the current state of the internal filter. A filter-description-file and some other parameters have to be passed to a program called "table" to create a file containing the Lookup-tables needed by this block.
A COMBIC-description-file can be created by:

```
table P,range,type Q filter norm bmj_fbk map_fbk reduction output
```

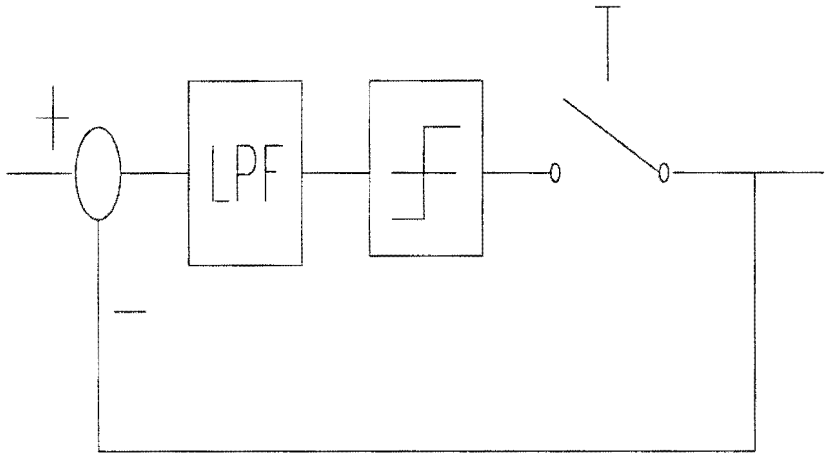

Figure 5.5: Configuration of a $\Sigma\Delta$-modulator.

Where:

- P is a positive *integer* defining the number of PCM *bits* per sample.
- *range* is a *double* defining the maximal amplitude of a PCM-value.
- *type* is a *character* 'r' for rounding and 't' for truncating, while using PCM values within the interval $[-range, range]$ to calculate the $B_{m(j)}$-tables.
- Q is a positive *integer* defining the number of 1-bit symbols per PCM sampling period.
- *filter* is the *name* of the used filter-description-file generated with "mlpf".
- *norm* is one of the *characters* 'y' or 'n' to normalize the Lookup-table or not. When the Lookup-tables are normalized, the last multiplication-factor ($\sum_{m=0}^{M-1} \alpha_m$ when $h[0] \neq 0$ or $\sum_{m=0}^{M-1} \alpha_m \times e^{s_m T}$ when $h[0] = 0$) in the COMBIC-"1-bit to PCM"-converter will be equal to unity, which saves a multiplication (See also figure 2.9 and figure 2.10.).
- *bmj_fbk* is one of the *characters* 'y' or 'n' to use the $B_{m(j)}$-tables for feedback in the COMBIC-"1-bit to PCM"-converter or not. When not, the feed-back will be performed in the internal calculation-accuracy.
- *map_fbk* is one of the *characters* 'y' or 'n' to use the MAP-table for feedback in the COMBIC-"1-bit to PCM"-converter or not. When not, the feed-back will be performed in the internal calculation-accuracy.
- *reduction* is one of the *characters* 'n', 'd', 'g' or 'z' which stand for respectively "no reduction of the MAP-table", "reduction with the distance-algorithm" (section D.2) and "reduction with the grid-algorithm" (section D.3).
- *output* denotes the *name* of the output-file. When no name is given, standard output will be used.

A typical ROM-table-file could be produced with:

```
table 8,.5,r 12 sys y y y n romtable
```

which results in $B_{m(j)}$-tables with 256 addresses, $A_{m(j)}$-tables of 4096 addresses and a non-reduced MAP-table, while all tables are normalized and feed-back is performed using the corresponding tables.

5.5 The COMBIC-"PCM to 1-bit"-converter

A ROM-table-file, created by "table", as used by the COMBIC-"1-bit to PCM"-converter (See section 5.4) is needed to define the behaviour of this block. The block takes an input sample which is rounded according to the chosen PCM format. A 1-bit vector is returned according to the Lookup-tables, the input sample and the current state of the leaky integrators in the internal filter. The use and possible reduction of the MAP-table gets some extra attention in appendix D.

Chapter 6

Simulations

6.1 Introduction

With the program "mlpf" the necessary filters can be generated and they can be examined with "agap" for their impulse- and frequency-responses. Using "table" a COMBIC-description-file can be generated to be used by the COMBIC-blocks. Now, an input sine wave produced by "sin" can be used to perform some simulations with "sim". At this point, it is time to think about a useful simulation configuration.

6.2 Simulation configuration

Because, it is preferred to measure the quality of signals in the 'continue' and 'analog' domain, simulations start and end with signals in this domain. The 1-bit signals can be generated with the COMBIC-"PCM to 1-bit"-converter and with the $\Sigma\Delta$-modulator. Obviously, the conversion of these 1-bit signals to PCM format has to be tested too. To check and compare the quality of the digital format signals in the continue and analog domain, all the available signals have to be filtered. Butterworth-filters will be used for this purpose. The configuration of all these conversions is displayed in figure 6.1.

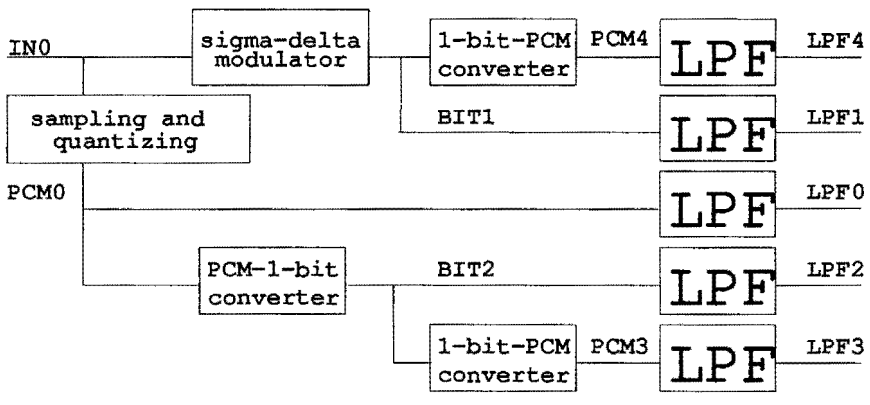

Figure 6.1: Configuration of the simulation blocks.

The signals in figure 6.1 can all be available in an agap-file, produced during the simulation. IN0 is the input signal taken from the input-file. BIT1 and BIT2 are the produced 1-bit signals, PCM0, PCM3 and PCM4 the PCM signals and LPF0 until LPF4 are the outputs of the respective low-pass filters.

A simulation can be made as follows:

```
sim bits,range,type config romtable sigma lpf input output
```

where:

- *bits* is the width of internal datapaths in bits which will be 'infinite' when *bits* is set to -1 or when *range* is 'infinite'.
- *range* is the internal calculation-range which will be 'infinite' when *range* is set to -1.
- *type* is one of the *characters* 'r', 't' or 'n' to round, truncate or to not round at all.
- *config* is a string with the indices of the signals to generate. A '0' means that the signals PCM0 and LPF0 will be generated, a '1' gives the signals BIT1 and LPF1 and so on. A complete simulation is performed with "01234" as the configuration-string. The user should take care for dependancies of the generated signals.
- *romtable* is the name of the COMBIC-description-file produced with "table" (See section 5.4).
- *sigma* is the filter-description-file to be used by the $\Sigma\Delta$-modulator (See subsection 5.2.4).
- *lpf* is the filter-description-file to be used to filter all the PCM and 1-bit signals. A Butterworth-filter could be used for this purpose (See subsection 5.2.3).
- *input* is the name of an agap-file describing an input-signal. Such a file, containing a sine-wave, can be generated with "sin" (See section 4.6).
- *output* is the name of an agap-file describing the created output-signals. When no name is given, standard-output will be used.

A typical simulation could be performed by:

```
sim -1,-1,n 01234 romtable sigma butterworth sine output
```

for 'infinite' internal calculation-accuracy and without range-limiting or using the same configuration

```
sim 14,40,r 01234 romtable sigma butterworth sine output
```

but now all the internal values are rounded to 14 bits in a range of 40.

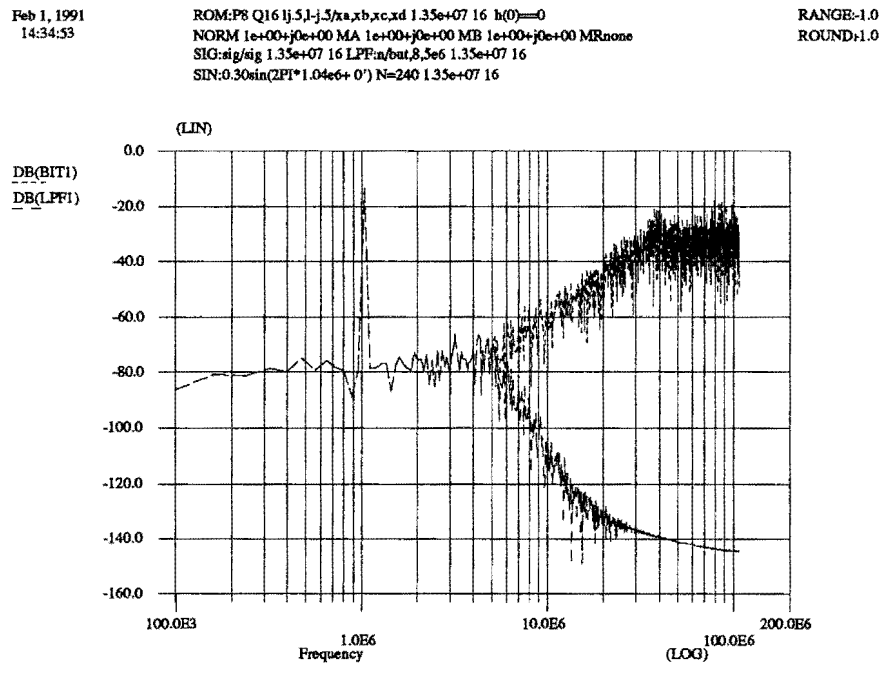

Figure 6.2: Spectrum of the output of the ΣΔ-modulator (upper curve) with a 1-bit symbol-rate of 216 MHz and a input sine of 1.04 MHz. The signal after low-pass filtering is shown too (lower curve) and has a RMS Signal/Noise-ratio of 42.9 dB.

6.3 The ΣΔ-modulator

The signals BIT1 and LPF1 have to be compared with IN0 and LPF0 to check the performance of the ΣΔ-modulator. It is important that a relatively high quality 1-bit signal is used when the performance of the COMBIC-"1-bit to PCM"-converter is tested. The error in the 1-bit signal should be small compared to the error introduced by the conversion.

The RMS Signal/Noise-ratio can be determined with "ugap", and it showed to be about 38 dB when using a 140 MHz 1-bit symbol-rate and a low-pass filter with a cutoff-frequency of 5 MHz (figure 6.2). Increase of the 1-bit symbol-rate to 216 MHz, improves the quality to a RMS Signal/Noise-ratio of 42.9 dB when using the same input-sine of 1.04 MHz. In section 6.4 this will show to be a limiting factor for the conversions, so the 1-bit signal can not be considered to be an ideal input-signal.

6.4 The COMBIC-"1-bit to PCM"-converter

The signal LPF4 has to be compared with LPF1 to check the performance of the "1-bit to PCM conversion", independently of the "PCM to 1-bit conversion". Some simulations where made while varying the poles of the filter and the internal calculation precision. Because feedback with PCM-precision caused low-quality conversions, the feedback was performed with the internal calculation precision. In figure 6.3 and figure 6.4 the filtered outputs of a ΣΔ-modulator and a COMBIC-"1-bit to PCM"-converter are shown, using

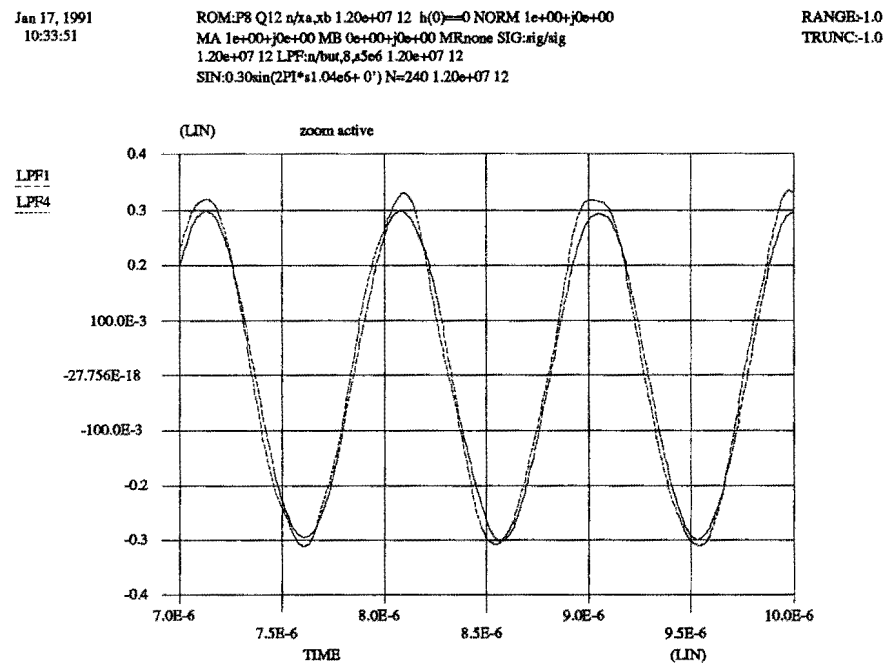

Figure 6.3: A 1-bit signal generated by a $\Sigma\Delta$-modulator with a RMS Signal/Noise-ratio of 38.3 dB and its 2^{nd} order COMBIC-converted PCM equivalent (LPF4), both filtered with a 8^{th} order Butterworth-filter with a normal frequency of 5 MHz.

