Eindhoven University of Technology

MASTER

An analog correlator for a high speed DS-CDMA modem

van der Heijden, J.

Award date:
1998

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

7581

Eindhoven University of Technology
Department of Electrical Engineering
Telecomunication Technology and
Electromagnetics (TTE)

Optical Communication Section

An Analog Correlator for a high speed DS-CDMA modem

J. van der Heijden

Report of a graduate project carried out in the period
August 1997 - June 1998

Mentors
ir H.P.A. v.d. Boom and ir F.J.J Kennis

Graduate professor and supervisor
prof. ir G.D. Khoe

The Department of Electrical Engineering of the
Eindhoven University of Technology is not
responsible for the contents of graduate reports

Summary

For upstream communication in CATV networks (communication from subscribers to head-end) CDMA seems to be the best multiple-access technique in comparison with TDMA and FDMA. The greatest advantage of CDMA is its robustness against narrowband interference and impulse noise. This makes the CDMA technique suitable for CATV networks where especially the lower frequency band is jammed by narrow band interference. Disadvantage of CDMA is its complex hardware.

An all digital transmitter and receiver are already realized for the upstream communication. This digital transmitter-receiver combination operates properly. An advantage of an all digital modem is its easy implementation. Almost all hardware is fitted into programmable devices. However, the received signal has to be converted from analog to digital domain. This requires a very high speed A/D-converter. At this moment one of the fastest A/D-converter available is used. If we want to speed up the bitrate of the data, we have to look for another approach.

It was proposed to use an analog correlator, the key part of the receiver, and make the analog to digital conversion in a later stage. After the correlator the received signal is despreaded and the A/D-converter can operate at a lower clock frequency.

An analog correlator circuit was developed in discrete hardware and measurements were carried out. The measurements has shown that the analog correlator can speed up the data rate by at least a factor of ten compared to the digital modem. Measurements of correlation functions with the analog correlator approximate the theoretical correlation functions very well. Also the correlator has been tested in a multi-user system where an arbitrary waveform generator has generated a 25 Mchips/s CDMA signal which corresponds to a bitrate of 400 kbit/s for each user. The analog correlator performs well if the number of users doesn't exceed 80. This was the maximum number of users that could be achieved during the period of research but a slight improvement in the synchronization can lead to better results.

List of symbols and abbreviations

ADC	Analog to digital converter
AGC	Automatic gain control
ASIC	Application specific integrated circuit
AWG	Arbitrary waveform generator
AWGN	Additive white gaussian noise
BER	Bit error rate
BPF	Band pass filter
CATV	Community antenna television
CDMA	Code division multiple access
CMF	Chip matched filter
DAC	Digital to analog converter
DECT	Digital enhanced cordless telephone
DS-CDMA	Direct sequence CDMA
DSP	Digital signal processor
EMC	Electromagnetic compatibility
FIR	Finite impulse response
I	In-phase component of quadrature signal
IC	Integrated circuit
IF	Intermediate frequency
IIR	Infinite impulse response
ISI	Inter symbol interference
I&D	Integrate and dump
LPF	Low pass filter
MAI	Multiple-access interference
MF	Medium filter
NRZ	Non return to zero
PRBS	Pseudo random binary sequence
PCB	Printed circuit board
PLD	Programmable logic device
PN	Pseudo noise
PP Gold code	Preferentially phased Gold code
PSF	Pulse shaping filter
Q	Quadrature component of quadrature signal
QPSK	Quadrature phase shift keying
RF	Radio frequency
RCF	Raised cosine filter
S-CDMA	Synchronous CDMA
SNR	Signal to noise ratio

List of figures

Contents

1. Introduction

The demand for telecommunication is growing very fast today. More and more telecom companies are coming and competing with each other. More and more energy companies and railway companies are interested in telecommunication business because they already have a lot of kilometers of cable all over the country buried in the ground which is the most cost like aspect. Also CATV operators are interested in telecommunication services apart from their radio and television signal distribution. They already have a lot of kilometers of cable to a lot of households so they can compete with the telephone companies in the near future. They can offer the so called *"last mile"* communication. A bandwidth of 25 to 60 MHz (depends on network) of the CATV network is still unused for radio and television broadcasting. Most CATV networks are not suitable for bi-directional communication because of the used filters and amplifiers. The network has to be updated for bi-directional communication by replacing the amplifiers for bi-directional ones [1]. A filter separates the upstream and downstream communication and two separated amplifiers amplify the separated signals. Some CATV network operators have adapted their network already and even offer datacommunication to their subscribers. Until today all these systems are based on a FDMA or TDMA technique (Cable-DECT etc.). These techniques are not robust against the interference and noise penetrating the network. When a subscriber wants access to the network, a channel with minimum noise at that moment is assigned to that user. However, the noise and jamming behavior of the CATV network is very hard to predict and time dependent [1]. It is not hard to understand that these systems are operating with high transmitter power and very poor spectral efficiency. It was found out that spread spectrum techniques can offer a solution to this problem [1]. As the name indicates, the spectrum of a datasignal will be spreaded many times compared to its datarate so the bandwidth becomes much larger than the datarate. This is not a really waste of bandwidth because this bandwidth is shared by all users. With FDMA each user uses a part of the whole available bandwidth at any time, with TDMA each user uses the whole available bandwidth in a part of the total time. With CDMA (which is a spread spectrum technique) each user uses the whole available bandwidth at the same time but here the bandwidth is much larger than the datarate. A CDMA modem is realized already and operates well. This modem is an all digital modem which is easy to implement but not suitable for higher bitrates due to the A/D-conversion process. A solution is found in the implementation of an analog correlator.

In this report a study of the possibility and implementation of an analog correlator is discussed. First an introduction of CDMA technique is given and the working of the digital modem will be discussed as well as its limitations. After that, theoretical aspects of the correlator will be given as well as simulation results. Then the hardware implementation will be discussed and finally some results of the measurements are given to indicate the correlator performance. The correlator is just a part of the modem. The modem we have at this moment consists of eight correlators, so the correlator is not tested in the modem yet and BER tests (as function of SNR) has to be done in the near future. It can be concluded that modems with analog correlators are more complex. As always a trade off has to be made between hardware complexity and data speed.

2. DS-CDMA and CDMA modem

2.1 Explanation of DS-CDMA technique

DS-CDMA stands for *Direct Sequence Code Division Multiple Access*. Where in TDMA systems data is kept apart by assigning a time slot to each user and in FDMA systems by assigning a carrier frequency, in CDMA each user gets its own code. Direct sequence means that each databit is multiplied by a code-sequence (a series of -1's and 1's, called *chips*). In our case one databit is multiplied by a sequence of 127 chips and hence the bandwidth is spreaded by a factor of 127. Each user has its own unique code-sequence which matches the receiver corresponding to that user. The code-sequences used are *PP-Gold codes* and are quasi-orthogonal. This means that the cross correlation between two arbitrary codes is not exactly zero but -1 and that is small compared to the peak value of the auto correlation. This property makes it possible to discriminate between different users. The PP-Gold code is a type of a *pseudo noise (PN)* signal. In this report PP-Gold sequence is meant when PN-sequence is mentioned.

Figure 2.1 illustrates the working of a DS-CDMA system in its simplest form [2,3]. The little diagrams above the signal path are the frequency spectrums of the corresponding points in the circuit. First, the information data is multiplied with a PN-sequence. The spectrum of the information data will be spreaded. This signal will be transmitted into the network. After passing the channel (network), the signal is corrupted with interference. After multiplication with the same sequence the interference energy remains the same but is spreaded over a larger bandwidth and the information data is despreaded. After the lowpass filter the interference energy will decrease and the BER will decrease. The combination of the multiplier and filter is known as the correlator and forms the key part of the receiver.

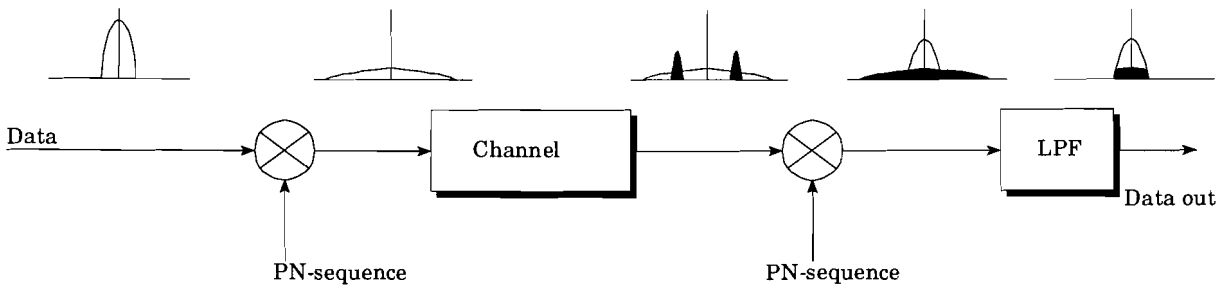


Figure 2.1: *Block diagram of DS-CDMA system.*

The larger the PN-code length the larger its robustness against jamming signals. A very often used value to measure CDMA performance is the *processing gain:* $G_p = R_c / R_b$. Where R_c is the chiprate and R_b the bitrate. The processing gain is a measure of

interference rejection. The greater the processing gain the greater the interference rejection.

Not only narrowband interference is of major importance but also *multiple access interference (MAI)* can degrade the system performance. This is interference caused by other users and can not completely be eliminated because the cross correlation is not zero. MAI can be reduced by synchronization of all users because the cross correlation has is lowest value when the codes are synchronized. This means that all transmitters has to start with their sequence at the same time, or better, the transmitted sequences must be synchronized at the point where the signal enters the receiver. All transmitters are located at different physical locations in the network so at the receivers, which are located at the same physical location, signals from the transmitters have different delay times due to different signal path lengths. In practice, the transmitters don't start at the same time with their sequences because this depends on the path length. A downstream channel takes care of this synchronization. This kind of DS-CDMA is called Synchronous DS-CDMA.

Figure 2.2 shows the working of DS-CDMA in time domain. In this example there are two users active and the code sequence length is 7. The input bits are multiplied by a Gold code and result in a higher information stream containing 1's and -1's. When two users are connected to the channel, the signals of each individual user will add up in the channel resulting in three discrete values: -2, 0, 2. At the receiver, the received signal is multiplied with the same code used in the corresponding transmitter. The result of this multiplication is accumulated for each sequence and the result of this accumulation determines the received bit. A positive result indicates a "1" and a negative result indicates a "0". Considering figure 2.2, the transmitted bits match the received bits. The output of the correlators in this

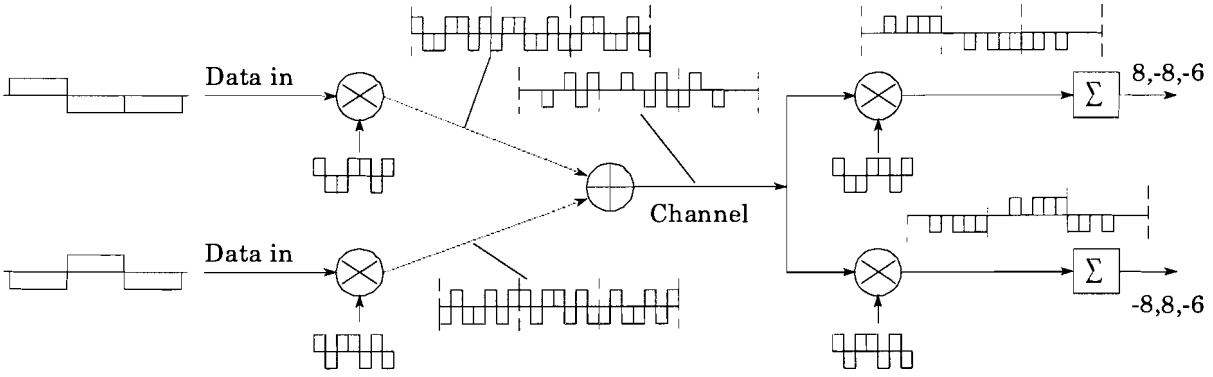


Figure 2.2: *Working of DS-CDMA in time domain.*

example can have four discerete values: 8, 6, -6 and -8. If there was one user active, the output can have two discrete values: 7 and -7. Due to the MAI of the second user, the output of the correlator becomes: 7 ± 1 or -7 ± 1.

2.2 The transmitter

Figure 2.3 illustrates the realization of the digital transmitter as was developed by Janssen [4]. After a differential encoder, the databits are splitted up in two branches by the serial to parallel converter. Then the databits are multiplied with the PN-sequence

(Exclusive-NOR in digital domain). The I and Q branch will be multiplied by two different PN-codes; each user uses two codes. Then the spreaded signal is upsampled by a factor of 4 and goes to the pulse shaping filter. This filter has a FIR structure and has a square-root raised cosine transfer function to obtain Nyquist pulse shaping. This filter limits the bandwidth of the transmitted signal with minimum ISI. The 12-bit output of the filter is connected to a DAC which converts the digital signal to an analog signal. After the DAC the signal is filtered by an analog filter to remove mirror frequency components due to the D/A-conversion (not in the figure). Last stage of the transmitter is the QPSK-modulator which modulates the I and Q signals on a carrier.

Speeding up the datarate is no problem for the transmitter. Fast digital circuits are available on the market today and DAC's are also very fast.

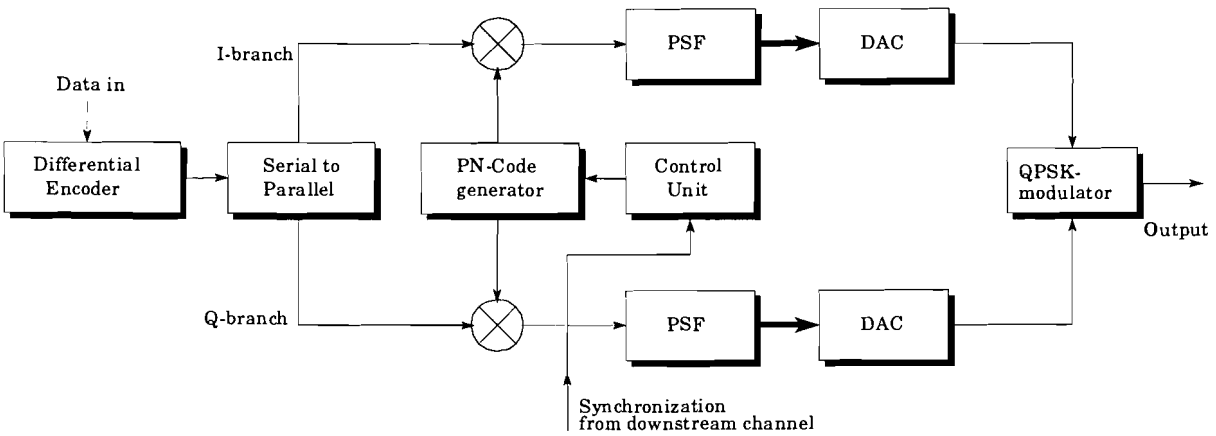


Figure 2.3: *Transmitter.*

2.3 The receiver

The receiver is much more complex in comparison with the transmitter because of its synchronization circuits. The receivers are all at the same physical location: the head-end. All PN-sequences of the receivers are clocked by the same local clock and are all synchronized with respect to each other. When an arbitrary subscriber wants access to the network, the PN-sequence of the transmitter is in general not synchronized with the PN-sequence of the receiver. During the access procedure the receiver measures the correlation between the replica code sequence and the received signal. The receiver transmits synchronization information to the transmitter by the downstream communication channel and the transmitter shifts its PN-sequence phase until the receiver finds a maximum in the correlation. When a maximum is found, the transmitter and receiver are in sync with an uncertainty of a ½ chiptime. Now the real data-transmission can start. During datatransmission the transmitter is fine tuned by the early/late-correlator in the receiver and the transmitter can be synchronized within 1/8 chiptime. Synchronization errors of 1/8 chiptime lead to a SNR degradation of approximately 0.5 dB which is acceptable [5].

All transmitters modulate the spreaded data on a carrier of the same frequency. However, it's very hard to synchronize all carrier phases of the transmitters so all transmitters have a little frequency- and phase deviation so a number of carriers with slight different frequencies and phases enter the receiver. A PLL can offer a solution for

carrier recovering but therefore despreading has to take place prior to carrier recovery [5]. Furthermore every single receiver must have its own demodulation stage and is rather expensive. To reduce hardware complexity and cost price, the QPSK-demodulator, A/D-converters and chip matched filters are shared for all receivers. Demodulation is done in one single device shared by all receivers at one single frequency. Consequence is that some restmodulation is still present after demodulation due to small carrier frequency deviations. This restmodulation is removed after the despreading process. This restmodulation is in the order of magnitude of a few hundred hertz and can be removed by *complex phase rotators* housed in a DSP. Another DSP takes care of the synchronization as final stage of the *early/late correlator.* For a more detailed description read [4,5].

Figure 2.4 depicts the receiver as was developed by Janssen [4]. The upper half of the picture contains the datarecovery part and the lower half of the picture contains the early/late correlator for synchronization purposes.

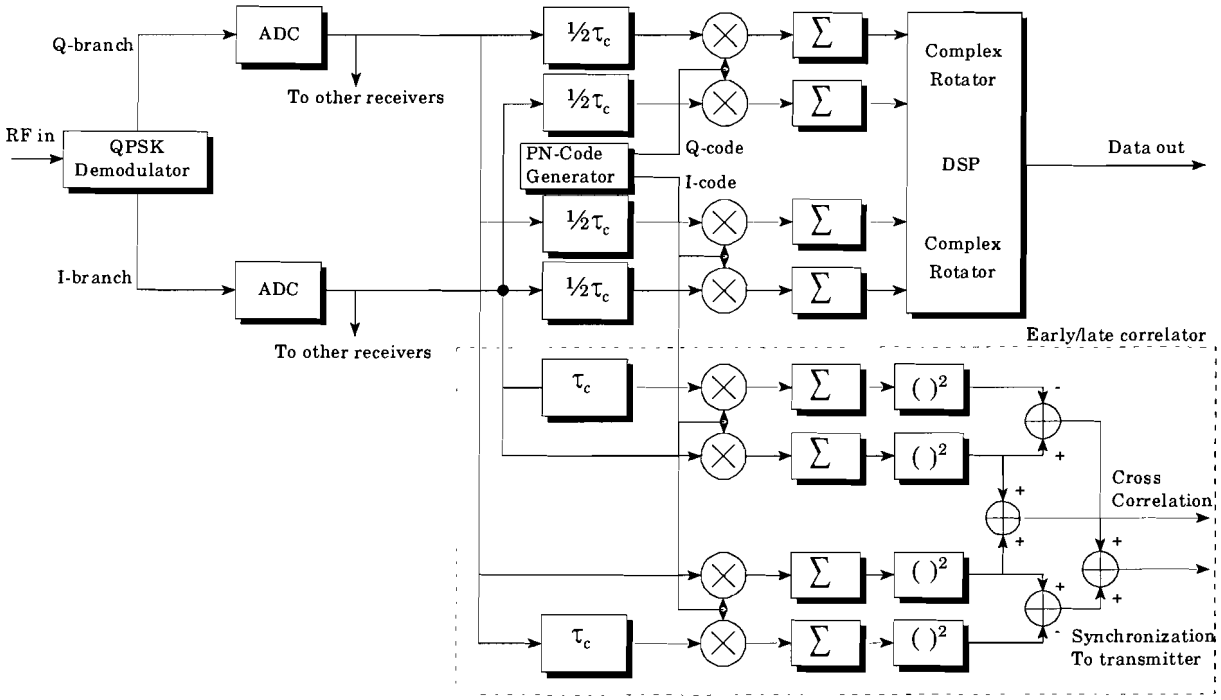


Figure 2.4: *Receiver.*

First the incoming signal is demodulated and splitted into an I and Q-branch. An A/D-converter converts these signals to 12-bit digital signals. A chip matched filter (CMF) reduces the noise-bandwidth (not implemented yet). Then the signal is delayed by a ½ chiptime because the early/late correlator needs a early signal which is a ½ chiptime earlier than the datarecovery branch. Then the signal is correlated with the PN-replicacode. Here an adder is used instead of a filter. The adder (accumulator) acts as an integrate and dump matched filter. The adder will reset to zero when a sequence is elapsed and starts over again with the next sequence. The output of the correlators are connected to a DSP. A software program takes care of the complex phase rotator and datarecovery. The outputs of the early/late correlators are connected to a second DSP. This one calculates a synchronization signal to update the transmitter.

The A/D-converters are the bottleneck if we want to speed up the datarate. The datarate of this modem is *64 kbit/s* so the bitrates of the I- and Q-branch are *32 kbit/s*. With a

sequence length of 127, the chiprate amounts *±4 Mchips/s*. The sample frequency amounts four times the chip rate and becomes 16 MHz. If we want to speed up the datarate by a factor of 4 or higher, no A/D-converter with a resolution of 12 bits is able to convert such high speed signals. So the modem has to be changed to combat this problem.

2.4 Solutions to speed up the datarate

From paragraph 2.3 it became clear that the A/D-converter is the bottleneck if we want to speed up the datarate. There are some solutions to solve this problem.

Assuming the digital hardware is fast enough a solution can be a parallel A/D-conversion process as depicted in figure 2.5. More A/D-converters working in a parallel configuration can process higher signal frequencies. With phase shifted clock pulses, all A/D-converters are processing a different internal state and alternate each other. This solution speeds up the datarate by a factor of two or three or even higher (depends on how much A/D-converters are used). In this configuration all A/D-converters must have the same conversion time so the A/D-converters must be *flash converter* types. This solution has not been considered yet.

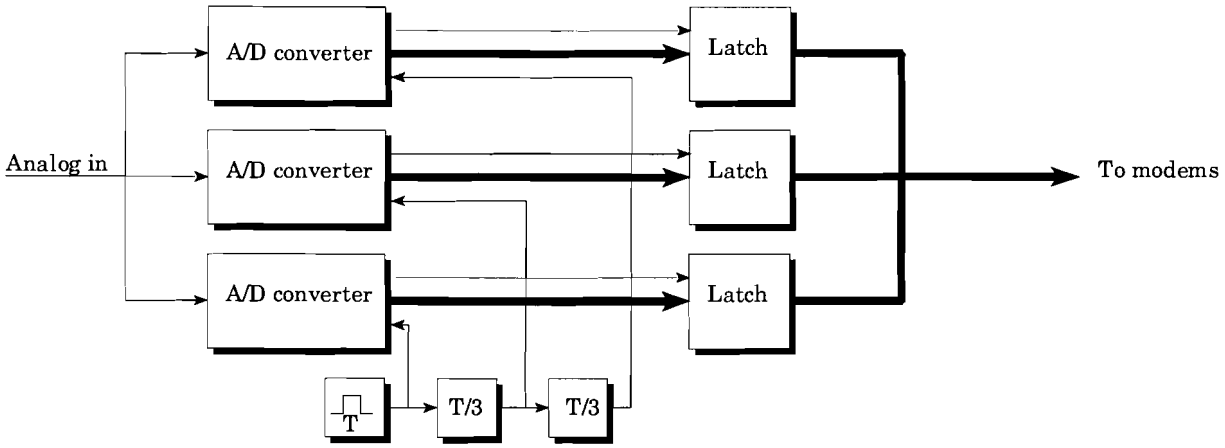


***Figure 2.5**: Parallel process of A/D-conversion.*

A complete different method of approach is to implement an analog correlator instead of a digital one and place the A/D-converter after the correlator where the signal is despreaded. This configuration is depicted in figure 2.6. This reduces the speed of the A/D-converter dramatically. Disadvantage of this approach is the increase of hardware complexity. The A/D-converters are not shared anymore by all receivers, but every receiver needs its own A/D-converters. Furthermore the analog correlator requires a lot of discrete components and one receiver contains eight correlators.