COMBIC-converters of different orders. In figure 6.3 and figure 6.4 it can be easily seen that the filtered PCM-output of 4^{th} order COMBIC-"1-bit to PCM"-converter is approximating the original signal much better than when using the 2^{nd} order COMBIC-"1-bit to PCM"-converter. Some spectrums of conversions with COMBIC-"1-bit to PCM"-converters of different orders are shown in appendix B while table 6.1 lists the results. In figure 6.5 a graphical presentation is given of the conversion quality versus the order of the used internal filter. It can be seen that an increase of the order of the converter leads to better conversion-results until the quality of the original 1-bit signal is reached. For some filters, the conversion quality, under variation of the internally used bits, is shown in figure 6.6. Due to the values of the residues α_m and the subtraction of larger close-to-eachother values (section 3.2), a higher order converter can perform worse than expected. An increase of over-sampling results in a higher quality of the 1-bit signals, while the output of the COMBIC-converter follows this gain in Signal/Noise-ratio.

6.5 The COMBIC-"PCM to 1-bit"-converter

BIT2 is the output of the COMBIC-"PCM to 1-bit"-converter and LPF2 is its filtered equivalent. LPF2 should be close to the original signal, and its quality is a measure of the performance of the MAP-table. Depending on which internal filter was used, the MAP-table mostly saturated. When the MAP-table saturated, clipping occurred and the result (BIT2) was not valid at all. But a COMBIC-"PCM to 1-bit"-converter with poles at $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$ and$\frac{1}{2}$ and a pair of zeros at the imaginary axis which didn't saturate, still gave

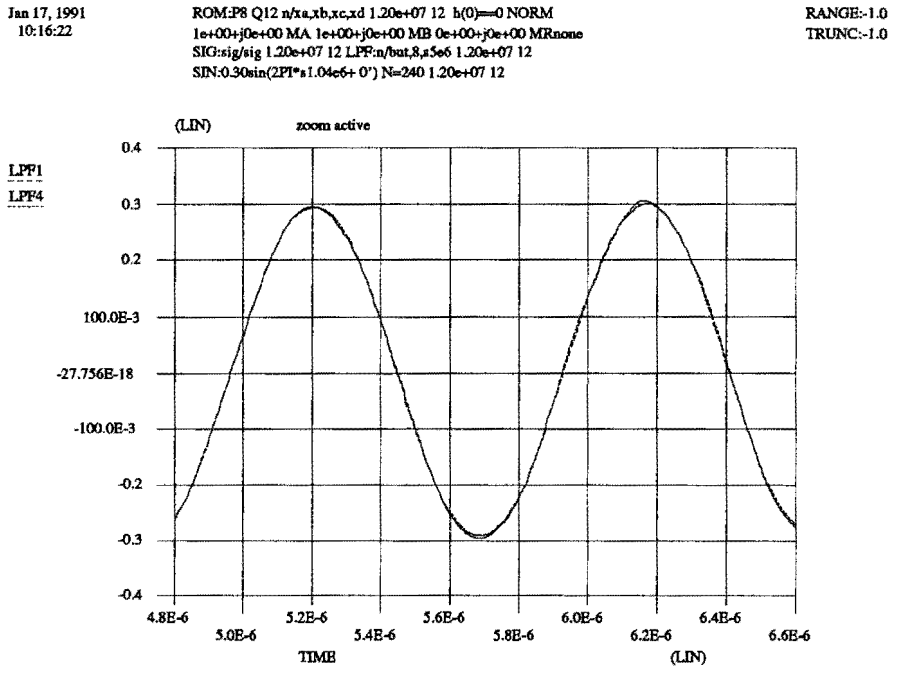

Figure 6.4: A 1-bit signal generated by a $\Sigma\Delta$-modulator with a RMS Signal/Noise-ratio of 38.3 dB and its 4^{nd} order COMBIC-converted PCM equivalent (LPF4), both filtered with a 8^{th} order Butterworth-filter with a normal frequency of 5 MHz.

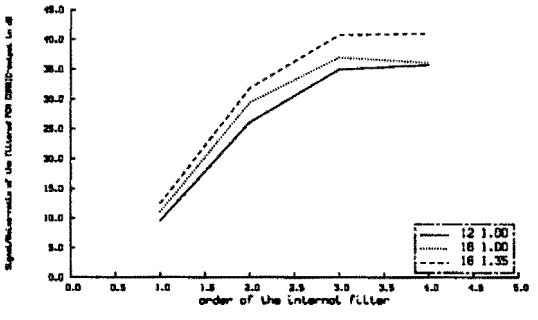

Figure 6.5: The RMS Signal/Noise-ratio of the filtered PCM output (LPF4) versus the order of the internally used filter of the COMBIC-"1-bit to PCM"-converter for several values of the relative 1-bit symbol-rate and oversampling-factors.

Table 6.1: Simulation-results of the COMBIC-"1-bit to PCM"-converter. The PCM sampling-rate is 12 MHz, the 1-bit symbol-rate 144 MHz and the input-signal was a sine of 1.04 MHz with an amplitude of 0.3 and no phase-shift. The 1-bit signal was generated by the predefined $\Sigma\Delta$-modulator, and has a RMS Signal/Noise-ratio of 38.3 dB (LPF1). The PCM output-signal was filtered (LPF4) with an 8^{th} order Butterworth-filter with its normal frequency at 5 MHz.

COMBIC-tables			Filter		Internal calculations			LPF4
P	Q	Rounding	Poles	Zeros	Range	Bits	Rounding	S/N
int	int	type	string	string	val	int	type	dB
8	12	round	abcd	none	-1	-1	none	35.6
8	12	round	abcd	none	12	16	round	35.5
8	12	round	abcd	none	12	15	round	35.1
8	12	round	abcd	none	12	14	round	35.2
8	12	round	abcd	none	12	13	round	34.1
8	12	round	abcd	none	12	12	round	29.7
8	12	round	abcd	none	12	11	round	26.0
8	12	round	abcd	none	12	10	round	19.4
8	12	round	abcd	none	12	9	round	14.8
8	12	round	abcd	none	12	16	trunc	35.5
8	12	round	abcd	none	12	14	trunc	35.2
8	12	round	abcd	none	12	12	trunc	28.9
8	12	round	abcd	none	12	10	trunc	17.0
8	12	round	abcd	none	12	9	trunc	13.0
8	12	round	abc	none	-1	-1	none	34.9
8	12	round	abc	none	4	16	round	35.0
8	12	round	abc	none	4	14	round	35.0
8	12	round	abc	none	4	12	round	34.3
8	12	round	abc	none	4	10	round	30.4
8	12	round	abc	none	4	8	round	21.3
8	12	round	ab	none	-1	-1	none	26.1
8	12	round	a	none	-1	-1	none	9.5

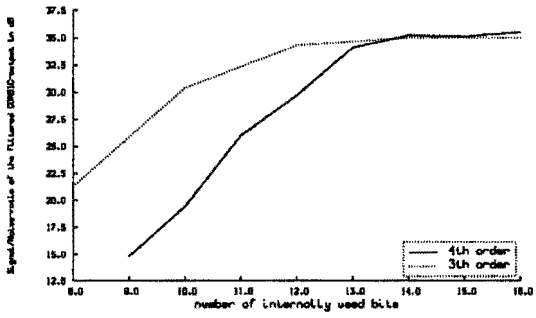

Figure 6.6: The RMS Signal/Noise-ratio of the filtered PCM output (LPF4) versus the number of internally used bits for a 4^{th} and a 3^{th} order COMBIC-"1-bit to PCM"-converter.

poor results with high noise-levels in the upper regions of the PCM signal-band.

Where the rounding-error introduced in the feedback of the COMBIC-"1-bit to PCM"-converter showed to be of critical importance, the rounding-error introduced by the MAP-table could cause a similar error in the conversion from PCM to 1-bit signals .

6.6 The COMBIC-"PCM–1-bit–PCM"-chain

The signals BIT2, LPF2 and LPF3 can be compared with IN0 and LPF0 to check the performance of the chained conversions from PCM to 1-bit and back to PCM. The COMBIC-"PCM–1-bit–PCM"-chain, which does not check the validity of the intermediate 1-bit signal, was used in the program described in [4] and appendix C. After checking the new software to be identical to the existing pascal-program, the COMBIC-"PCM–1-bit–PCM"-chain was not of great importance anymore. The conversions showed to be eachothers inverses (almost exact), only influenced by the occuring roundings during calculations and possibly the saturation of the intermediate 1-bit stream.

Chapter 7

Conclusions and recommendations

7.1 Introduction

Simulations showed that the COMBIC-algorithm itself is correct for the conversion for 1-bit signals to PCM signals. The conversion quality depends on the chosen filter parameters. Later on, it was found that the rounding of the internal feedback signal easily makes the converter instable. The rounding-effect of the MAP-table could be the cause of the poor results for the conversion from PCM to 1-bit signals. Because the format-conversion in this direction is not the most interesting of the two, it is left for later studies.

7.2 The COMBIC-algorithm

Although the COMBIC-algorithm showed to be usable for conversions of 1-bit signals to PCM signals The measured Signal/Noise-ratio after the conversion was still a bit less than calculations (see [4]) foredicted, but of a satisfying quality for the tested video applications. The difference can be caused by the use of a less band-limiting filter with n leaky integrating sections instead of an ideal n^{th} order integrator.

The implementation of COMBIC-converters, starts with the choice of the internally used filter. Principally, this choice is only restricted by the needed band-limitation. The choice of the positions of zeros and poles s_m is quite free, but to obtain a simple implementation the placement of the poles is restricted such that the terms $e^{s_m T}$ are negative powers of two.

Within these restrictions it is still possible to realize converters approximating the theoretic calculations for ideal integrators of different order. Simulations showed that increasing the order, results in a gain of conversion-quality, conform the theoretic calculations, although the higher order COMBIC-converters do not reach their expected precision entirely. The restriction to only single poles, obligates the use of leaky integrators which decreases the band-limitation of the filter and therefore the performance of conversion.

Another cause is the increase of the residues α_m (positive an negative) for higher order filters, which leads to the subtraction of bigger close-to-eachother numbers. The needed calculation-range will be larger for higher-order filters while the calculation-results can come closer to the quantization-step when using the same number of internal bits.

7.3 Complex poles

Because the internal filter was firstly found to be non-satisfying when it had only real poles, it was decided to examine the use of complex poles. For this the simulation software has been changed. A circuit, performing the COMBIC-algorithm, can still be simple as long as RE and IM in $RE + jIM = e^{s_m T}$ are simple values. This extra dimension of freedom in the choice of a filter could lead to better implementations but makes the determination of an optimal configuration more complicated.

7.4 Feedback

Rounding of the feedback-terms in the COMBIC-converters for conversion to PCM format caused limit-cycles, when simulating with a sine as input-signal. When the feedback was performed with the internal calculation precision, no limit-cycles occurred and the conversion became much better. Unfortunately, this principally caused the need for larger Lookup-tables.

7.5 Not using Lookup-tables for simple calculations

When $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$ and$\frac{1}{2}$ are used as the recursive multiplication-factors $e^{s_m T}$, the $B_{m(j)}$-tables perform multiplications with simple values for the residues α_m which can be done with a bit-wise shift-operation too. A multiplication with for example 96 (which equals $2^5 + 2^6$) can easily being implemented as the addition of two bit-wise shifts. While the $B_{m(j)}$-tables should be increased in size by performing the feedback in the COMBIC-"1-bit to PCM"-converter with the internal precision, they could be left out and replaced by full-adders when using the specific multiplication-factors.

7.6 COMBIC Design-steps

Before starting with the design of a COMBIC-converter, some considerations are to be made about the PCM and 1-bit signals concerned to their formats and frequencies. As soon as these parameters are fixed by the specification of a $\Sigma\Delta$-modulator, signal-qualities and bandwidth-requirements, a COMBIC-converter can be designed.

When designing a COMBIC-converter the following path of decisions should be used:

- Choice of the order of the internal filter. It is recommended to use poles with their position fixed by $e^{s_m T}$ equaling $\frac{1}{2}$, $\frac{1}{4}$ and so on in that order. The use of the less leaky integrating sections will increase the performance.
- Choice of the number of internally used calculation bits. A good start can be made by using the same number of bits as used for a sequence of 1-bit symbols during a PCM-sampling period.

- Because the relative values of residues α_m of the filter are determined by the number and positions of the poles, an extra pole could cause a gain of performance which is smaller than expected so the comparison of filters with different order and the same internal calculation precision could pay.
- The choice between better-performing rounding-off and simpler-to-implement truncating should be taken with care. Simulations with sine-waves caused limit-cycles when using rounding-off for feed-back signals, while this is much more unlikely to happen, when using non-periodic signals. Simulations with non-periodic signals could bring clarity to this subject.

Bibliography

[1] G.H. Hostetter, C.J. Savant Jr., and R.T. Stefani. *Design of feedback control systems.* Holt-Saunders International Editions, New York, 1982.

[2] E. Roza. A practical design approach to decision feedback receivers with conventional filters. *IEEE Transactions on communications*, COM(26):679–689, May 1978.

[3] E. Roza. Bi-directionele conversie van 1-bits-formaat naar pcm-formaat. January 1989. Ref.no.:20,088.

[4] Chen Meng-Hui. *Analysis of a new algorithm for the digital format conversion between 1-bit signal and PCM signal.* Technical Report, Philips Research Laboratories, June-August 1989.