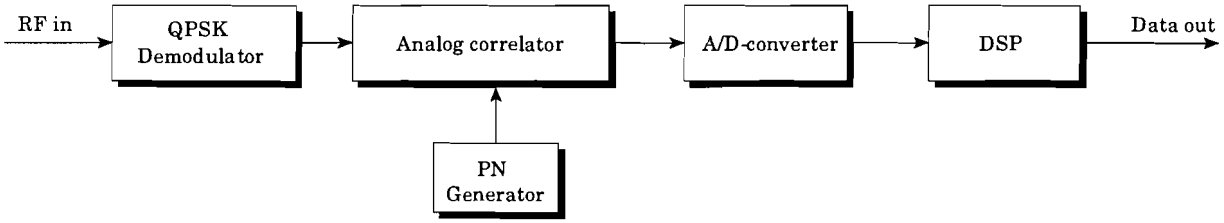


Figure 2.6: *Modem setup with an analog correlator.*

From figure 2.1 it becomes clear that the correlator consists of a multiplier (mixer) followed by an accumulator or a filter. A digital correlator is easy to implement. The multiplier doesn't has to be a multiplier for two analog signals since the incoming signal data is multiplied by -1 or 1 depending on the actual PN-chip. The data signal remains the same when the actual replicacode-chip is 1 and changes to its negative counterpart when the actual replicacode-chip is -1. In two's complement representation this means inverting of the databits and increasing by 1. The accumulator is an adder where the output is connected to one of its inputs. The actual value is added to the value of the last accumulation. When all 127 accumulations are done, the accumulator will be resetted immediately after reading its output by setting all bits to zero [4]. All this is easy to implement in digital hardware.

In analog domain when the correlator consists of discrete analog components, this correlator function is more difficult to implement. A discrete analog mixer has to be applied which is not the most difficult part. The most difficult part is the accumulator. In analog domain an integrate and dump (I&D) circuit has the same function as the accumulator. However, a digital accumulator can reset its output within one chiptime. One short pulse is needed to reset the accumulator. An analog I&D circuit uses a capacitor as memory element where the charge of the capacitor is the value of the accumulator output. The problem is that resetting of the I&D circuit takes time caused by discharging the capacitor. At high chiprates where resetting of the accumulator needs more time than one chiptime, one or more chips are discarded by the accumulator.

Another disadvantage of the analog correlator is that the pulse shapes have a great influence on the correlation since the correlation process is a continuous process. In a digital system a sample of the signal is taken on an optimized instant and the signal is considered to be rectangular for one chip time during further processing.

3. Theory of the correlator

3.1 Correlation functions

The function of the correlator is to return a value which depends on how much two signals look like each other. When these two signals are the same, this is called the auto correlation and when two different signals are applied to a correlation process we call it cross correlation [6]. The auto correlation function is given by:

$$R_{xx}(t_1, t_2) = \mathrm{E}\left[X(t_1)X(t_2)\right] \tag{3.1}$$

where E means the expectation. Mostly we're interested in the auto correlation as function of a time shifted replica of itself. We can replace t_2 by $t_1 + \tau$ and we obtain:

$$R_{xx}(t, t+\tau) = \mathrm{E}\left[X(t)X(t+\tau)\right] \tag{3.2}$$

If the signal X is a wide-sense stationary process, the auto correlation only depends on the time shift and doesn't depend on the time. The auto correlation function is thus only a function of τ. The statistical expectation of a signal can be seen as the time average (ergodic process). The expectation then becomes:

$$\mathrm{E}\left[X(t)\right] = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} X(t)\mathrm{d}t \tag{3.3}$$

When $X(t)$ is a periodical signal the function reduces to:

$$\mathrm{E}\left[X(t)\right] = \frac{1}{T} \int_{0}^{T} X(t)\mathrm{d}t \tag{3.4}$$

and the auto correlation function becomes:

$$R_{xx}(\tau) = \frac{1}{T} \int_{0}^{T} X(t)X(t+\tau)\mathrm{d}t \tag{3.5}$$

From this expression the implementation of the correlator becomes clear: a multiplier followed by an integrator. The constant $1/T$ is of non serious importance because it is just a scaling of the signal. The auto correlation has always a maximum at $\tau = 0$ and is always positive at $\tau = 0$.

The correlator in the CDMA modem doesn't calculate the auto correlation but the cross correlation because in general the PN-replica signal is different from the received signal.

Only if one transmitter is active, the output equals the auto correlation. The cross correlation is given by:

$$R_{XY}(\tau) = \frac{1}{T}\int_0^T X(t)Y(t+\tau)\mathrm{d}t \tag{3.6}$$

Assume for convenience there's only one user active. The correlator receives only -1's and 1's and correlates this received signal with its replicacode. As a matter of fact the correlator calculates the difference between the agreements and disagreements. For example when $\tau = 0$ the received code exactly matches the replica code and there are only agreements. The output of the multiplier is 1 in case of an agreement (-1 * -1 = 1 and 1 * 1 = 1) and -1 in case of a disagreement. After a summation the number of disagreements are subtracted from the number of agreements. When $\tau = 0$ the output of the multiplier has a constant value of 1 and the output of the integrator equals the number of chips and equals 127 in this case (when the constant $1/T$ is taken into account, the output is normalized to 1). The Gold-codes used here have the property of low out-of-phase correlation. This means that the values of the auto correlation functions are small at $\tau \neq 0$ compared with the peak value at $\tau = 0$. The correlation of a Gold-code sequence is smaller than 1/8 of its maximum for $\tau = n.\tau_c$ for $n \neq 0$ where n is an integer. The correlation never equals zero for integer value of n because the sequence length is odd and hence the number of agreements never equals the number of disagreements. The auto correlation function of the Gold-code 0 sequence is depicted in figure 3.1.

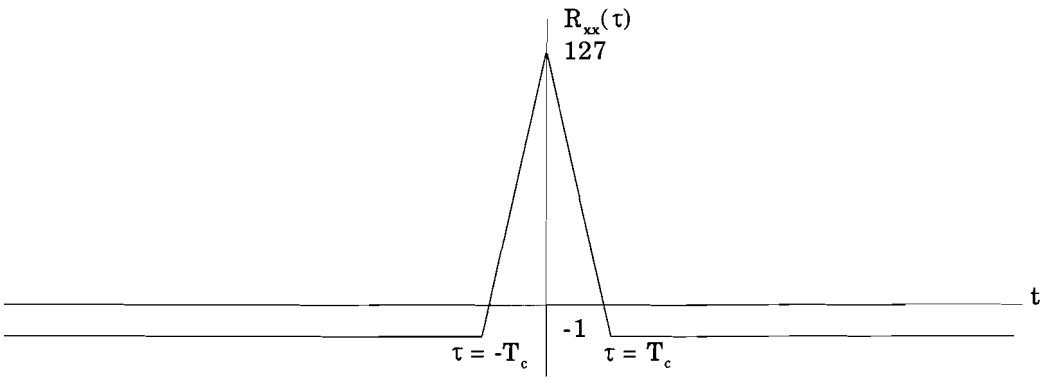


Figure 3.1: *Auto correlation function of Gold-code 0.*

The cross correlation of an arbitrary Gold sequence with another Gold-sequence equals -1 if the sequences are synchronized. This property makes it important to synchronize all code sequences of the transmitters because only then the interference becomes as small as possible.

3.2 Reduction of an interfering signal

As mentioned before, an important property of the spread spectrum system is its robustness against interference. In this paragraph some theoretical attention is paid to the reduction of an interfering signal. This can be a 27 MHz signal coming from an amateur radio transmitter or a 30 MHz signal coming from an in-house cordless telephone for instance.

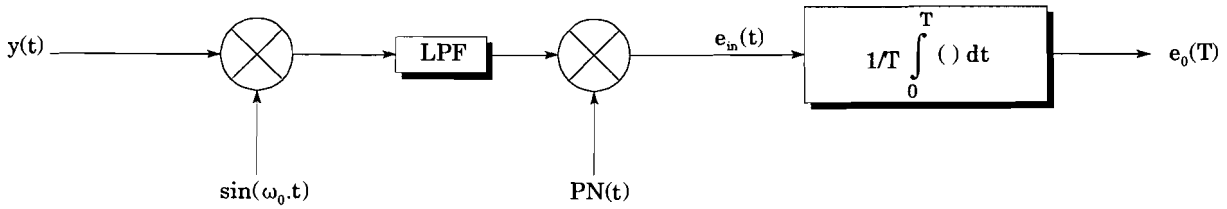


Figure 3.2: *Coherent CDMA receiver.*

Consider the simplified model of a coherent receiver shown in figure 3.2 for the case of a non coherent jammer. Let the input signal plus tone interferer (near the carrier frequency) be modeled as

$$y(t) = 2\sqrt{P}\cdot a(t)\cdot PN(t)\cdot \sin\omega_0 t + 2\sqrt{P_I}\cdot \sin\left[(\omega_0 + \Delta\omega)\cdot t + \theta_0\right] \tag{3.7}$$

which includes the desired signal with power P and a tone interferer with power P_I. The data sequence $a(t)$, has symbol duration T seconds. θ_0 is an uniform random variable and $\Delta\omega$ is small compared to ω_0. We shall assume that the replica code generator and the carrier frequency of the local oscillator are exactly synchronized so the demodulated signal is baseband data. For convenience we shall assume that the data symbols are NRZ so the optimum data demodulator is an integrate-and-dump matched filter [2,7]. When the received signal is multiplied by the local oscillator signal and the replica code, the signal at the input of the integrate-and-dump filter is given by

$$e_{in}(t) = \sqrt{P}\cdot a(t) + \sqrt{P_I}\cdot PN(t)\cdot \cos\left[\Delta\omega\cdot t + \theta_0\right] \tag{3.8}$$

with the assumption that the $2\omega_0$ term is effectively removed by the filter. When $\Delta\omega/2\pi << 1/T_c$ the cosine term varies not much in one sequence time and this simplifies the calculation. The output signal voltage of the data yields

$$\overline{e_0}(T) = \sqrt{P}\cdot a(t) \tag{3.9}$$

because the average of a NRZ bit equals its own voltage level again. The mean square interference power is given by

$$\overline{(e_I - \overline{e_I})^2} = P_I T_c\left(\frac{1}{T}\right) \tag{3.10}$$

assuming the PN(t) is a very long code. This results follows since the two-sided spectra of a long PN-code is $P_I T_c$ near $f=0$ and the two-sided noise bandwidth of the integrate-and-dump filter is $1/T$. The two spectra centered at $\pm\Omega$ add in the I&D filter bandwidth. The output SNR is given by

$$SNR_0 = \frac{\left(\overline{e_0}\right)^2}{\overline{\left(e_I - \overline{e_I}\right)^2}} = \frac{P}{P_I}\left(\frac{R_c}{R}\right) \tag{3.11}$$

where $R_c = 1/T_c$ and $R = 1/T$. The effective reduction of the interference is thus R_c/R and is called the processing gain. Hence higher PN-code rates reduce the interference to a greater degree.