[5] A.W.M. van den Enden and N.A.M. Verhoeckx. *Discrete-time signal processing, an introduction.* Prentice Hall, New York, 1989.

[6] Brian W. Kernighan and Dennis M. Ritchie. *The C programming language.* Prentice Hall, New York, second edition, 1988.

Appendix A

Derivation of the COMBIC-equations

A.1 Filtering PCM and 1-bit signals

The PCM signal $x_{pcm}[jT]$ can be represented by,

$$x_{pcm}[jT] = \sum_{k=0}^{\infty} b_k \delta[jT - kT]$$

where b_k are the PCM samples and k and j are integers.

The 1-bit signal with the symbol period $\frac{T}{q}$ can be presented by,

$$x_{1bit}[jT] = \sum_{n=0}^{\infty} a_n \delta[jT - n\frac{T}{q}]$$

where $a_n = 1$ or -1, n is an integer, and q is an integer larger than 1.

Considering a causal filter with an impulse response $h[jT]$ and an input-signal $x[jT]$ the output of the filter $y[jT]$ can be described by:

$$\begin{aligned} y[jT] &= x[jT] * h[jT] \\ &= \sum_{k=0}^{j} x[kT] \times h[jT - kT] \end{aligned}$$

where $*$ denotes convolution.

If the input $x[jT]$ is the PCM signal $x_{pcm}[jT]$, then the output $y_{pcm}[jT]$ is:

$$\begin{aligned} y_{pcm}[jT] &= h[jT] * x_{pcm}[jT] \\ &= h[jT] * \sum_{k=0}^{\infty} b_k \delta[jT - kT] \\ &= \sum_{k=0}^{j} b_k h[jT - kT] \end{aligned}$$

If the input $x[jT]$ is the 1-bit signal $x_{1bit}[jT]$, then the output $y_{1bit}[jT]$ is:

$$\begin{aligned} y_{1bit}[jT] &= h[jT] * x_{1bit}[jT] \\ &= h[jT] * \sum_{n=0}^{\infty} a_n \delta[jT - n\frac{T}{q}] \\ &= \sum_{n=0}^{jq} a_n h[jT - n\frac{T}{q}] \end{aligned}$$

A.2 The difference-equation

Let's define:

$$h[jT] - h[(j-1)T] = \Delta h[jT] \tag{A.1}$$

Then the differences between two output signals at time instants jT and $(j+1)T$ are as follows:

For the PCM input signal:

$$\begin{aligned} y_{pcm}[jT] &= \sum_{k=0}^{j} b_k h[jT - kT] \\ y_{pcm}[(j+1)T] &= \sum_{k=0}^{j+1} b_k h[(j+1)T - kT] \end{aligned}$$

$$\begin{aligned} y_{pcm}[(j+1)T] - y_{pcm}[jT] &= b_{j+1} h[0] + \sum_{k=0}^{j} b_k \{h[(j+1)T - kT] - h[jT - kT]\} \\ &= b_{j+1} h[0] + \sum_{k=0}^{j} b_k \times \Delta h[(j+1)T - kT] \end{aligned} \tag{A.2}$$

For the 1-bit input signal:

$$\begin{aligned} y_{1bit}[jT] &= \sum_{n=0}^{jq} a_n h[jT - n\frac{T}{q}] \\ y_{1bit}[(j+1)T] &= \sum_{n=0}^{(j+1)q} a_n h[(j+1)T - n\frac{T}{q}] \\ y_{1bit}[(j+1)T] - y_{1bit}[jT] &= \sum_{n=0}^{jq} a_n \{h[(j+1)T - n\frac{T}{q}] - h[jT - n\frac{T}{q}]\} \\ &\quad + a_{jq+1} h[(j+1)T - (jq+1)\frac{T}{q}] \\ &\quad + a_{jq+2} h[(j+1)T - (jq+2)\frac{T}{q}] \\ &\quad \vdots \end{aligned}$$

$$
\begin{aligned}
& + a_{jq+q} h[(j+1)T - (jq+q)\frac{T}{q}] \\
= & \sum_{n=0}^{jq} a_n \{ h[(j+1)T - n\frac{T}{q}] - h[jT - n\frac{T}{q}] \} \\
& + a_{jq+1} h[T - \frac{T}{q}] \\
& + a_{jq+2} h[T - 2\frac{T}{q}] \\
& \vdots \\
& + a_{jq+q} h[T - q\frac{T}{q}] \\
= & \sum_{n=0}^{jq} a_n \Delta h[(j+1)T - n\frac{T}{q}] \\
& + \sum_{i=1}^{q} a_{qj+i} h[T - i\frac{T}{q}] \qquad \text{(A.3)}
\end{aligned}
$$

Considering that, the sampled response of 1-bit signals must be equal to the sampled response of PCM signals, a difference-equation can be written, which means equation (A.2) = equation (A.3):

$$y_{pcm}[(j+1)T] - y_{pcm}[jT] = y_{1bit}[(j+1)T] - y_{1bit}[jT]$$

and it follows that

$$
\begin{aligned}
b_{j+1} h[0] = & \sum_{i=1}^{q} a_{qj+i} h[T - i\frac{T}{q}] \\
& + \sum_{n=0}^{jq} a_n \Delta h[(j+1)T - n\frac{T}{q}] \\
& - \sum_{k=0}^{j} b_k \Delta h[(j+1)T - kT] \qquad \text{(A.4)}
\end{aligned}
$$

Another way to write equation (A.4) is

$$
\begin{aligned}
b_{j+1} h[0] = & \sum_{i=1}^{q} a_{qj+i} h[T - i\frac{T}{q}] \\
& + \sum_{n=0}^{jq} a_n \{ h[(j+1)T - n\frac{T}{q}] - h[jT - n\frac{T}{q}] \} \\
& - \sum_{k=0}^{j} b_k \{ h[(j+1)T - kT] - h[jT - kT] \} \qquad \text{(A.5)}
\end{aligned}
$$

From equation (A.4), the PCM samples b_j can be determined iteratively. However, we have to recalculate the coefficients $\Delta h(t)$ periodically. Although it is not required to extend this calculation over a larger interval than the duration of the impulse response, the procedure requires nevertheless a large amount of calculations.

A.3 Filter simplification

In order to simplify equation (A.5), it is assumed that $h(t)$ is the impulse response of a causal filter with M single poles. Then $h(t)$ can be expressed as a sequence of exponential functions:

$$h(t) = U(t) \sum_{m=0}^{M-1} \alpha_m e^{s_m t} \tag{A.6}$$

where $U(t)$ is the unit step function and $\mathrm{Re}[s_m] < 0$.

The following terms of equation (A.5) can be derived with equation (A.6):

For the 1-bit signal:

$$\begin{aligned} & h[(j+1)T - n\frac{T}{q}] - h[jT - n\frac{T}{q}] \\ &= \sum_{m=0}^{M-1} \alpha_m e^{s_m[(j+1)T - n\frac{T}{q}]} - \sum_{m=0}^{M-1} \alpha_m e^{s_m[jT - n\frac{T}{q}]} \\ &= \sum_{m=0}^{M-1} \{\alpha_m[e^{s_m T} - 1] \times e^{s_m(jT - n\frac{T}{q})}\} \end{aligned} \tag{A.7}$$

For the PCM signal:

$$\begin{aligned} & h[(j+1)T - kT] - h[jT - kT] \\ &= \sum_{m=0}^{M-1} \alpha_m e^{s_m[(j+1)T - kT]} - \sum_{m=0}^{M-1} \alpha_m e^{s_m[jT - kT]} \\ &= \sum_{m=0}^{M-1} \{\alpha_m[e^{s_m T} - 1] \times e^{s_m(jT - kT)}\} \end{aligned} \tag{A.8}$$

And in general:

$$h[0] = \sum_{m=0}^{M-1} \alpha_m \tag{A.9}$$

Substituting equation (A.7), equation (A.8) and equation (A.9) into equation (A.5) gives:

$$\begin{aligned} & b_{j+1} \sum_{m=0}^{M-1} \alpha_m \\ &= \sum_{n=1}^{q} a_{qj+n} \sum_{m=0}^{M-1} \alpha_m e^{s_m(T - n\frac{T}{q})} \\ &\quad + \sum_{n=0}^{jq} a_n \times \sum_{m=0}^{M-1} \{\alpha_m[e^{s_m T} - 1] \times e^{s_m(jT - n\frac{T}{q})}\} \\ &\quad - \sum_{k=0}^{j} b_k \times \sum_{m=0}^{M-1} \{\alpha_m[e^{s_m T} - 1] \times e^{s_m(jT - kT)}\} \end{aligned} \tag{A.10}$$

and rearranging equation (A.10) gives

$$\begin{aligned}
&\sum_{m=0}^{M-1} \alpha_m b_{j+1} \\
&= \sum_{m=0}^{M-1} \alpha_m \sum_{n=1}^{q} a_{qj+n} e^{s_m(T-n\frac{T}{q})} \\
&\quad +\alpha_m \sum_{m=0}^{M-1} [1 - e^{s_m T}] \times \\
&\quad [\sum_{k=0}^{j} b_k e^{s_m(jT-kT)} - \sum_{n=0}^{jq} a_n e^{s_m(jT-n\frac{T}{q})}]
\end{aligned} \tag{A.11}$$

which means that the PCM samples can be deduced from the 1-bit input vectors in a recursive way.

A.4 Using Lookup-tables

In section A.3 it is shown that, using a causal filter with M single poles, the PCM samples b_j can be found according to

$$\sum_{m=0}^{M-1} B_{m(j+1)} = \sum_{m=0}^{M-1} A_{m(j)} + \sum_{m=0}^{M-1} \Gamma_{m(j)}[1 - e^{s_m T}] \tag{A.12}$$

where

$$\begin{aligned}
B_{m(j+1)} &= \alpha_m b_{j+1} \\
A_{m(j)} &= \alpha_m \times \sum_{n=1}^{q} a_{qj+n} e^{s_m(T-n\frac{T}{q})} \\
\Gamma_{m(j)} &= \alpha_m \{\sum_{k=0}^{j} b_k e^{s_m(jT-kT)} \\
&\quad - \sum_{n=0}^{jq} a_n e^{s_m(jT-n\frac{T}{q})}\} \\
&= e^{s_m T} \times \alpha_m \{\sum_{k=0}^{j-1} b_k e^{s_m((j-1)T-kT)} \\
&\quad - \sum_{n=0}^{(j-1)q} a_n e^{s_m((j-1)T-n\frac{T}{q})}\} \\
&\quad -\alpha_m \times \sum_{n=(j-1)q+1}^{jq} a_n e^{s_m(jT-n\frac{T}{q})} + \alpha_m b_j \\
&= \Gamma_{m(j-1)} \times e^{s_m T} - A_{m(j-1)} + B_{m(j)}
\end{aligned}$$

If the impulse-response $h[t]$ is chosen such that:

$$h[0] = \sum_{m=0}^{M-1} \alpha_m = 0$$

the equations have to be rewritten after filling in this statement and then:

$$\sum_{m=0}^{M-1} B_{m(j)} = \sum_{m=0}^{M-1} A_{m(j)} + \sum_{m=0}^{M-1} \Gamma_{m(j)}[1 - e^{s_m T}] \tag{A.13}$$

where

$$\begin{aligned} B_{m(j)} &= \alpha_m b_j e^{s_m T} \\ A_{m(j)} &= \alpha_m \times \sum_{n=0}^{q-1} a_{qj+n} e^{s_m (T - n\frac{T}{q})} \\ \Gamma_{m(j)} &= \Gamma_{m(j-1)} \times e^{s_m T} - A_{m(j-1)} + B_{m(j-1)} \end{aligned}$$

It can be easily seen that the system-dependent terms $B_{m(j)}$ and $A_{m(j)}$ can be put in Lookup-tables where the bit-representations of the PCM and 1-bit signals are used as addresses. A COMBIC-"1-bit to PCM"-converter using these Lookup-tables is displayed in figure 2.9.

For the "PCM to 1-bit conversion" some similar equations can be derived by rewriting equation (A.11) :

If the internal filter $h[t]$ is chosen such that:

$$h[0] = \sum_{m=0}^{M-1} \alpha_m \neq 0$$

then

$$\sum_{m=0}^{M-1} A_{m(j)} = \sum_{m=0}^{M-1} B_{m(j+1)} + \sum_{m=0}^{M-1} \Gamma_{m(j)}[1 - e^{s_m T}] \tag{A.14}$$

where

$$\begin{aligned} B_{m(j+1)} &= \alpha_m b_{j+1} \\ A_{m(j)} &= \alpha_m \times \sum_{n=1}^{q} a_{qj+n} e^{s_m (T - n\frac{T}{q})} \\ \Gamma_{m(j)} &= \Gamma_{m(j-1)} \times e^{s_m T} + A_{m(j-1)} - B_{m(j)} \end{aligned}$$

and if the internal filter $h[t]$ is chosen such that:

$$h[0] = \sum_{m=0}^{M-1} \alpha_m = 0$$

then

$$\sum_{m=0}^{M-1} A_{m(j)} = \sum_{m=0}^{M-1} B_{m(j)} + \sum_{m=0}^{M-1} \Gamma_{m(j)}[1 - e^{s_m T}] \tag{A.15}$$

where

$$\begin{aligned} B_{m(j+1)} &= \alpha_m b_{j+1} e^{s_m T} \\ A_{m(j)} &= \alpha_m \times \sum_{n=0}^{q-1} a_{qj+n} e^{s_m (T - n\frac{T}{q})} \\ \Gamma_{m(j)} &= \Gamma_{m(j-1)} \times e^{s_m T} + A_{m(j-1)} - B_{m(j-1)} \end{aligned}$$

The use of Lookup-tables for the COMBIC-"PCM to 1-bit"-converter is shown in figure 2.11.