3.3 Multiple access interference

In the previous paragraphs we assumed a one-user system for convenience. With the one-user system is was clear that the correlator counts the difference between the number of agreements and the number of disagreements. However, in practice there're more simultaneous users active and hence the signal entering the receiver doesn't contain only -1's and 1's anymore but contains a set of discrete values so it is difficult to talk about agreements and disagreements now. We have seen that the correlation function of the one-user system has a sharp peak for $\tau = 0$ and low values for $\tau \neq 0$. The more users are active, the lower the distance between the peak value and the low values of the correlation function. Worst case situation occurs when all 127 users are active and all transmitting the same symbol ("1" or "0"). In this case all transmitters transmit their (unmodulated) Gold-sequence and will add up to a sequence which contains again only 1's and -1's. When this resulting sequence is correlated with an arbitrary replica sequence, at the receiver the correlator outputs 1 or -1 and the decision detector detects a "1" or "0" respectively. If during transmission one chip is corrupted by channel noise, the correlator output becomes -1 or 1 respectively, so the output bit can be corrupted by one single chip error and the system performance becomes more poor than a conventional BPSK system! However, this is very theoretical and this situation will almost never occur. If the probability of a "1" and a "-1" equals ½, the probability that all users transmit a "1" or a "0" equals $2 * (½)^{127} \approx 10^{-38}$ which is far below any BER in telecom systems so this situation is not taken in consideration anymore. But it is clear that the number of users has a great influence on the correlation peak[1].

The received signal at the receiver can be expressed as

$$r(t) = \sum_{i=1}^{N_u} A \cdot b_i(t) \cdot c_i^N(t) \tag{3.12}$$

assuming no noise and modulation for convenience. Where N_u is the number of active users, A is the amplitude, c is the code sequence of user i with length N and b is the bipolar bitstream of user i which can be expressed by

$$b_i(t) = \sum_{k=-\infty}^{\infty} b_{k,i} \cdot p(t - kT) \tag{3.13}$$

where $p(t)$ is an unit-amplitude rectangular pulse, T is the bit time and $b_{k,i}$ is a series of bipolar data bits of user i. The output of the correlator of user 0 can be expressed as

[1] If we look back at figure 2.2, this effect reveals at bit 3 of the input sequence. When the input bits of the users are different the correlator outputs 8 or -8 and when both users transmit the same bit, the correlator outputs 6 or -6. The magnitude of this effect is very small here since there are just two users active.

$$
\begin{aligned}
s(t) &= \sum_{n=0}^{N_c-1} c_{0,n}^{N}(t)\cdot r(t) = \sum_{n=0}^{Nc-1} c_{0,n}^{N}(t)\cdot \sum_{i=0}^{Nu-1} A\cdot b_i(t)\cdot c_{i,n}^{N}(t) \\
&= \sum_{n=0}^{Nc-1} \left\{ c_{0,n}^{N}(t)\cdot A\cdot b_0(t)\cdot c_{0,n}^{N}(t) \right\} + \sum_{n=0}^{Nc-1}\sum_{i=1}^{Nu-1} A\cdot b_i(t)\cdot c_{i,n}^{N}(t)\cdot c_{0,n}^{N}(t) \\
&= N_c\cdot A\cdot b_0(t) + \sum_{n=0}^{Nc-1}\sum_{i=1}^{Nu-1} A\cdot b_i(t)\cdot c_{i,n}^{N}(t)\cdot c_{0,n}^{N}(t)
\end{aligned}
\tag{3.14}
$$

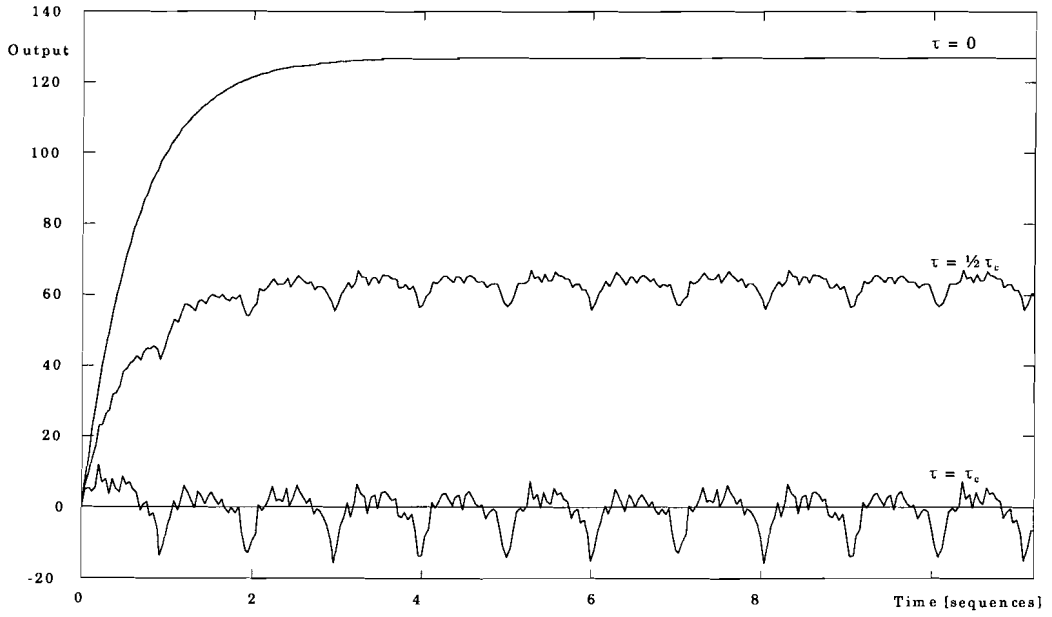
where the first term is the desired bit stream and the second term is the multi access interference (MAI) due to non perfect orthogonality of the codes. So besides channel noise and narrowband interference, MAI is an undesired phenomenon. The more users the more MAI and hence the greater the BER.

4. Implementation of the correlator

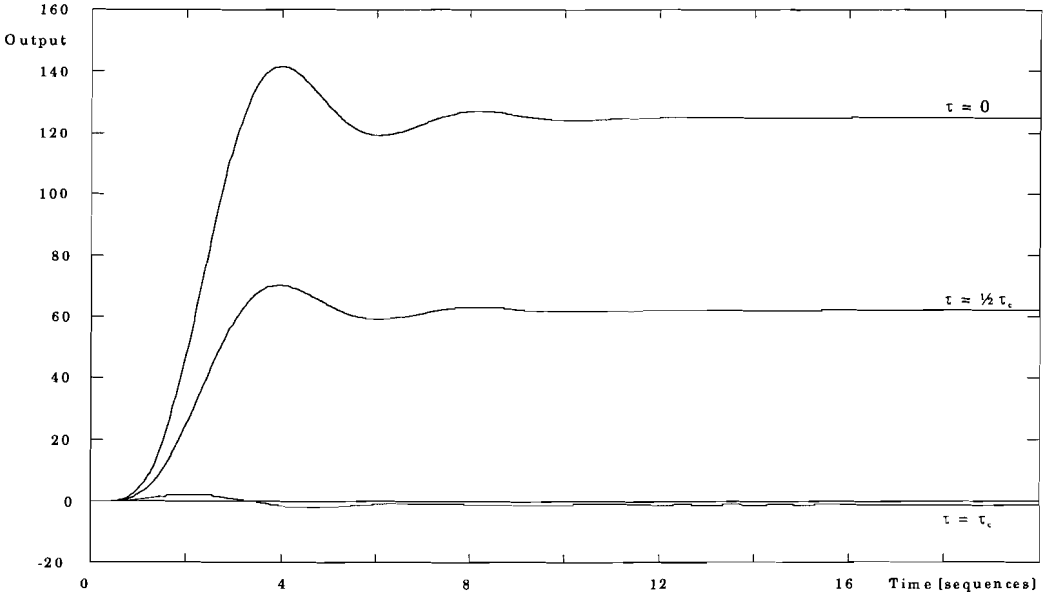
4.1 Correlator with an I&D-filter and with a low pass filter

As we've seen in the previous sections there are two ways to look at the correlator. In figure 2.1 the working of the correlator was explained in frequency domain, where the multiplier is followed by a low pass filter. In the last section we've paid attention to the correlation functions which are in time domain and here the multiplier is followed by an integrate and dump filter. In practice both implementations will do, finally an integrate and dump filter approximates a LPF (after sampling). However, the principle is different.

In almost all published literature correlators are represented as a multiplier and a filter. However, little attention has been paid to the form of this filter and the impact it has on the system performance. Street [8] investigated the effects of different filters. The results suggest that the low pass filter parameters have a dramatic effect on the correlation function. Street also suggests that the cut-off frequency must increase if the filter order increases. Own research suggests that a higher order filter gives better correlation values when measuring the correlation over a certain time but has a certain settling time which becomes important when this filter has to recover a data stream. Figure 4.1 shows the output of a single and a five pole Butterworth filter when Gold code 0 is used as correlation signal. The output of the 5-pole Butterworth filter approximates the theoretical values very well (127 at $\tau = 0$, 63 at $\tau = \frac{1}{2}\tau_c$ and -1 at $\tau = \tau_c$, see also figure 3.1). The output of the 1-pole Butterworth filter is much worse and does not stabilize to an end value. The values at $\tau = 0$ are trivial since the output of the multiplier equals 1 and the filter output equals the unit step response.



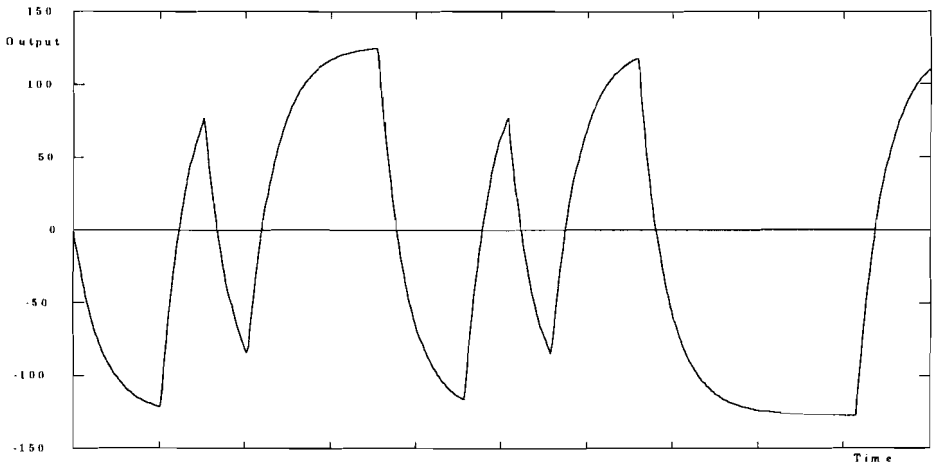
a. 1-pole Butterworth responses ($f_0 = (1/127).R_c$).



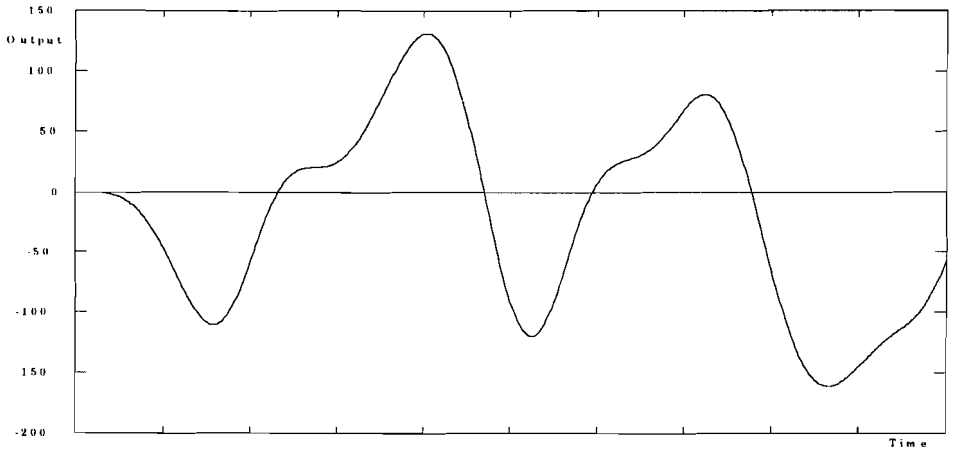
b. 5-pole Butterworth responses ($f_0 = (1/127).R_c$).

Figure 4.1: *Correlation with a 1 and 5-pole Butterworth low pass filter.*

The results in figure 4.1 were achieved by repeating the same input sequences with a certain constant delay τ. After ± 12 sequences the output is stabilized to its end value in case of a 5-pole filter. In spite of the fact that the 5-pole filter gives the best correlation values, this filter is not suitable for a data recovery circuit with these parameters. Figure 4.2 illustrates the output of the filters when the sequences are modulated by a bitstream.



a. Filter output using a 1-pole Butterworth filter.



b. Filter output using a 5-pole Butterworth filter.

Figure 4.2: *Filter output when input sequences are modulated by a bitstream.*

The sequences were modulated with the following bitstream: *00101110010110000011*. If we take a look at figure 4.2 we see that the 1-pole Butterworth is able to reconstruct the bitstream where the 5-pole Butterworth (with the same cut-off frequency) fails. However, when more simultaneous sequences (users) are considered, the 1-pole filter will also fail. It must be remarked that the 5-pole filter in figure 4.2 can be improved by taking a higher cut-off frequency and can lead to the modulated bitstream again. So a compromise must be made between cut-off frequency and filter order to realize a *symbol matched filter* (without ISI).

Another simulation was done. Here a 5-pole Butterworth filter with a cut-off frequency of $1/64.R_c$ is used to recover data from a CDMA signal containing 128 users. In this simulation an I&D-filter was simulated as well. Figure 4.3 shows the simulation set-up and figure 4.4. shows the output of the LPF and the I&D-filter for a time range of 12 bits.

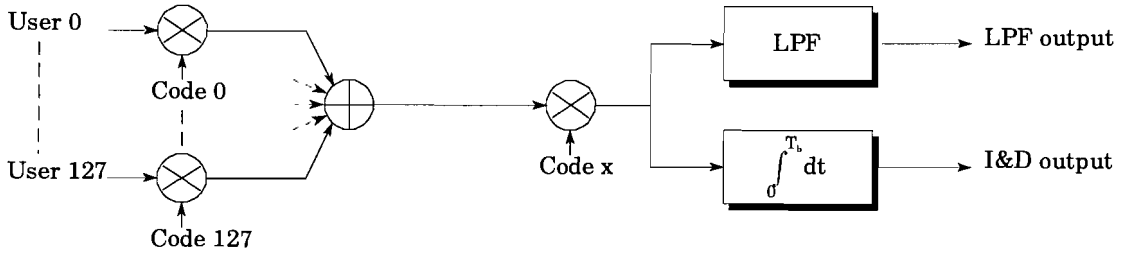


Figure 4.3: *Simulation set-up of LPF and I&D-filter.*

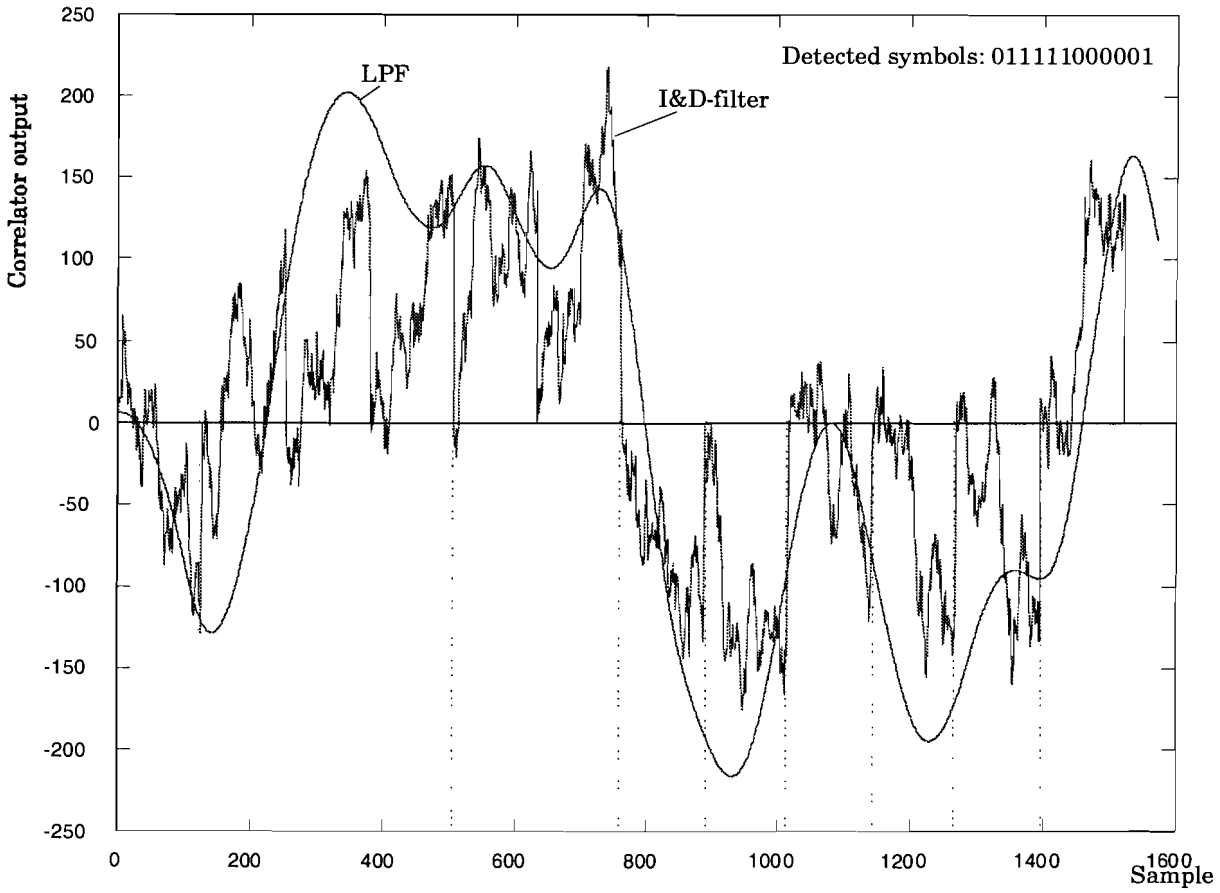


Figure 4.4: *Correlator output with an I&D and with a LPF.*

Figure 4.4 illustrates the output signal of the correlator using an I&D-filter and LPF of one single user calculated by a simulation to illustrate that both implementations will lead to the same detected bits. In this simulation the input signal contains 128 users transmitting random data and is correlated with an arbitrary Gold code of one of the users. The simulation is done in baseband (no modulation) and no noise is assumed. The first 12 bits (12 * 127 = 1524 samples) are plotted in the figure. The sample instants are marked in the figure by the dotted lines. As can be seen in the figure, both filters lead to the same detected bits, however, longer simulations indicated some errors for the LPF (so again the LPF was not an optimized matched filter) and no errors for the I&D-filter. The simulation source code is printed in appendix A. Figure 4.5 depicts the transfer

function of the five pole Butterworth filter (normalized frequency 1 corresponds with ½R_c) .

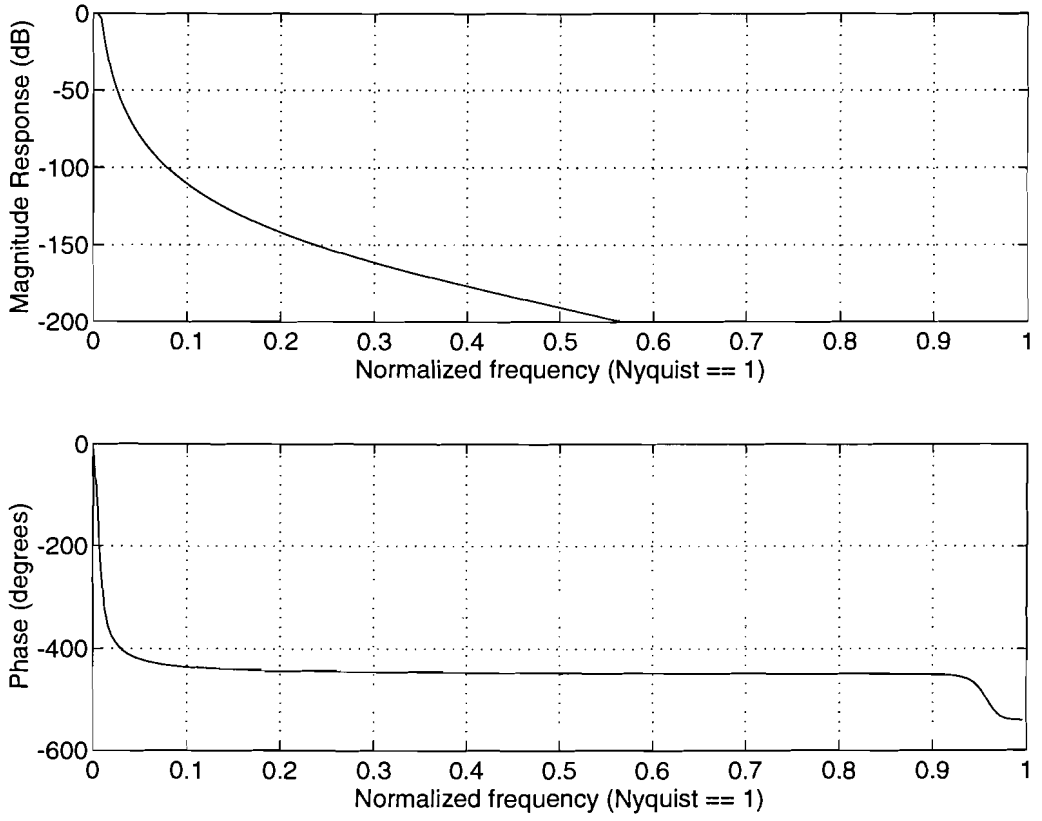


Figure 4.5: *Transfer function of 5 pole Butterworth filter* $f_0 = (1/64).f_{chip}$

If a CDMA-modem is realized in digital domain [4], it's obvious to use an accumulator as I&D-filter but in analog domain a compromise has to be made between complexity and performance. However, an I&D-filter not necessarily needs more hardware components. Unlike a LPF, an I&D-filter needs control signals for resetting the filter to zero. But the hardware needed for this control signals can be shared by all I&D-filters and can be housed in the PLD chip which is already present for generating Gold-codes anyway. Furthermore, the parameters of an I&D matched filter are much easier to determine. In the next paragraphs we consider the hardware of the correlator.

4.2 Integrate and dump circuits

As was described before the correlator consists of a multiplier and an integrator. For now we only consider the I&D-filter because this is the most difficult part. In the previous paragraph we discussed the LPF and I&D-filter properties. It was proposed to use an I&D-filter rather than a LPF. An I&D-filter is easier to implement than a Butterworth filter because the additional digital hardware needed for the I&D-filter can be housed in an ASIC which is already present for Gold-code generating. Furthermore, the

parameters of the I&D filter are less critical than the parameters of a LPF. Figure 4.6 shows some I&D circuits.