Appendix B

Results

B.1 Varying the order of the internal filter

Some simulations have been performed with an increasing order of the internally used filter and it can be seen that this leads to a gain of quality of the output-signal of the COMBIC-converters. The following configuration-description was used with only a variation of the poles of the internal COMBIC-filter.

```
mlpf 1 R xa,xb,xc,xd none 12e6 12 combic
```

The PCM sampling-rate is 12 MHz and the 1-bit sampling-rate is 144 MHz. The COMBIC-filterhas no zeros and poles are chosen such that the internally used feedback-factors $e^{s_m T}$ are of the set $\frac{1}{2}$ ('xa'), $\frac{1}{4}$ ('xb'), $\frac{1}{8}$ ('xc') and $\frac{1}{16}$ ('xd').

```
table 8,.5,r 12 combic y n y n romtable
```

The COMBIC-conversion is performed with 8 bits PCM signals, rounded in a range $[-0.5, 0.5]$, at a sampling-rate of 12 MHz, and a 1-bit sampling-rate of 144 MHz, the Lookup-tables are normalized and the $B_{m(j)}$-tables are skipped in the feedback.

```
mlpf 1 R sig sig 12e6 12 sigma
```

For the $\Sigma\Delta$-modulator, a filter with a structure as described in subsection 5.2.4 was used.

```
mlpf 1 R but,8,5e6 none 12e6 12 butterworth
```

The low-pass-filtering is performed with a 8^{th} order Butterworth-filter with a normal frequency of 5 MHz.

```
sin 18e-6 0.3 1.04e6 0 12e6 12 sine
```

The input-signal is a sine of 1.04 MHz and an amplitude of 0.3 without a phase-shift over 240 PCM sampling-periods.

```
sim -1,-1,n 01234 romtable sigma butterworth sine output
```

The RMS Signal/Noise-ratios are calculated with "ugap" with a rectangular window from 5 until 17.5 μsecs where the signal-spectrum was taken from 0.9 MHz until 1.2 MHz and the noise-spectrum from 0 until 5 MHz. In figure B.1, figure B.2, figure B.3 and figure B.4 the results of some simulations with COMBIC-converters of several orders are shown.

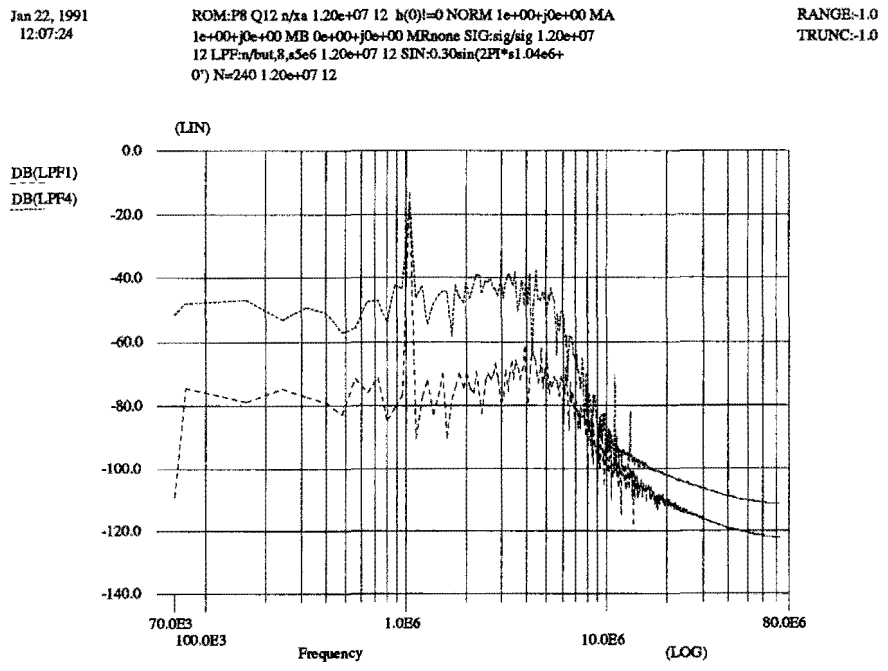

Figure B.1: Conversion of a 1-bit signal with a RMS Signal/Noise-ratio of 38.3 dB, with a 1^{th} order COMBIC-"1-bit to PCM"-converter using 'infinite' calculation-precision gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 9.5 dB.

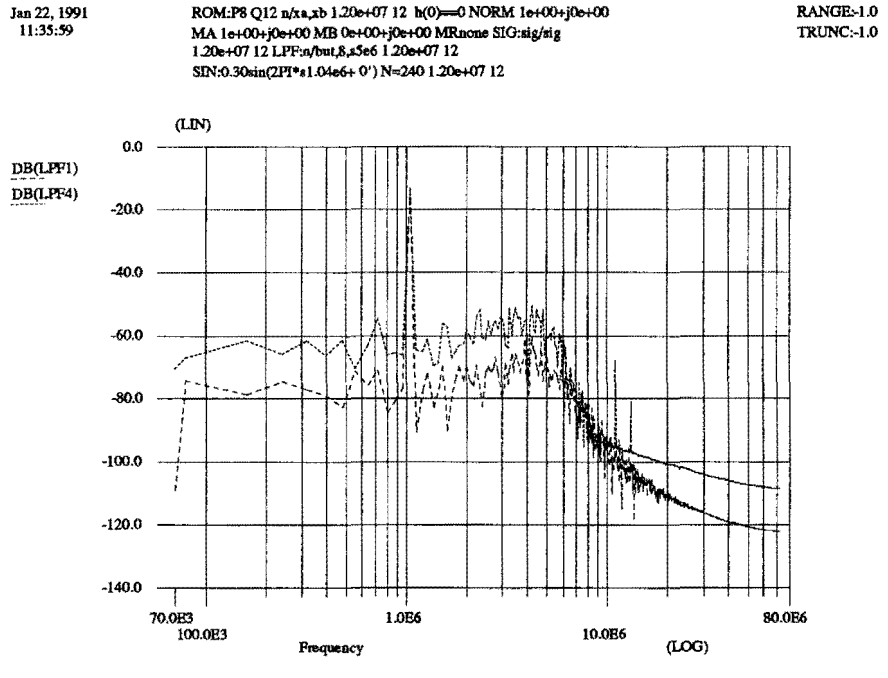

Figure B.2: Conversion of a 1-bit signal with a RMS Signal/Noise-ratio of 38.3 dB, with a 2^{th} order COMBIC-"1-bit to PCM"-converter using 'infinite' calculation-precision gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 26.1 dB.

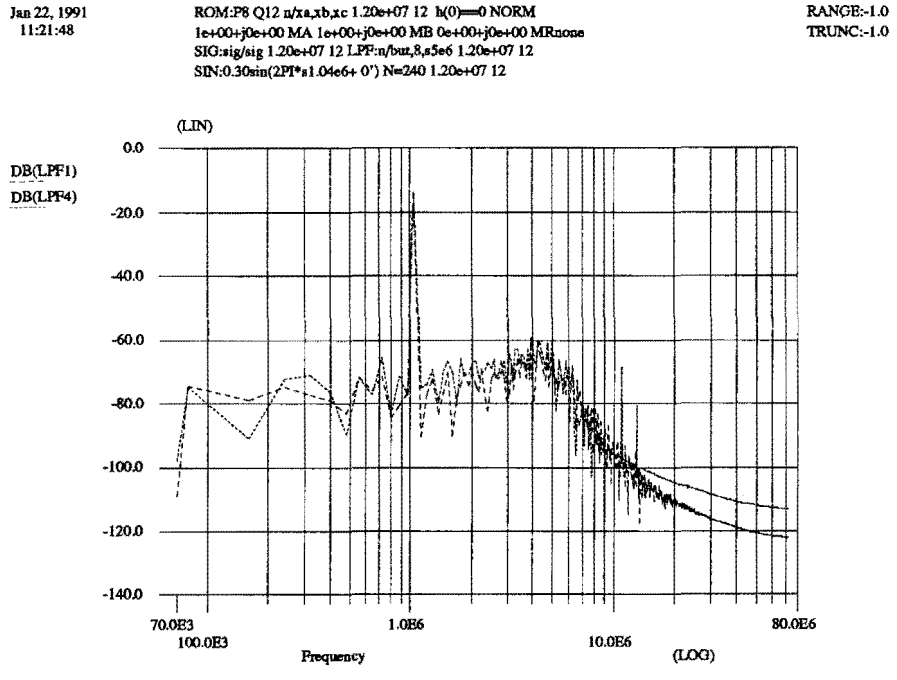

Figure B.3: Conversion of a 1-bit signal with a RMS Signal/Noise-ratio of 38.3 dB, with a 3^{th} order COMBIC-"1-bit to PCM"-converter using 'infinite' calculation-precision gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 34.9 dB.

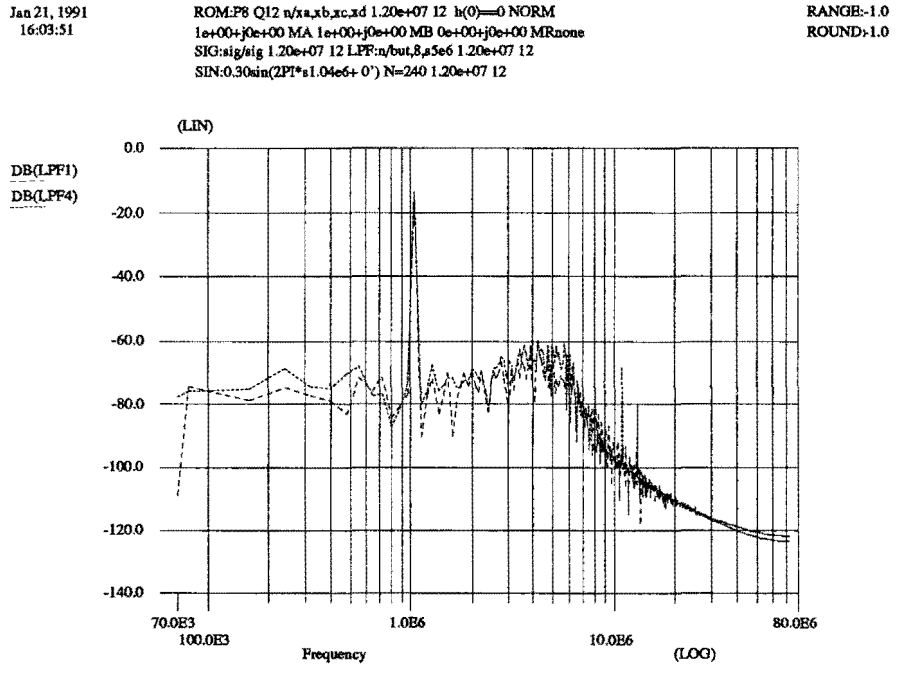

Figure B.4: Conversion of a 1-bit signal with a RMS Signal/Noise-ratio of 38.3 dB, with a 4^{th} order COMBIC-"1-bit to PCM"-converter using 'infinite' calculation-precision gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 35.6 dB.

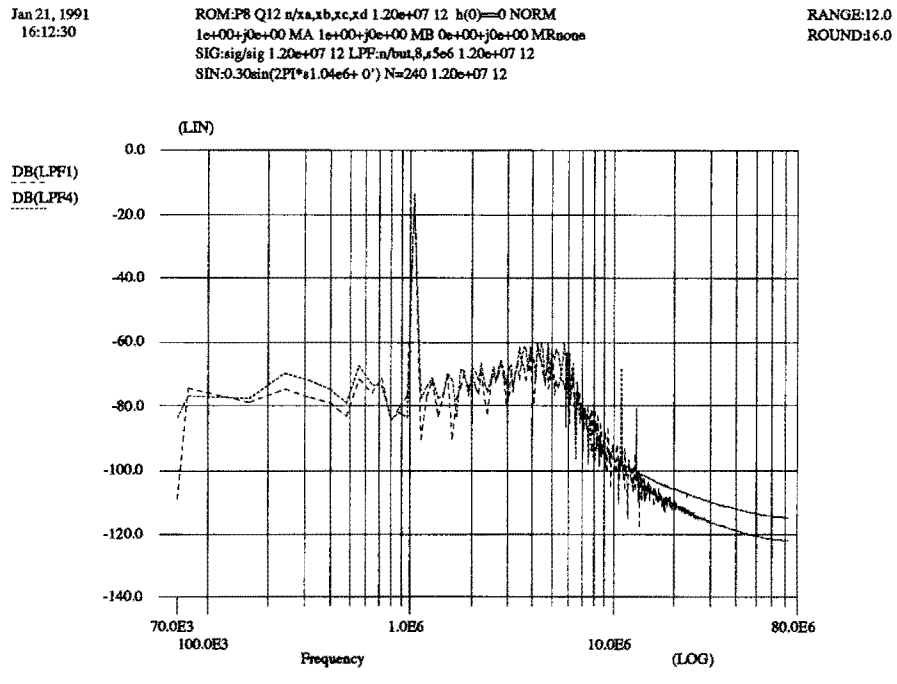

Figure B.5: Conversion of a 38.3 dB 1-bit signal, by a COMBIC-"1-bit to PCM"-converter using internal rounding to 16 bits gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 35.5 dB.

B.2 Varying internal calculation accuracy

In figure B.4 the conversion with 'infinite' calculation-precision is shown and this simulation can be used as a reference. In the following figures the results with other internal calculation-precisions are displayed.