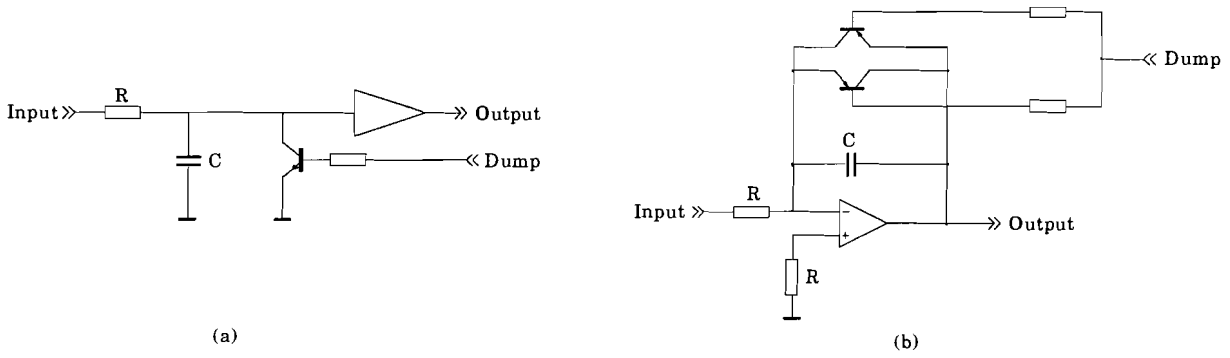


Figure 4.6: *Integrate and dump circuits.*

In figure 4.6a, the most simple implementation of an I&D circuit, a transistor is used as switch. Disadvantage of this circuit is that the input current depends on the capacitor voltage so the input to output voltage transfer function of this circuit contains a differential equation with an exponential transfer function as solution. For the correlator we want a (linear) summation so a linear integration is wanted. The reason for this is that the complex phase rotator needs the values of a linear correlation process in order to remove the rest modulation. Figure 4.6b depicts an active integration circuit which can handle bipolar signals [21]. Also here transistors are used as switch to discharge the capacitor. The transistors are connected in anti-parallel so one of them is biased in forward mode independently of the output polarity. Both transistors turn on during the dump interval. The transistor in forward mode determines the initial discharge rate until it saturates, after which the inverted mode transistor continues to discharge the capacitor. Here the input current is independent of the capacitor voltage due to the virtual ground potential on the inverting input of the OPAMP. The capacitor voltage is given by:

$$V_c(t) = \frac{1}{C}\int i(t)\mathrm{d}t \tag{4.1}$$

where the capacitor current $i(t)$ equals the input current. The input current equals $i(t) = V_{in} / R$. So the transfer function of this circuit becomes

$$V_c(t) = \frac{1}{RC}\int V_i(t)\mathrm{d}t \tag{4.2}$$

Unlike circuit 4.6a, circuit 4.6b is a linear integrator circuit and is therefore used in the correlator.

The values of R and C must be chosen very carefully. First the RC product has to be chosen such that a maximum input signal cannot saturate the OPAMP. If we compare equation 3.5 with equation 4.2, RC must be equal to T, the symbol time (if the input signal varies between +1 and -1 volts the maximum output voltage becomes 1 volt). The capacitor must have a small value in order to obtain fast dump times. On the other hand, when the capacitor becomes too small, the junction capacitors of the transistors

will influence the circuit. So a compromise of R and C values has to be made (see appendix B for the circuits).

4.3 Effects of imperfect integration

Fast I&D-filters are uncommon, as they require the integrator be reset in a fraction of the integration period [10]. When the local replica code generator is at the end of its sequence, the output of the I&D-filter has to be sampled prior to the dump instant. All this must happen within one chip time. Every type of analog I&D-filter has to do with the so called *"dead time"*. This is the time needed to discharge the capacitor and has to be small compared to the integration time [10,11]. If we want to process high bit rates, in practical circuits the dead time of the I&D-filter becomes greater than one chip time and some chips of the sequence will be discarded. To determine the degradation of the BER as function of the discarded chips caused by imperfect integration, a simulation program was written (see appendix A). This program calculates the BER using the Monte Carlo simulation method (therefore low E_b/N_0 ratio's were chosen to avoid very long simulation run times). The simulations are done with constant SNR ratio and different number of users and with a constant number of users and different SNR's. During the simulations, perfect synchronization of all users is assumed. The range of the simulation is from 127 chips (perfect integration) to 64 chips (one half of one sequence, which is very unacceptable). Figure 4.7 shows the simulation results. The values at 127 chips match the theoretical values of the BER [5]. To achieve a BER of 10^{-4} a E_b/N_0 ratio greater than 8.5 dB is needed with perfect synchronization [5]. In practice a greater E_b/N_0 ratio will be needed to obtain BER less than 10^{-4} due to imperfect synchronization.

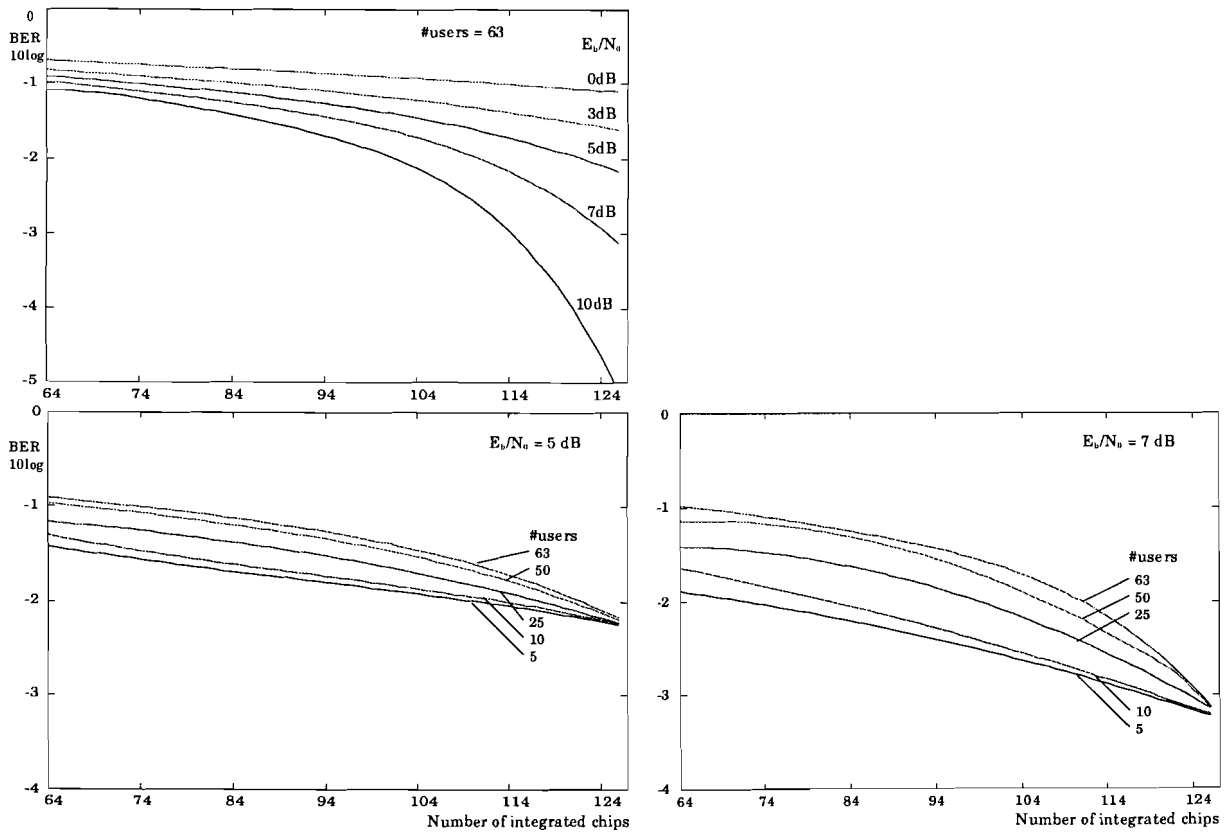


Figure 4.7: *Simulation results of imperfect integration*

4.3 Dual integrate and dump circuit

According to figure 4.7 the BER will increase rapidly when some chips are discarded, especially when there are a lot of users active. If we don't want to discard any chip of the sequence and we can't make a faster dump circuit, we have to look for another solution.

An idea was to make an alternating I&D circuit as depicted in figure 4.8. When the local replica code generator is at the end of its sequence, the input signal is switched to a second I&D circuit and starts a new integration. Meanwhile the first I&D circuit can be sampled by an A/D-converter and can be dumped afterwards.

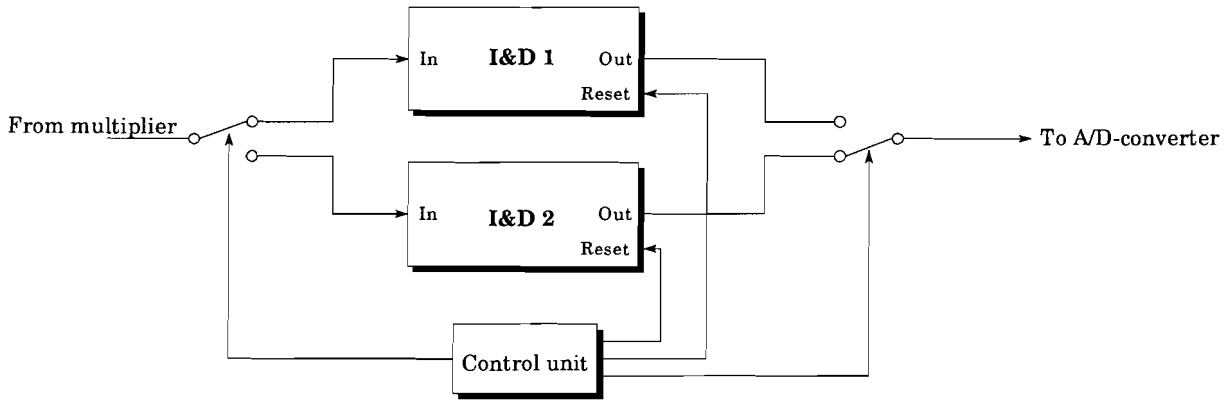


Figure 4.8: *Dual I&D circuit.*

With this configuration the sum of sample- and dead time can be as long as the total sequence length. An important condition is that the settling time of the switches is smaller than one chip time. Another advantage of this configuration is the hold function. The CDMA modem consists of 8 correlators. An A/D-converter for each correlator will drive up the price too much. With this hold function it is possible to share the A/D-converter by all I&D-circuits without external hold circuits. The A/D-converter can sample the I&D circuits successively.

The settling time of the used switches is approximately 10 ns. This means that at a chiprate of 100 Mchips/s (± 1.5 Mbit/s) only one chip will be missed and this is acceptable considering figure 4.7.

4.4 Control hardware for the integrate and dump circuit

As mentioned before, the integrate and dump circuit requires some additional control signals to determine the sample instant and to discharge the capacitor. When the local Gold-code generator is at the end of its sequence, the output of the I&D-filter has to be sampled by an A/D-converter prior to the dump instant. The control signals for the I&D circuit are generated by some digital hardware. The states of the control signals depend on the state of the Gold-code generator. A counter counts the clock pulses of the Gold-code generator. The output of this counter can be used to trigger some activities. This digital circuit is housed in ALTERA's programmable IC and was developed with ALTERA's software packet *"Max2Plus"* [12]. In appendix B this circuit is depicted. In the table shown below the timing of the control signals are clarified.