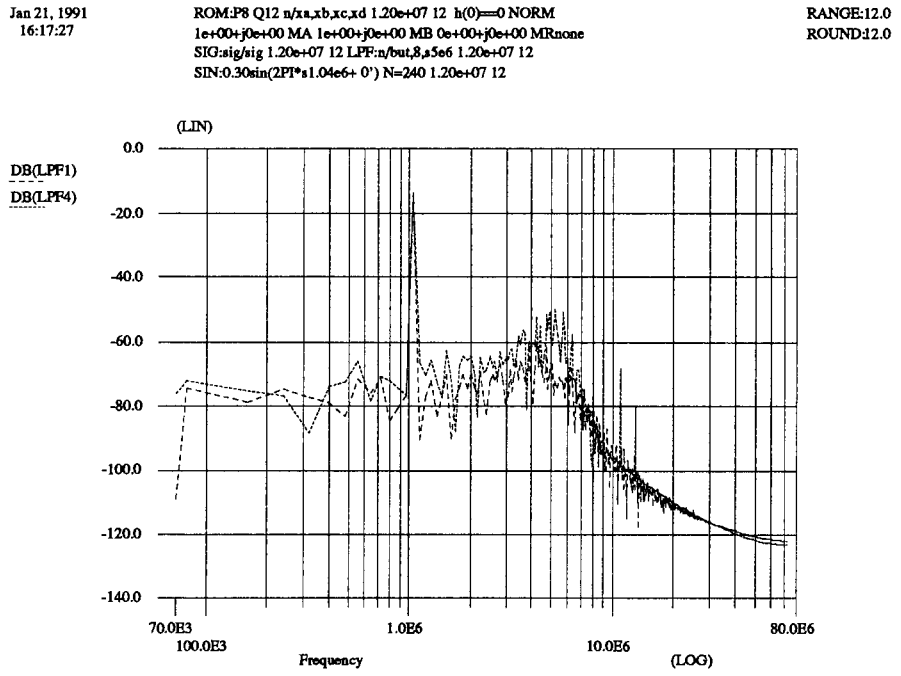

Figure B.6: Conversion of a 38.3 dB 1-bit signal, by a COMBIC-"1-bit to PCM"-converter using internal rounding to 12 bits gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 29.7 dB.

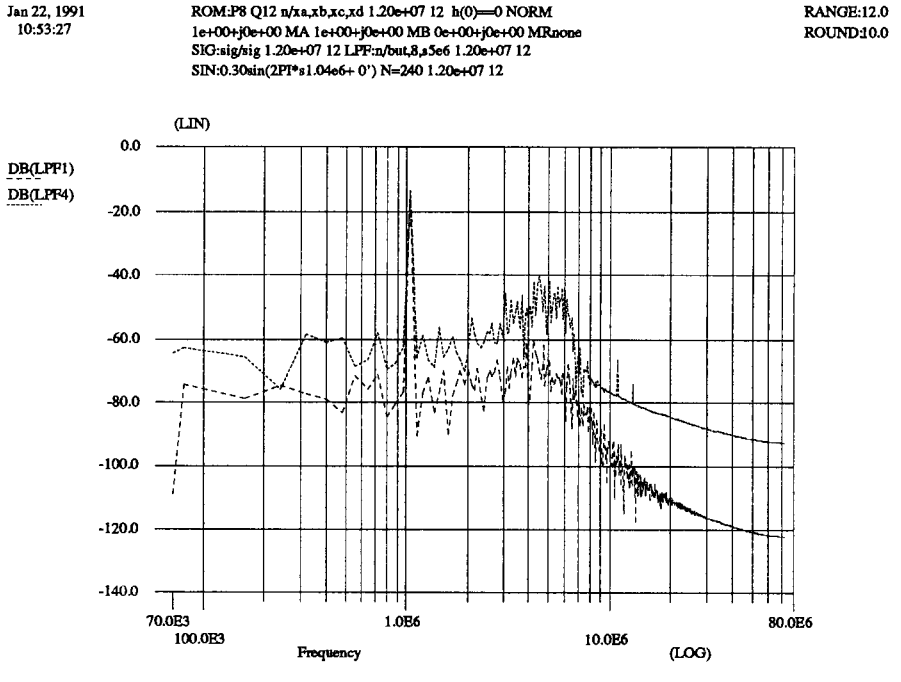

Figure B.7: Conversion of a 38.3 dB 1-bit signal, by a COMBIC-"1-bit to PCM"-converter using internal rounding to 10 bits gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 19.4 dB.

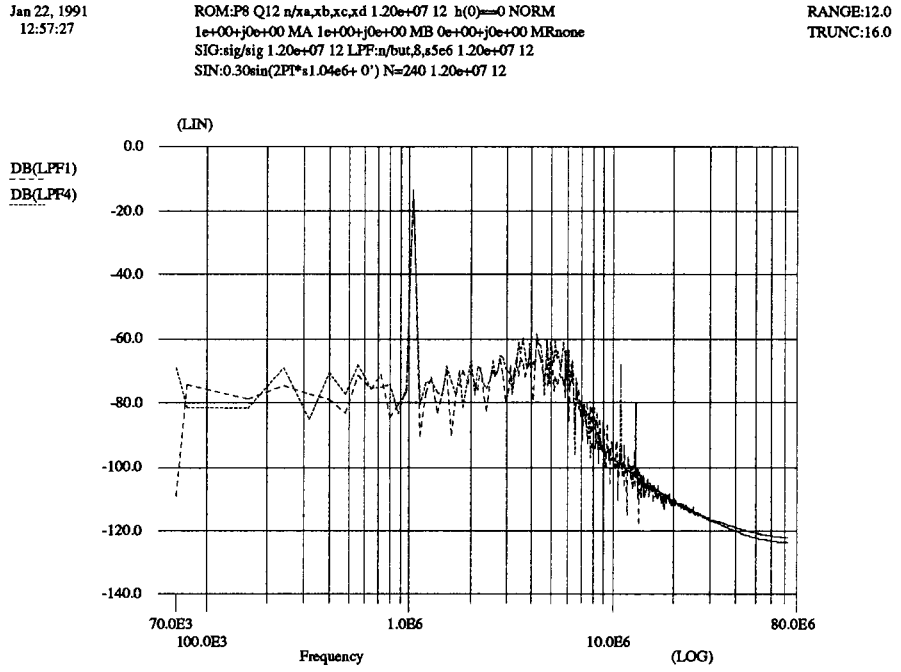

Figure B.8: Conversion of a 38.3 dB 1-bit signal, by a COMBIC-"1-bit to PCM"-converter using internal truncation to 16 bits gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 35.5 dB.

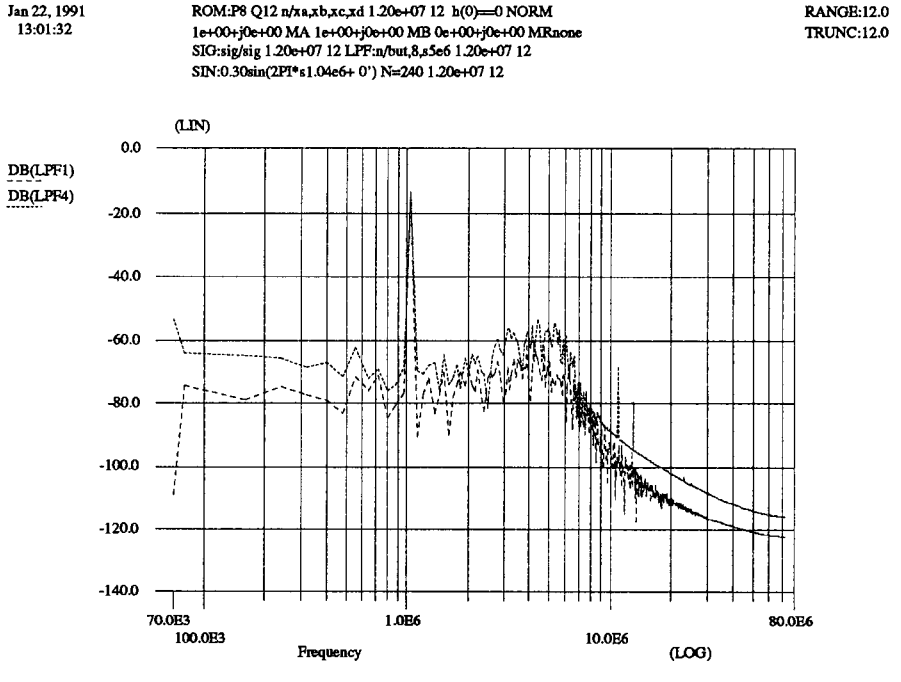

Figure B.9: Conversion of a 38.3 dB 1-bit signal, by a COMBIC-"1-bit to PCM"-converter using internal truncation to 12 bits gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 28.9 dB.

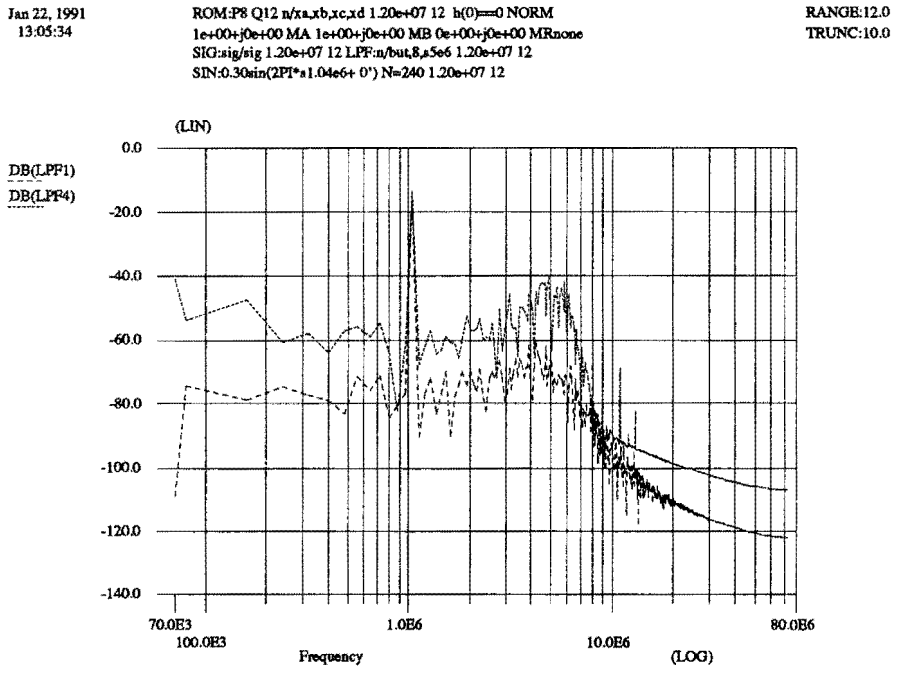

Figure B.10: Conversion of a 38.3 dB 1-bit signal, by a COMBIC-"1-bit to PCM"-converter using internal truncation to 10 bits gives a PCM signal (LPF4) with a RMS Signal/Noise-ratio of 17.0 dB.

Appendix C

The original simulation program

C.1 Introduction

In the summer of 1989, a simulation-program has been written to study the proposed COMBIC-algorithm (See appendix A and [4]). This program was written in Pascal, which is not a very common-used programming language in the group. For this and to get known with the subject the program was converted to C. At first, a direct translation took place to be sure that the produced output would be the same, supposing identical calculation precision and rounding. Later on, the software was divided in separate modules, forming the COMBIC-simulation package as shown in figure 5.1.

C.2 Possibilities of the original program

Several functions needed for simulations of the algorithm were provided by the program:

- The generation of the needed Lookup-tables, according to the entered parameters. The parameters and the tables are dumped in a file.
- The generation of PCM samples of a user-defined sine wave. A file containing the samples is generated.
- The simulation of the algorithm itself with the chosen Lookup-tables and PCM input using 'infinite' calculation precision. Both the conversions are chained to a so-called COMBIC-"PCM–1-bit–PCM"-chain for this purpose, so it is possible to make a direct comparison of the PCM output with its original PCM signal.

An advantage of this configuration is that the PCM input samples are much easier to generate than its 1-bit equivalent. It is disadvantage of using the COMBIC-"PCM–1-bit–PCM"-chain, that there is no check for the validity of the intermediate 1-bit signal. The results are dumped in a text-file, which is not easy to analyse in the time-domain while spectral analyses are even impossible. Dumping the resulting signals in agap-files, gives the opportunity to view the signals graphicly with "agap" and to calculate spectrums using Fast Fourier Transformation with "ugap".

C.3 Bugs in the original program

C.3.1 Scaling

First simulations with the original COMBIC-"PCM–1-bit–PCM"-chain showed that the so-called "dynamical range" was quite small and definitely not satisfying. Here, the "dynamical range" is the range in which the PCM input can vary without saturating the 1-bit symbols output-sequence. This was caused by the chosen signal format scales; the response of the internal filter to the PCM signal was not in the same order as its response to the 1-bit signal. Besides the introduction of scaling based on energy-considerations, in the newly developed software, multiplication-factors are implemented as parameters which can be defined by the user at run-time.

C.3.2 Sorting

In the original program, the sorting of the MAP-table didn't work perfectly which resulted in strange events when the simulated input signals reached the corresponding MAP values. Rounding errors and limit cycles occurred because the routine looking for the closest value considers the MAP-table to be a sorted list. In the C software the standard-function "qsort" is used to sort the MAP-table and the MAP-table can be displayed graphicly with "agap" to check its contents.