Chip count	Activity
0	Switch input to I&D circuit 2 and I&D circuit 1 to output
3	Sample the output (I&D 1)
64	Reset I&D 1
127	Switch input to I&D circuit 1 and I&D circuit 2 to output
129	Sample the output (I&D 2)
190	Reset I&D 2
253	Reset the chip counter

Table 4.1: *Timing sequence of the control signals.*

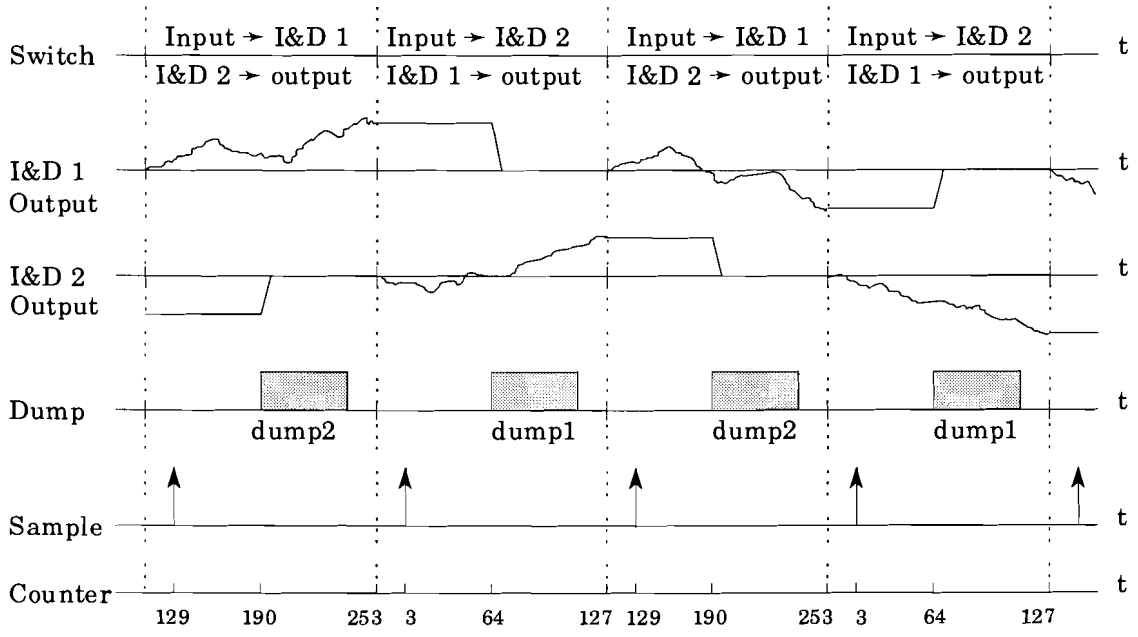


Figure 4.9: *Timing diagram of the I&D-filter.*

Figure 4.9 illustrates the signals of the I&D-circuits in a timing diagram. From this figure we see that the I&D-circuits are alternating. If the switch selects the input to I&D-circuit 1 and I&D-circuit 2 to the output, circuit 1 is in "integrating mode" and circuit 2 is in "hold mode". When the A/D-converter receives a sample pulse, the output will be sampled; in this case circuit 2 which is in the "hold mode" is the output. The timing simulation performed by *"Max2Plus"* is illustrated in appendix C.

5. Measurements

5.1 Description of measurements methods

First it was proposed to check the working of the entire circuit with the same clock frequency as the conventional digital correlator (4 Mchips/s) and then test the circuit with increasing clock frequencies. To test the circuit, it was proposed to measure some auto correlation functions of some Gold codes and compare these with the theoretical ones. Therefore we need another synchronic Gold code signal to correlate with (see figure 5.1). For test purposes, a second Gold-code generator is applied and is also housed in ALTERA's PLD. A delay circuit is developed to delay the second Gold-code sequence so we can determine the auto- or cross correlation function by changing the delay and measuring the output of the correlator. The correlation of only one sequence can tell us a lot about the performance of the correlator. The entire circuit is depicted in appendix B and can be consulted for more details. For test purposes a selection can be made between the analog multiplier and an ideal digital multiplier (EX-NOR-gate) inside the digital hardware to test the multiplier performance. Note that two digital signals can be multiplied by an EX-NOR-gate unlike the analog signal of several users. Measurements were done using the configuration shown in figure 5.1.

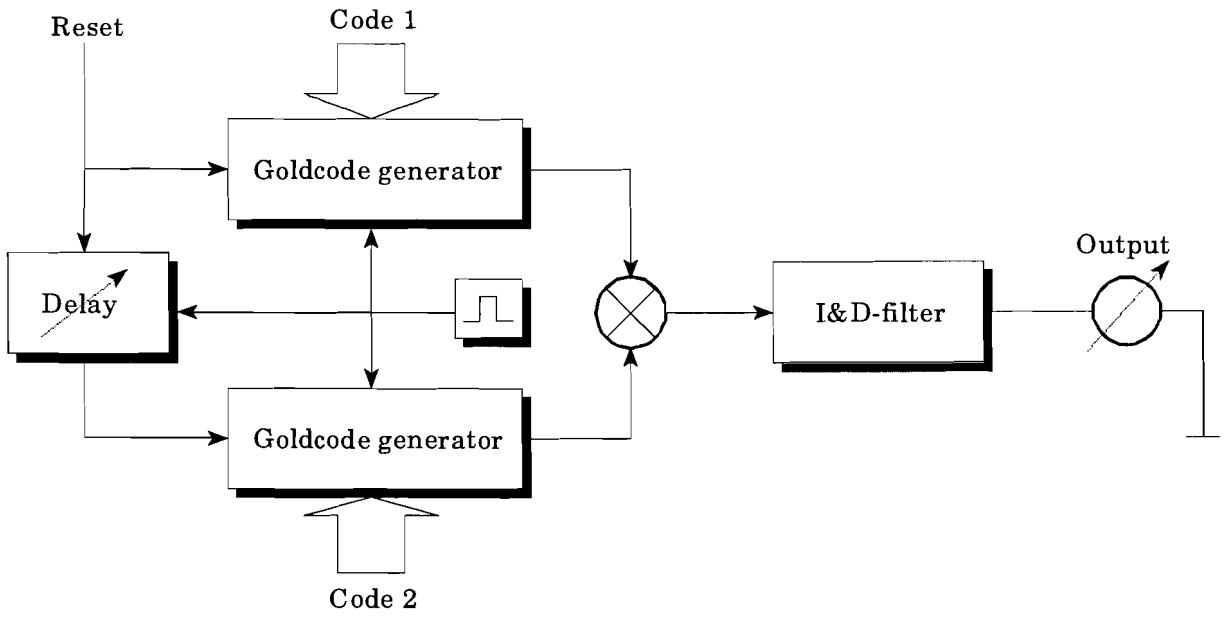


Figure 5.1: *Configuration of the test circuit.*

5.2 First measurement: conventional chiprate

To get a reasonable impression of the correlator performance, auto correlation functions of different Gold codes were measured. Figure 5.2 shows the auto correlation function of Gold code 0, 1 and 64. These are the calculated theoretical values.

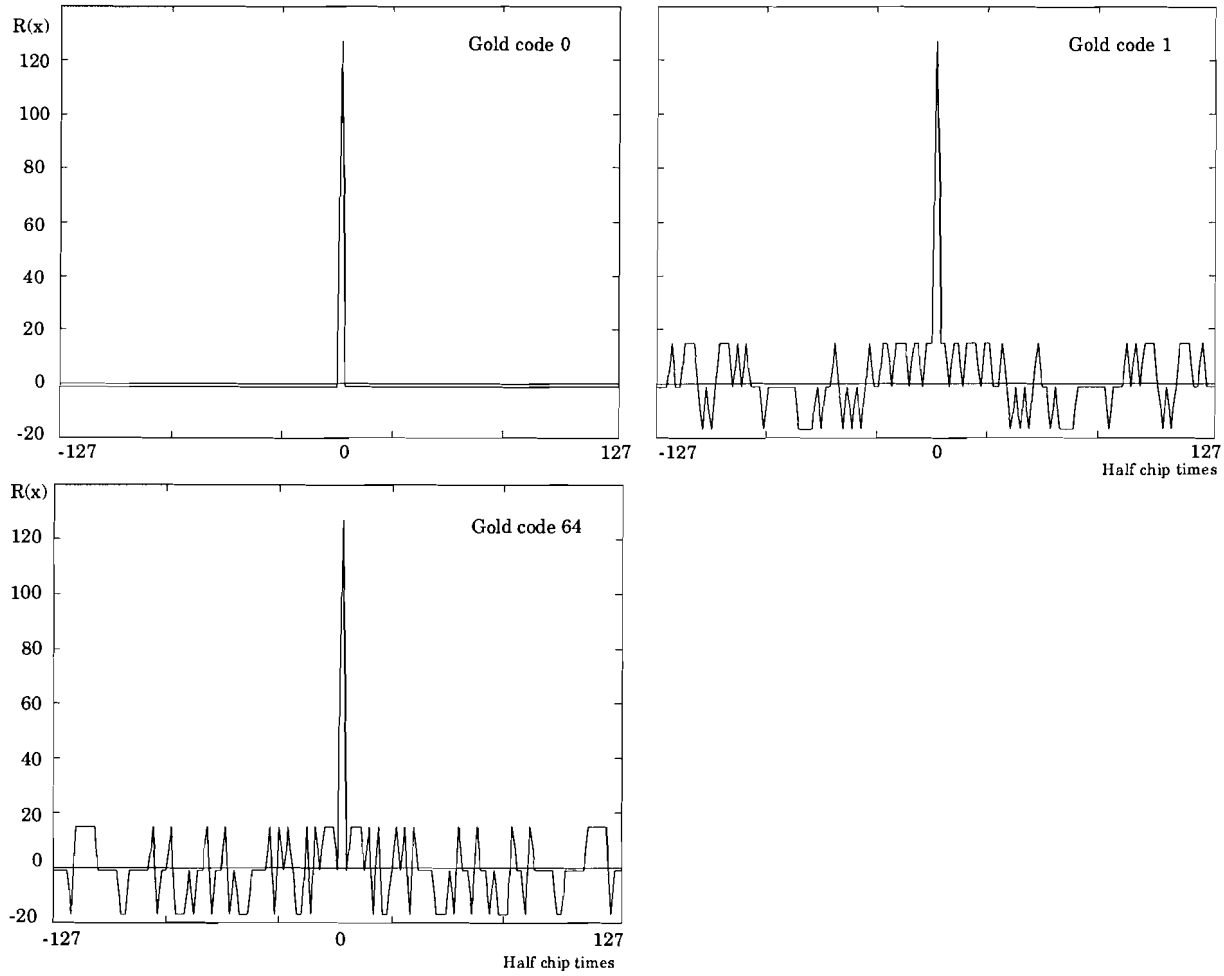


Figure 5.2: *Auto correlation functions of Gold code 0, 1 and 64.*

The measurement were done with steps of ½ chip times. This is a trade-off between clock frequency and accuracy. With ½ chip times increments, the clock frequency of the digital hardware must be twice the chiprate. The sequence length amounts 127 so 254 measurements can be done for each code sequence. Measuring all values of 254 delays will take to much time so 30 values were measured (15 values on both sides of zero). This gives enough information about the performance of the correlator. Figure 5.3 shows the magnification of the same auto correlation functions as depicted in figure 5.2 (dashed lines) near zero. Also the measured values are plotted in the same figures (solid lines). The measured values are scaled such that the mean of the maximum output voltages match 127.