C.4 Differences caused by the conversion from Pascal to C

The original Pascal program used several mathematical functions which are not directly provided by C. For this, functions like rounding and truncation of values had to be defined (See [6]). For a later study regarding to the needed word-lengths, a rounding function with the calculation *range*, the word-length in *bits* and the *type* of rounding as parameters, has been implemented in the new software.

Appendix D

Reducing the MAP-table

D.1 Introduction

Unfortunately, a reduction as used for the $A_{m(j)}$-tables (section 2.10), can not be made with the MAP-table, because it is sorted in increasing order and therefore the addresses are not directly related to the input-vectors.

Since the MAP-table-values are responses of 1-bit vectors to an integrating low-pass filter its values are not uniformly distributed over the entire range. It can happen that two 1-bit vectors have an almost identical response to the filter. It should be useful to have an algorithm that decreases this redundancy and reduces the MAP-table to a size which corresponds to the precision of the used PCM format.

D.2 An algorithm based on the distances between entries

The smallest possible MAP-table has uniformly distributed values in the entire possible range of its input at distances according to the used PCM accuracy. The original MAP-table is symmetrical for positive and negative values and shows small distances for small values and larger distances at the top of its range (see figure D.1).

Deleting an entry is the same as adding two distances. This will be done when two entries are too close to eachother compared to an arbitrary minimum distance. When the distances to the two other neighbours of these values are determined, one of the two middle values is deleted which leads to a minimal newly created distance. When the MAP-table is inspected with a slowly increasing minimum distance for each following inspection it will end up in a more uniformly distributed form.

Using this algorithm, the resulting MAP-table can be less then optimal in its uniformity because only two neighbours of the two values are inspected when it is decided which entry is going to be deleted. At the moment of a deletion, there is no check for its consequences for the next deletions.

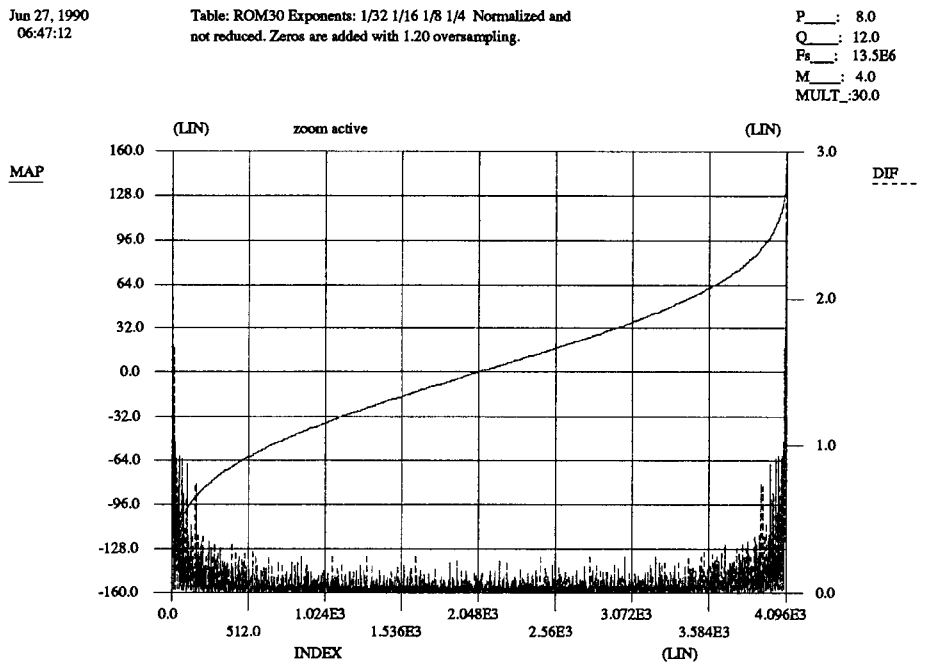

Figure D.1: Non-reduced MAP-table. Printed are the MAP-table values and the differences between two following entries (DIF).

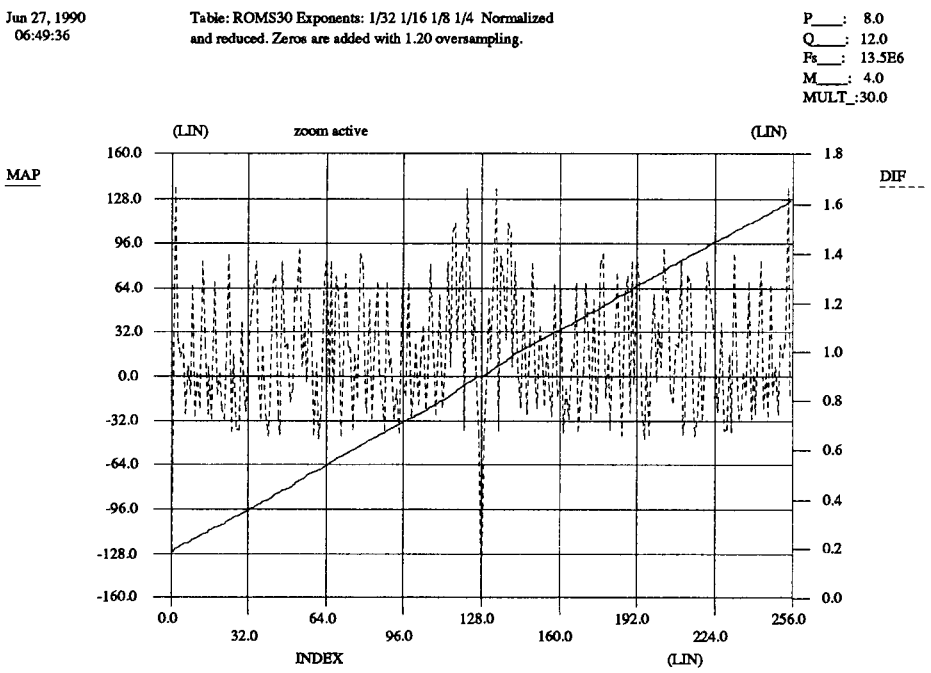

Figure D.2: MAP-table reduced with the distance algorithm. Printed are the MAP-table-values and the differences between two following entries (DIF).

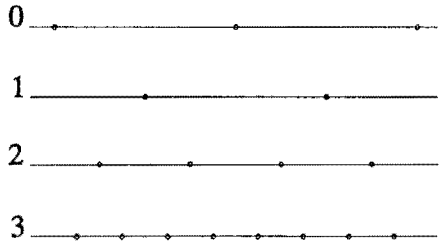

Figure D.3: Grid points with decreasing priority for the shown precision bits. Each row contains the grid-points that are added by introducing the corresponding bit.

D.3 An algorithm based on a grid of desired values

It can be interesting to think about the values which are finally desired in the MAP-table. Taking account of the finite word-length for the internal calculations, it is possible to say something about the values that will be the inputs of the MAP-table. Considering a PCM signal of 8 bits without decimals, it is obvious that the most interesting are integer values and eventually the values which become possible with extra bits for the decimals in the internal calculations. These values will be called "grid-values".

The importance of the grid-values decreases in the order they become available by increasing the word-length. Entry collection is started (arbitrary) in the lower range of the MAP-table for each new inspection, introducing an extra bit, because rounding errors are relatively bigger for small values.

The MAP-table will be inspected with a decreasing grid-size for the values that are "close enough" to these grid-values just until enough entries are collected. An entry is defined to be "close enough" when it is closer than other grid-values in the same or upper layers in figure D.3.

D.4 Comparison of the algorithms

The distance-algorithm is using a deletion-criterium, and the grid-algorithm is using a selection-criterium. At first sight, the "What do we want to end with?" criterium should lead to better results than a "What do we've got to start with?" criterium.

Both the algorithms have been implemented and tested for several MAP-tables. Diagrams were drawn for both the MAP-tables themselves and the step-sizes between the values. Both the algorithms deliver MAP-tables with step-sizes close to the PCM accuracy. The second algorithm shows a more uniformly distributed MAP-table and the steps in the lower range are closer to the PCM resolution, which decreases the relative error.

During simulations with the COMBIC-"PCM–1-bit–PCM"-chain, the performance was not always better with MAP-tables reduced with the grid-algorithm. Whenever it is decided to use a reduced MAP-table, a more detailed study in the effects of both the algorithms is suggested.

For the moment, reducing the MAP-table seems to give quite a variable decrease

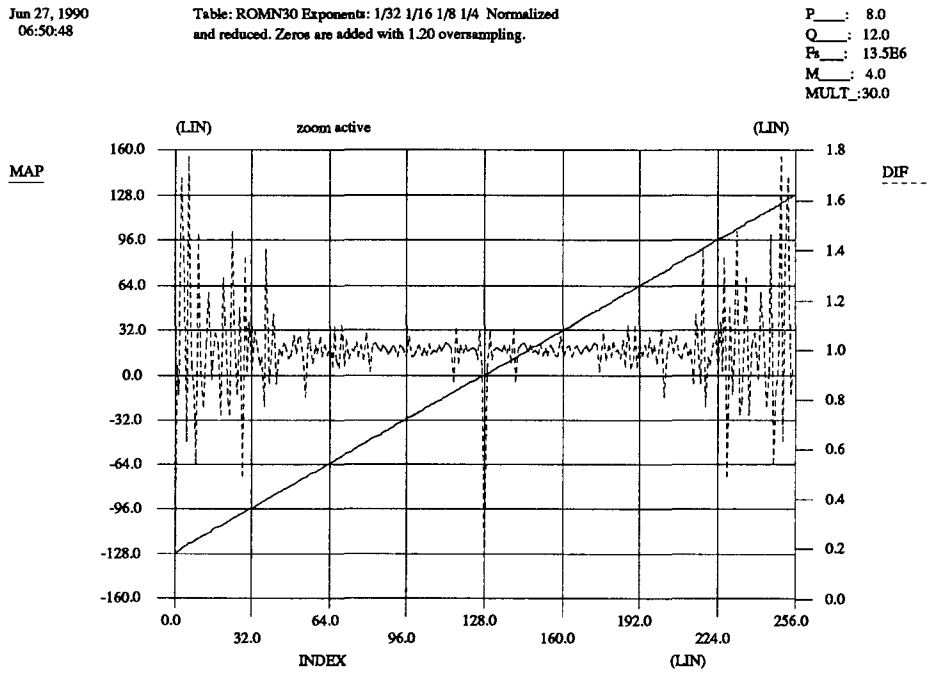

Figure D.4: MAP-table reduced with the grid algorithm. Printed are the MAP-table-values and the differences between two following entries (DIF).

of performance. A comparison of signals after low-pass filtering could lead to a final conclusion, as soon as the use of the non-reduced MAP-table itself gives satisfying results.

Appendix E

Used files

E.1 Introduction

To guarantee consistent simulation-results, a script-file has been made to generate all the necessary files according to the parameters in one unique file. A simulation is started by the execution of "script/simu" while all the parameters are defined in the file "simu/param". It is also possible to make only the programs consistent to their source- and include-files. This is performed by executing "script/prog", resulting in a consistent set of programs under the "bin" directory and a file "simu/list" which contains a list of all the programs. The base directory for compilation of the modules and simulations is always "simu".

E.2 Directories

All the files needed for COMBIC-simulations are grouped in a few directories:

- "bin" is linked to "$(ISP)/bin" (where "ISP" denotes the processor-type on which the programs will be executed) and these directories contain the executable programs needed for the COMBIC-simulations.
- "obj" is linked to "$(ISP)/obj" and these directories contain the object-files to build the programs under "bin".
- "src" contains the source-files of all the modules.
- "inc" contains the specific include-files as used by the source-files in "src".
- "make" contains make-files. The file "make/programs" defines the construction of the programs under "bin", "make/simulation" defines which files depend on which files and programs during a simulation. When only compiling the programs, a list of these programs will be generated according to "make/list" and will be put in the file "simu/list".
- "script" contains script-files which are directly runnable. See section E.3 for a detailed description.

- "simu" is the directory where all the produced data-files are put and it contains the file "simu/param" with all the simulation-parameters in it.

E.3 Script-files

The directory "script" contains a few script-files which can be executed locally on the apollo work-station and on the DN10000 machines.

- "script/simu" will concatenate the files "simu/param", "make/simulation" and "make/prograı to a file called "make/simu". This file will be passed to the program "/bin/make". Through this, a consistent simulation is performed with all its files in the directory "simu". File-names are constructed according to the simulation-parameters and some links are set to the results of this simulation. See section E.6.

- "script/prog" will concatenate the files "make/list" and "make/programs" to a file called "make/prog" which will be passed to "/bin/make". This results in a set of executable files under the directory "bin" which are listed in the file "simu/list".

E.4 Programs

Under the directory "bin", the following programs should be available or can be build by executing "script/prog". All the programs will return a short list of the arguments they take, when they are executed without arguments. All the programs check the environment for the existance and value of the environment-variable "DEBUG". When "DEBUG=true", the programs will be executed in the debug-mode which causes the printing of extra information during execution.

- "bin/mlpf" generates filter-description-files or impulse- and frequency-responses of filters in agap-format (section 5.2).

- "bin/lpf" filters an input signal-file in agap-format according to a filter-description-file.

- "bin/sin" generates a signal-file describing a sine-wave (section 4.6).

- "bin/table" generates a COMBIC-description-file according to a filter-description-file and some other parameters (section 5.4).

- "bin/print" generates a text-file describing the Lookup-tables and an agap-file describing the MAP-table as they are represented in the given COMBIC-description-file.

- "bin/sim" performs a simulation of the given configuration according to the COMBIC-description-file, some filter-description-files and an input signal-file (chapter 6).

- "bin/testing" performs some tests of functions which are critical for the use of Lookup-tables.

E.5 File-types

The output-files are put in the directory "simu" and can be one of the following types:

- Text-files.
- Agap-files, which are readable by the programs "agap" and "ugap" to view signals or to calculate the FFT of these signals.
- Filter-description-files, describing a filter by poles, zeros, a multiplying constant and a low and a high symbol-rate. They can be generated with "mlpf".
- COMBIC-description-files, containing the parameters and the Lookup-tables generated by "table" and used by the COMBIC-converters in the program "sim".

E.6 The simulation-directory

The directory "simu" is the central simulation directory. It contains the links to all the related directories such as "src", "inc", "obj", "bin", and "make". The simulation-script-files use these links to find the files they need. All the data-files produced during simulations are put in the "simu"-directory itself. After a run of "script/simu", "simu" will contain some links to the latest results:

- "rom" links to the COMBIC-description-file, "sys" to the COMBIC-filter-file.
- "sig" to the filter-file used by the $\Sigma\Delta$-modulator.
- "but" to the Butterworth-file, "sin" to the used input-signal-file.
- "dat" is a link to the generated output-file.

Appendix F

Example of a simulation-run

The following is the output of a complete simulation which is generated with the make-file "make/simu", as invoked by the script-file "script/simu".

```
------- "simu" for "franken" in "johnson" on "concorde"

  Job   : //carp/local/user/franken/script/simu
  Dir   : //carp/local/user/franken/simu
  Lim   : 00:05:00
  At    : Wed Feb 13 15:32:46 1991

------

Background .profile
PARAMETER-FILE: param
bin/mlpf 1 r xa,xb,xc,xd        n       13.5e6        10 sys
—DEBUG MODE—

Opening "sys" for writing.
Filter:
      n/xa,xb,xc,xd 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR  :  1.363e+21+j0.000e+00

      POLE  ( 0) -1.489e+06+j0.000e+00 ALFA  ( 0)  2.773e-01+j0.000e+00
      POLE  ( 1) -2.979e+06+j0.000e+00 ALFA  ( 1) -8.318e-01+j0.000e+00
      POLE  ( 2) -4.468e+06+j0.000e+00 ALFA  ( 2)  8.318e-01+j0.000e+00
      POLE  ( 3) -5.957e+06+j0.000e+00 ALFA  ( 3) -2.773e-01+j0.000e+00

      LO-EXP( 0)  5.000e-01+j0.000e+00 HI-EXP( 0)  9.330e-01+j0.000e+00
      LO-EXP( 1)  2.500e-01+j0.000e+00 HI-EXP( 1)  8.706e-01+j0.000e+00
      LO-EXP( 2)  1.250e-01+j0.000e+00 HI-EXP( 2)  8.123e-01+j0.000e+00
      LO-EXP( 3)  6.250e-02+j0.000e+00 HI-EXP( 3)  7.579e-01+j0.000e+00

No warnings.
No errors.


bin/sn sys 1e-2 1e-4
—DEBUG MODE—
Opening "sys" for reading.
Filter:
      n/xa,xb,xc,xd 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR  :  1.363e+21+j0.000e+00

      POLE  ( 0) -1.489e+06+j0.000e+00 ALFA  ( 0)  2.773e-01+j0.000e+00
      POLE  ( 1) -2.979e+06+j0.000e+00 ALFA  ( 1) -8.318e-01+j0.000e+00
      POLE  ( 2) -4.468e+06+j0.000e+00 ALFA  ( 2)  8.318e-01+j0.000e+00
      POLE  ( 3) -5.957e+06+j0.000e+00 ALFA  ( 3) -2.773e-01+j0.000e+00

      LO-EXP( 0)  5.000e-01+j0.000e+00 HI-EXP( 0)  9.330e-01+j0.000e+00
      LO-EXP( 1)  2.500e-01+j0.000e+00 HI-EXP( 1)  8.706e-01+j0.000e+00
      LO-EXP( 2)  1.250e-01+j0.000e+00 HI-EXP( 2)  8.123e-01+j0.000e+00
      LO-EXP( 3)  6.250e-02+j0.000e+00 HI-EXP( 3)  7.579e-01+j0.000e+00

h(0):
       1.110e-16+j0.000e+00

h(T):
       1.733e-02+j0.000e+00

Algorithm for:
      h(0) == 0.

SIGNAL-POWER:
       3.003e-04

NOISE-POWER:
       1.941e-06

(S/N)db:
       2.189e+01


No warnings.
No errors.


bin/table 8,.5,r 10 sys y n y n rom
—DEBUG MODE—

PCM Format:
      Range [-5.00e-01,5.00e-01].
      Rounding values.
      Using 8 bits.
1-bit Format:
      Using 10 bits.

Opening "sys" for reading.
Filter:
      n/xa,xb,xc,xd 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
```

```
      FACTOR :  1.363e+21+j0.000e+00

      POLE ( 0)-1.489e+06+j0.000e+00 ALFA ( 0)  2.773e-01+j0.000e+00
      POLE ( 1)-2.979e+06+j0.000e+00 ALFA ( 1)-8.318e-01+j0.000e+00
      POLE ( 2)-4.468e+06+j0.000e+00 ALFA ( 2)  8.318e-01+j0.000e+00
      POLE ( 3)-5.957e+06+j0.000e+00 ALFA ( 3)-2.773e-01+j0.000e+00

      LO-EXP( 0)  5.000e-01+j0.000e+00 HI-EXP( 0)  9.330e-01+j0.000e+00
      LO-EXP( 1)  2.500e-01+j0.000e+00 HI-EXP( 1)  8.706e-01+j0.000e+00
      LO-EXP( 2)  1.250e-01+j0.000e+00 HI-EXP( 2)  8.123e-01+j0.000e+00
      LO-EXP( 3)  6.250e-02+j0.000e+00 HI-EXP( 3)  7.579e-01+j0.000e+00

h(0):
      1.110e-16+j0.000e+00

h(T):
      1.733e-02+j0.000e+00

Algorithm for:
      h(0) == 0.

Normalizing:
      Yes

Filter:
      n/xa,xb,xc,xd 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR :  7.866e+22+j0.000e+00

      POLE ( 0)-1.489e+06+j0.000e+00 ALFA ( 0)  1.600e+01+j0.000e+00
      POLE ( 1)-2.979e+06+j0.000e+00 ALFA ( 1)-4.800e+01+j0.000e+00
      POLE ( 2)-4.468e+06+j0.000e+00 ALFA ( 2)  4.800e+01+j0.000e+00
      POLE ( 3)-5.957e+06+j0.000e+00 ALFA ( 3)-1.600e+01+j0.000e+00

      LO-EXP( 0)  5.000e-01+j0.000e+00 HI-EXP( 0)  9.330e-01+j0.000e+00
      LO-EXP( 1)  2.500e-01+j0.000e+00 HI-EXP( 1)  8.706e-01+j0.000e+00
      LO-EXP( 2)  1.250e-01+j0.000e+00 HI-EXP( 2)  8.123e-01+j0.000e+00
      LO-EXP( 3)  6.250e-02+j0.000e+00 HI-EXP( 3)  7.579e-01+j0.000e+00

SIGNAL-POWER:
      1.000e+00

NOISE-POWER:
      5.391e-03

(S/N)db:
      2.268e+01

Contributions:

 8.000e+00+j0.000e+00 -1.200e+01+j0.000e+00  6.000e+00+j0.000e+00 -1.000e+00+j0.000e+00  1.000e+00+j0.000e+00   0.100
 8.574e+00+j0.000e+00 -1.378e+01+j0.000e+00  7.387e+00+j0.000e+00 -1.320e+00+j0.000e+00  8.572e-01+j0.000e+00   0.186
 9.190e+00+j0.000e+00 -1.583e+01+j0.000e+00  9.094e+00+j0.000e+00 -1.741e+00+j0.000e+00  7.087e-01+j0.000e+00   0.257
 9.849e+00+j0.000e+00 -1.819e+01+j0.000e+00  1.120e+01+j0.000e+00 -2.297e+00+j0.000e+00  5.596e-01+j0.000e+00   0.313
 1.056e+01+j0.000e+00 -2.089e+01+j0.000e+00  1.378e+01+j0.000e+00 -3.031e+00+j0.000e+00  4.158e-01+j0.000e+00   0.354
 1.131e+01+j0.000e+00 -2.400e+01+j0.000e+00  1.697e+01+j0.000e+00 -4.000e+00+j0.000e+00  2.843e-01+j0.000e+00   0.383
 1.213e+01+j0.000e+00 -2.757e+01+j0.000e+00  2.089e+01+j0.000e+00 -5.278e+00+j0.000e+00  1.722e-01+j0.000e+00   0.400
 1.300e+01+j0.000e+00 -3.167e+01+j0.000e+00  2.572e+01+j0.000e+00 -6.964e+00+j0.000e+00  8.601e-02+j0.000e+00   0.408
 1.393e+01+j0.000e+00 -3.638e+01+j0.000e+00  3.167e+01+j0.000e+00 -9.190e+00+j0.000e+00  3.021e-02+j0.000e+00   0.411
 1.493e+01+j0.000e+00 -4.179e+01+j0.000e+00  3.899e+01+j0.000e+00 -1.213e+01+j0.000e+00  4.483e-03+j0.000e+00   0.412
 1.600e+01+j0.000e+00 -4.800e+01+j0.000e+00  4.800e+01+j0.000e+00 -1.600e+01+j0.000e+00 -7.105e-15+j0.000e+00   0.412

BMJ'FBK:
      n

MAP'FBK:
      y

Generating the AMJ and MAP tables.

Sorting the MAP table in ascending order.

MAP reduction:
      none

Smallest MAP value:
      -7.299420693352e-05

MAP Format:
      Range [-4.12e-01,4.12e-01].
      Using 10 bits.

Generating the BMJ table.

Storing the parameters in the ROM table file.
Opening "rom" for writing.

No warnings.
No errors.


bin/mlpf 1 r but,8,5e6       n       13.5e6       10 but
---DEBUG MODE---
```

```
Opening "but" for writing.
Filter:
      n/but,8,5e6 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR  :  7.029e+51-j3.323e+35

      POLE  ( 0) -4.904e+06-j9.755e+05 ALFA  ( 0)  8.303e-01+j1.243e+00
      POLE  ( 1) -4.157e+06-j2.778e+06 ALFA  ( 1) -9.794e-01+j1.948e-01
      POLE  ( 2) -2.778e+06-j4.157e+06 ALFA  ( 2)  8.069e-02-j4.057e-01
      POLE  ( 3) -9.755e+05-j4.904e+06 ALFA  ( 3)  6.841e-02+j4.571e-02
      POLE  ( 4) -9.755e+05+j4.904e+06 ALFA  ( 4)  6.841e-02-j4.571e-02
      POLE  ( 5) -2.778e+06+j4.157e+06 ALFA  ( 5)  8.069e-02+j4.057e-01
      POLE  ( 6) -4.157e+06+j2.778e+06 ALFA  ( 6) -9.794e-01-j1.948e-01
      POLE  ( 7) -4.904e+06+j9.755e+05 ALFA  ( 7)  8.303e-01-j1.243e+00

      LO-EXP( 0)  9.170e-02-j4.475e-02 HI-EXP( 0)  7.951e-01-j3.612e-02
      LO-EXP( 1)  3.963e-02-j1.389e-01 HI-EXP( 1)  8.172e-01-j1.062e-01
      LO-EXP( 2) -9.775e-02-j2.565e-01 HI-EXP( 2)  8.623e-01-j1.690e-01
      LO-EXP( 3) -4.147e-01-j4.810e-01 HI-EXP( 3)  9.308e-01-j2.162e-01
      LO-EXP( 4) -4.147e-01+j4.810e-01 HI-EXP( 4)  9.308e-01+j2.162e-01
      LO-EXP( 5) -9.775e-02+j2.565e-01 HI-EXP( 5)  8.623e-01+j1.690e-01
      LO-EXP( 6)  3.963e-02+j1.389e-01 HI-EXP( 6)  8.172e-01+j1.062e-01
      LO-EXP( 7)  9.170e-02+j4.475e-02 HI-EXP( 7)  7.951e-01+j3.612e-02

No warnings.
No errors.


bin/mlpf 1 r sig            sig            13.5e6       10 sig
—DEBUG MODE—

Opening "sig" for writing.
Filter:
      sig/sig 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR  :  2.992e-11+j0.000e+00

      ZERO  ( 0) -1.254e+07+j0.000e+00
      ZERO  ( 1) -1.408e+07+j0.000e+00
      ZERO  ( 2) -3.857e+07+j0.000e+00

      POLE  ( 0) -2.237e+06+j0.000e+00 ALFA  ( 0)  7.775e-01+j0.000e+00
      POLE  ( 1) -2.700e+06+j0.000e+00 ALFA  ( 1) -8.806e-01+j0.000e+00
      POLE  ( 2) -4.551e+06+j0.000e+00 ALFA  ( 2)  1.135e-01+j0.000e+00

      LO-EXP( 0)  3.530e-01+j0.000e+00 HI-EXP( 0)  9.011e-01+j0.000e+00
      LO-EXP( 1)  2.846e-01+j0.000e+00 HI-EXP( 1)  8.819e-01+j0.000e+00
      LO-EXP( 2)  1.202e-01+j0.000e+00 HI-EXP( 2)  8.091e-01+j0.000e+00

No warnings.
No errors.


bin/sin 2e-6 .45 1e6 0 13.5e6 10   sin
—DEBUG MODE—

Producing PCM input file.
0.45sin(2PI*1e6+ 0') 1.35e+07 10 [0,2.0e-06]
Opening "sin" for writing.
Writing blocks: INFO TITLE NAMES HEADING NUMBERS

No warnings.
No errors.


bin/sim -1,-1,n 014  rom sig but sin  13.5e6/10/xa,xb,xc,xd'n'ynyn'-1n
—DEBUG MODE—

Reading ROM tables.
Opening "rom" for reading.
PCM Format:
      Range [-5.00e-01,5.00e-01].
      Rounding values.
      Using 8 bits.
1-bit Format:
      Range [-4.12e-01,4.12e-01].
      Using 10 bits.

Filter:
      n/xa,xb,xc,xd 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR  :  7.866e+22+j0.000e+00

      POLE  ( 0) -1.489e+06+j0.000e+00 ALFA  ( 0)  1.600e+01+j0.000e+00
      POLE  ( 1) -2.979e+06+j0.000e+00 ALFA  ( 1) -4.800e+01+j0.000e+00
      POLE  ( 2) -4.468e+06+j0.000e+00 ALFA  ( 2)  4.800e+01+j0.000e+00
      POLE  ( 3) -5.957e+06+j0.000e+00 ALFA  ( 3) -1.600e+01+j0.000e+00

      LO-EXP( 0)  5.000e-01+j0.000e+00 HI-EXP( 0)  9.330e-01+j0.000e+00
```

```
      LO-EXP( 1)  2.500e-01+j0.000e+00 HI-EXP( 1)  8.706e-01+j0.000e+00
      LO-EXP( 2)  1.250e-01+j0.000e+00 HI-EXP( 2)  8.123e-01+j0.000e+00
      LO-EXP( 3)  6.250e-02+j0.000e+00 HI-EXP( 3)  7.579e-01+j0.000e+00

Algorithm for:
      h(0)==0

Normfactor:
       1.000e+00+j0.000e+00

BMJ'FBK:
      n

MAP'FBK:
      y

MAP-reduction:
      none

SN-ratio:
      22.7

Reading LPF for sigma delta modulator.
Opening "sig" for reading.
Filter:
      sig/sig 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR :  2.992e-11+j0.000e+00

      ZERO  ( 0) -1.254e+07+j0.000e+00
      ZERO  ( 1) -1.408e+07+j0.000e+00
      ZERO  ( 2) -3.857e+07+j0.000e+00

      POLE  ( 0) -2.237e+06+j0.000e+00 ALFA  ( 0)  7.775e-01+j0.000e+00
      POLE  ( 1) -2.700e+06+j0.000e+00 ALFA  ( 1) -8.806e-01+j0.000e+00
      POLE  ( 2) -4.551e+06+j0.000e+00 ALFA  ( 2)  1.135e-01+j0.000e+00

      LO-EXP( 0)  3.530e-01+j0.000e+00 HI-EXP( 0)  9.011e-01+j0.000e+00
      LO-EXP( 1)  2.846e-01+j0.000e+00 HI-EXP( 1)  8.819e-01+j0.000e+00
      LO-EXP( 2)  1.202e-01+j0.000e+00 HI-EXP( 2)  8.091e-01+j0.000e+00

Reading general LPF.
Opening "but" for reading.
Filter:
      n/but,8,5e6 1.35e+07 10

      LO-FREQ :  1.350e+07
      HI-FREQ :  1.350e+08
      FACTOR :  7.029e+51-j3.323e+35

      POLE  ( 0) -4.904e+06-j9.755e+05 ALFA  ( 0)  8.303e-01+j1.243e+00
      POLE  ( 1) -4.157e+06-j2.778e+06 ALFA  ( 1) -9.794e-01+j1.948e-01
      POLE  ( 2) -2.778e+06-j4.157e+06 ALFA  ( 2)  8.069e-02-j4.057e-01
      POLE  ( 3) -9.755e+05-j4.904e+06 ALFA  ( 3)  6.841e-02+j4.571e-02
      POLE  ( 4) -9.755e+05+j4.904e+06 ALFA  ( 4)  6.841e-02-j4.571e-02
      POLE  ( 5) -2.778e+06+j4.157e+06 ALFA  ( 5)  8.069e-02+j4.057e-01
      POLE  ( 6) -4.157e+06+j2.778e+06 ALFA  ( 6) -9.794e-01-j1.948e-01
      POLE  ( 7) -4.904e+06+j9.755e+05 ALFA  ( 7)  8.303e-01-j1.243e+00

      LO-EXP( 0)  9.170e-02-j4.475e-02 HI-EXP( 0)  7.951e-01-j3.612e-02
      LO-EXP( 1)  3.963e-02-j1.389e-01 HI-EXP( 1)  8.172e-01-j1.062e-01
      LO-EXP( 2) -9.775e-02-j2.565e-01 HI-EXP( 2)  8.623e-01-j1.690e-01
      LO-EXP( 3) -4.147e-01-j4.810e-01 HI-EXP( 3)  9.308e-01-j2.162e-01
      LO-EXP( 4) -4.147e-01+j4.810e-01 HI-EXP( 4)  9.308e-01+j2.162e-01
      LO-EXP( 5) -9.775e-02+j2.565e-01 HI-EXP( 5)  8.623e-01+j1.690e-01
      LO-EXP( 6)  3.963e-02+j1.389e-01 HI-EXP( 6)  8.172e-01+j1.062e-01
      LO-EXP( 7)  9.170e-02+j4.475e-02 HI-EXP( 7)  7.951e-01+j3.612e-02

Reading input samples.
Opening "sin" for reading.
Reading blocks: INFO TITLE NAMES HEADING NUMBERS
0.45sin(2PI*1e6+ 0') 1.35e+07 10 [0,2.0e-06]



Internal calculation Format:
      No range limiting.
      No rounding or truncating.

Writing agap results.
Opening "13.5e6/10/xa,xb,xc,xd'n'ynyn'-1n" for writing.
Writing blocks: INFO TITLE NAMES HEADING NUMBERS

—Start of simulation—

TIME 1.00e-30
      IN0 1.00e-30 LPF0
      SDM 1001101011 LPF1
      1TP a 2.461e-02+j0.000e+00 r 0.000e+00+j0.000e+00 b 2.461e-02+j0.000e+00 LPF4
TIME 7.41e-08
```

```
      IN0 2.02e-01 LPF0
      SDM 0101101101 LPF1
      1TP a 7.199e-03+j0.000e+00 r 2.099e-01+j0.000e+00 b 2.171e-01+j0.000e+00 LPF4
TIME 1.48e-07
      IN0 3.61e-01 LPF0
      SDM 1011101011 LPF1
      1TP a 1.663e-01+j0.000e+00 r 1.998e-01+j0.000e+00 b 3.661e-01+j0.000e+00 LPF4
TIME 2.22e-07
      IN0 4.43e-01 LPF0
      SDM 1101101110 LPF1
      1TP a 2.123e-01+j0.000e+00 r 2.302e-01+j0.000e+00 b 4.425e-01+j0.000e+00 LPF4
TIME 2.96e-07
      IN0 4.31e-01 LPF0
      SDM 1101101110 LPF1
      1TP a 2.123e-01+j0.000e+00 r 2.064e-01+j0.000e+00 b 4.187e-01+j0.000e+00 LPF4
TIME 3.70e-07
      IN0 3.27e-01 LPF0
      SDM 1011100110 LPF1
      1TP a 1.482e-01+j0.000e+00 r 1.717e-01+j0.000e+00 b 3.199e-01+j0.000e+00 LPF4
TIME 4.44e-07
      IN0 1.54e-01 LPF0
      SDM 1100110100 LPF1
      1TP a 1.168e-01+j0.000e+00 r 1.007e-02+j0.000e+00 b 1.269e-01+j0.000e+00 LPF4
TIME 5.19e-07
      IN0-5.22e-02 LPF0
      SDM 1100101001 LPF1
      1TP a 7.809e-02+j0.000e+00 r-1.356e-01+j0.000e+00 b-5.750e-02+j0.000e+00 LPF4
TIME 5.93e-07
      IN0-2.47e-01 LPF0
      SDM 0100010100 LPF1
      1TP a-1.663e-01+j0.000e+00 r-9.964e-02+j0.000e+00 b-2.660e-01+j0.000e+00 LPF4
TIME 6.67e-07
      IN0-3.90e-01 LPF0
      SDM 0100100010 LPF1
      1TP a-1.512e-01+j0.000e+00 r-2.437e-01+j0.000e+00 b-3.949e-01+j0.000e+00 LPF4
TIME 7.41e-07
      IN0-4.49e-01 LPF0
      SDM 0100010010 LPF1
      1TP a-1.775e-01+j0.000e+00 r-2.608e-01+j0.000e+00 b-4.383e-01+j0.000e+00 LPF4
TIME 8.15e-07
      IN0-4.13e-01 LPF0
      SDM 0010010010 LPF1
      1TP a-2.072e-01+j0.000e+00 r-2.053e-01+j0.000e+00 b-4.125e-01+j0.000e+00 LPF4
TIME 8.89e-07
      IN0-2.89e-01 LPF0
      SDM 1000110001 LPF1
      1TP a-7.092e-02+j0.000e+00 r-1.858e-01+j0.000e+00 b-2.568e-01+j0.000e+00 LPF4
TIME 9.63e-07
      IN0-1.04e-01 LPF0
      SDM 1010011001 LPF1
      1TP a 2.209e-02+j0.000e+00 r-1.276e-01+j0.000e+00 b-1.055e-01+j0.000e+00 LPF4
TIME 1.04e-06
      IN0 1.04e-01 LPF0
      SDM 1100110101 LPF1
      1TP a 1.177e-01+j0.000e+00 r 2.001e-02+j0.000e+00 b 1.377e-01+j0.000e+00 LPF4
TIME 1.11e-06
      IN0 2.89e-01 LPF0
      SDM 1101011101 LPF1
      1TP a 1.809e-01+j0.000e+00 r 9.079e-02+j0.000e+00 b 2.717e-01+j0.000e+00 LPF4
TIME 1.19e-06
      IN0 4.13e-01 LPF0
      SDM 1011101101 LPF1
      1TP a 1.775e-01+j0.000e+00 r 2.689e-01+j0.000e+00 b 4.464e-01+j0.000e+00 LPF4
TIME 1.26e-06
      IN0 4.49e-01 LPF0
      SDM 1101101110 LPF1
      1TP a 2.123e-01+j0.000e+00 r 1.964e-01+j0.000e+00 b 4.087e-01+j0.000e+00 LPF4
TIME 1.33e-06
      IN0 3.90e-01 LPF0
      SDM 1101101101 LPF1
      1TP a 2.072e-01+j0.000e+00 r 2.078e-01+j0.000e+00 b 4.150e-01+j0.000e+00 LPF4
TIME 1.41e-06
      IN0 2.47e-01 LPF0
      SDM 0110110011 LPF1
      1TP a 4.829e-02+j0.000e+00 r 1.552e-01+j0.000e+00 b 2.034e-01+j0.000e+00 LPF4
TIME 1.48e-06
      IN0 5.22e-02 LPF0
      SDM 0011010010 LPF1
      1TP a-9.529e-02+j0.000e+00 r 1.477e-01+j0.000e+00 b 5.244e-02+j0.000e+00 LPF4
TIME 1.56e-06
      IN0-1.54e-01 LPF0
      SDM 1010001100 LPF1
      1TP a-1.846e-02+j0.000e+00 r-1.576e-01+j0.000e+00 b-1.760e-01+j0.000e+00 LPF4
TIME 1.63e-06
      IN0-3.27e-01 LPF0
      SDM 0100100010 LPF1
      1TP a-1.512e-01+j0.000e+00 r-1.797e-01+j0.000e+00 b-3.309e-01+j0.000e+00 LPF4
TIME 1.70e-06
      IN0-4.31e-01 LPF0
      SDM 1000010010 LPF1
      1TP a-1.489e-01+j0.000e+00 r-2.866e-01+j0.000e+00 b-4.356e-01+j0.000e+00 LPF4
TIME 1.78e-06
      IN0-4.43e-01 LPF0
      SDM 0010100001 LPF1
      1TP a-1.860e-01+j0.000e+00 r-2.347e-01+j0.000e+00 b-4.207e-01+j0.000e+00 LPF4
```

```
TIME 1.85e-06
      IN0-3.61e-01 LPF0
      SDM 0100011000 LPF1
      1TP a-1.491e-01+j0.000e+00 r-2.155e-01+j0.000e+00 b-3.646e-01+j0.000e+00 LPF4
TIME 1.93e-06
      IN0-2.02e-01 LPF0
      SDM 1010100110 LPF1
      1TP a 3.631e-02+j0.000e+00 r-2.162e-01+j0.000e+00 b-1.799e-01+j0.000e+00 LPF4
TIME 2.00e-06
      IN0-1.02e-15 LPF0
      SDM 0110101011 LPF1
      1TP a 2.587e-02+j0.000e+00 r-7.667e-03+j0.000e+00 b 1.820e-02+j0.000e+00 LPF4

—End of simulation—

Internal calculation Format:
      No range limiting.
      No rounding or truncating.
      Maximum = 1.11e+01 Minimum = 7.20e-03 .
PCM signal Format:
      Range [-5.00e-01,5.00e-01].
      Rounding values.
      Using 8 bits.
      Maximum = 4.49e-01 Minimum = 1.17e-02 .
1-bit signal Format:
      Range [-4.12e-01,4.12e-01].
      Using 10 bits.

No warnings.
No errors.


crl dat 13.5e6/10/xa,xb,xc,xd'n'ynyn'-1n -r

———— "simu" terminated in "johnson" on "concorde"

 Stat  : 0
 Time  : 00:01:40  [ 75% user time ]
 Pflt  : 72
 At    : Wed Feb 13 15:35:47 1991

————
```

Index