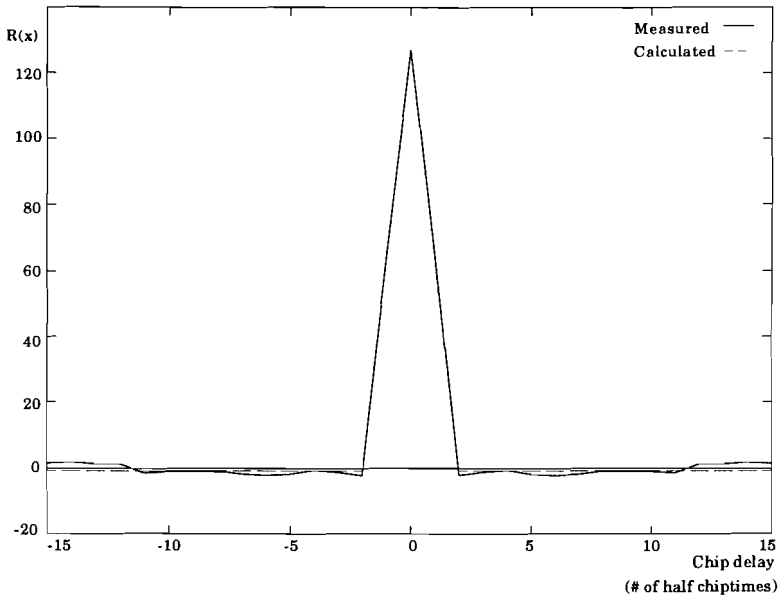
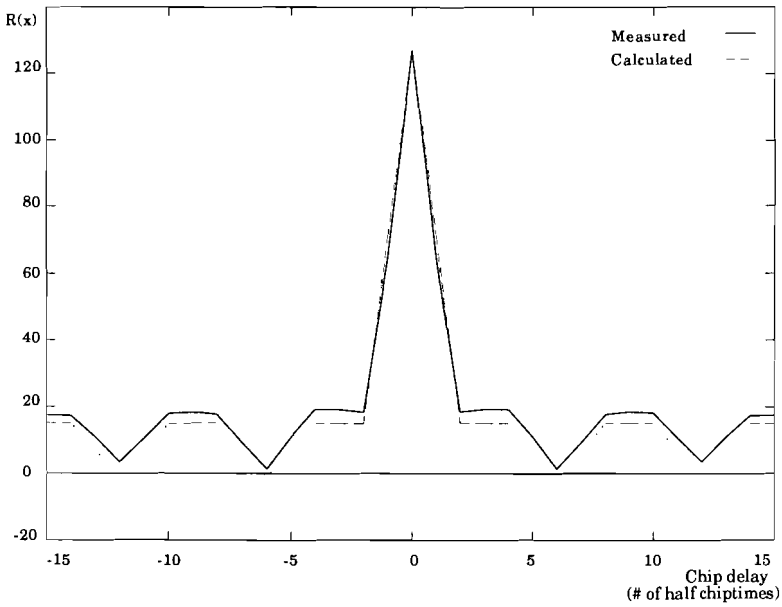
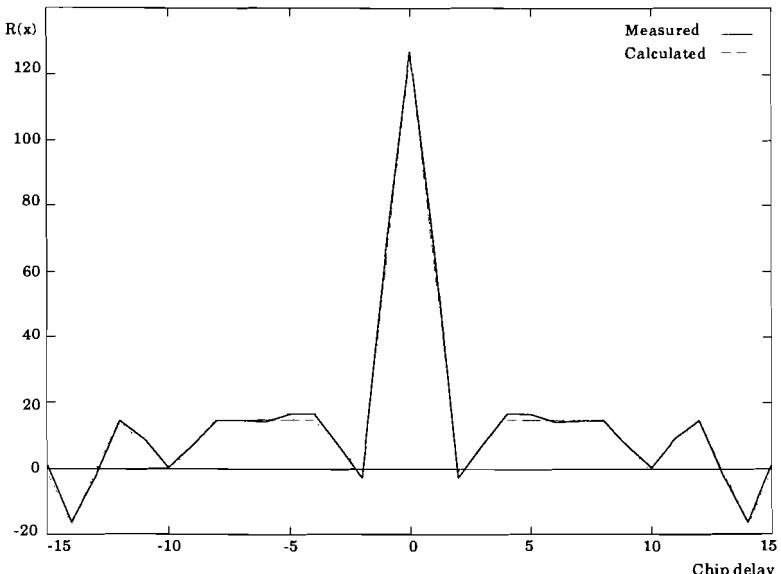


Figure 5.3: *Measurements*
Chiprate = 5 Mchips / s

a. Auto correlation Goldcode 0



b. Auto correlation Goldcode 1



c. Auto correlation Goldcode 64

From figure 5.3 it becomes clear that the realized analog correlator is able to approximate the theoretical values very well. The auto correlation function of each Gold code is different but they have some properties in common. They are all symmetrical with respect to zero and have a peak value of 127 and a half peak value at $\pm\frac{1}{2}\tau_c$. The remaining part of the auto correlation function consists of the five discrete values -17, -9, -1, 7 and 15 except code 0 which has only -1's besides the main peak. Because of the fact that all other Gold codes have the same properties, it would not be necessary to measure all code sequence to get an impression of the correlator performance.

5.3 Second measurement: higher chiprates

The measured values depicted in figure 5.3 were measured at a chiprate of approximately 5 Mchips/s; a little higher than the conventional chiprate and these measurements match the theory very well. We can conclude the electrical circuit operates well and now we can start determining the circuit behavior with increasing clock frequencies. Therefore only the clock frequency has to be increased and the resistors and the capacitors (integration constant) of the I&D dump circuits has to be adjusted to obtain maximum dynamic range.

At the second measurement, the clock frequency was doubled to 10 Mchips/s. Measurements of the auto correlation doesn't deviate much from figure 5.3 so these results are not revealed.

Next measurement the clock frequency is doubled again to 20 Mchips/s and again the auto correlation functions of Goldcode 0, 1 and 64 are measured. The results are plotted in figure 5.4. These results are still acceptable.

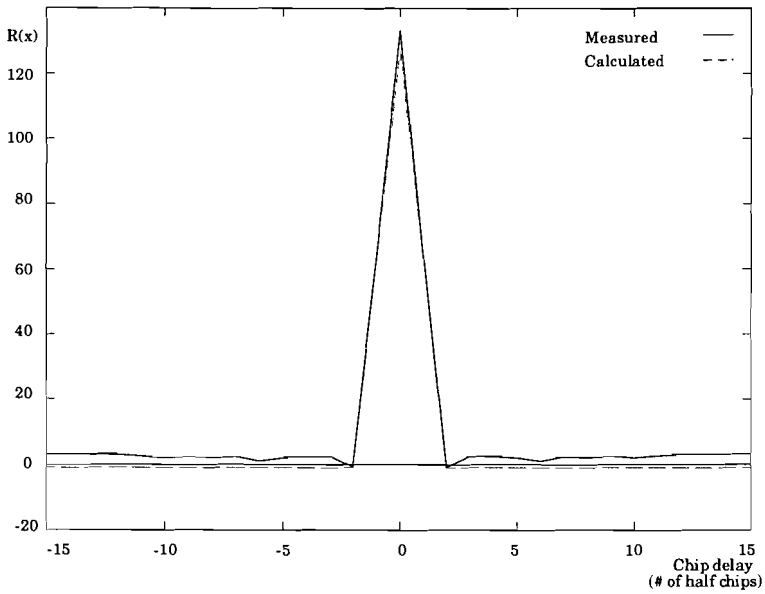
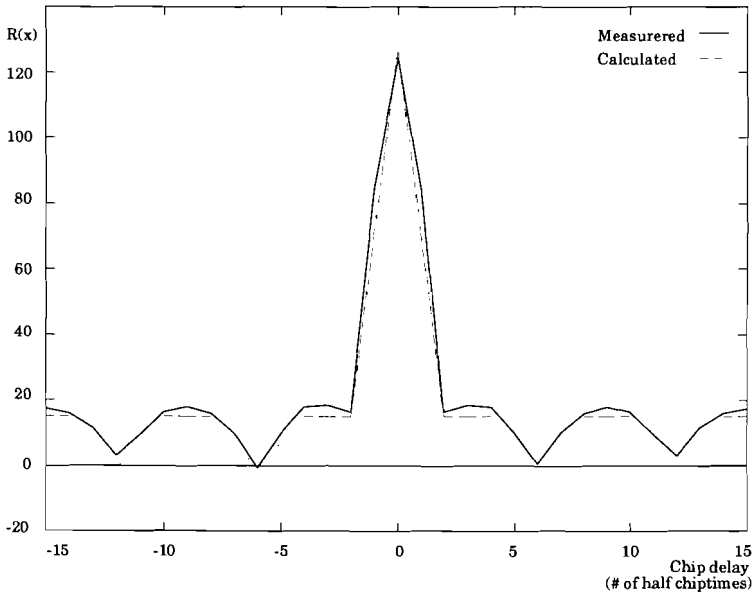
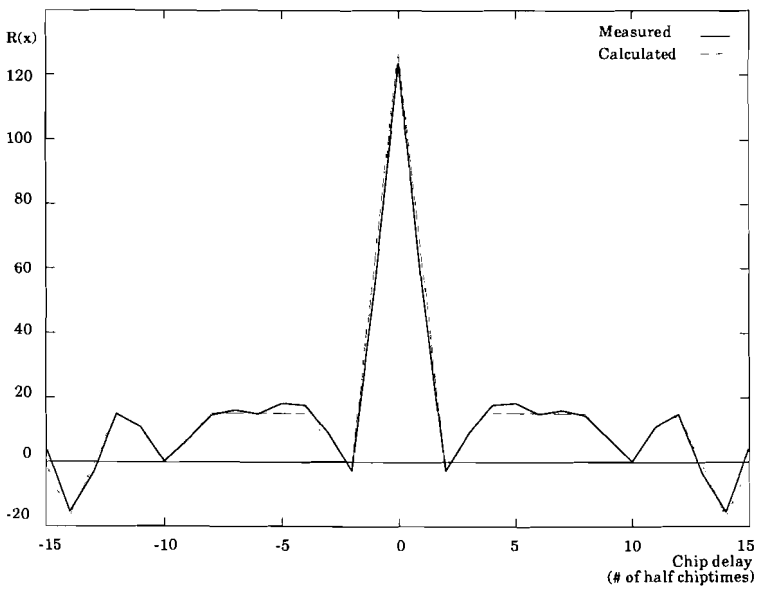


Figure 5.4: *Measurements Chiprate = 20 Mchips/s*

a. *Auto correlation Goldcode 0*



b. *Auto correlation Goldcode 1*



c. *Auto correlation Goldcode 64*

If we compare the measurements at 5 Mchips/s and at 20 Mchips/s we observe almost no differences in the auto correlation functions. Figure 5.5 shows the results of the measurements at 25 Mchips/s (only Gold code 1) and figure 5.6 shows the results of the measurements at 50 Mchips/s.

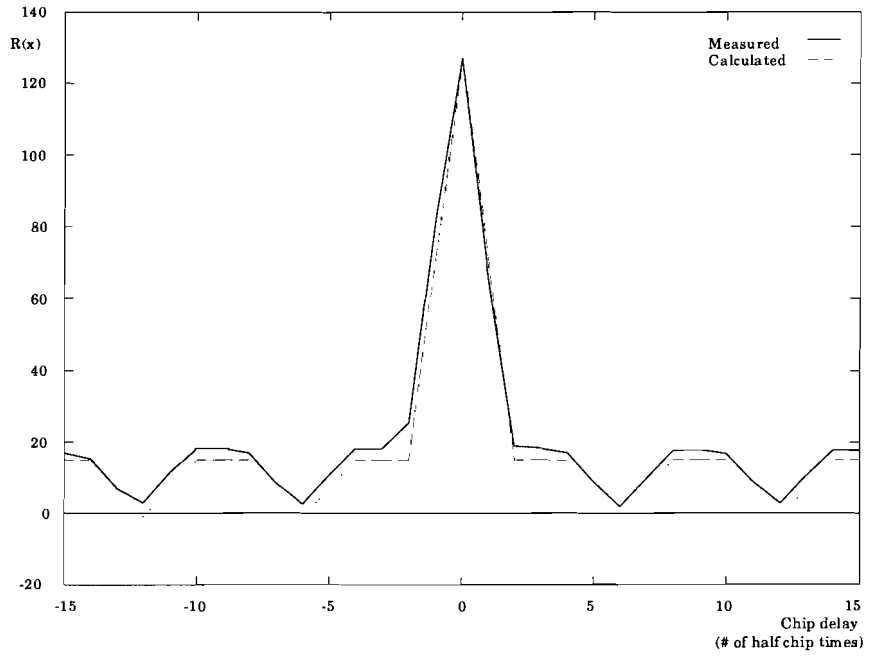


Figure 5.5: *Measurement at 25 Mchips/s (auto correlation code 1).*

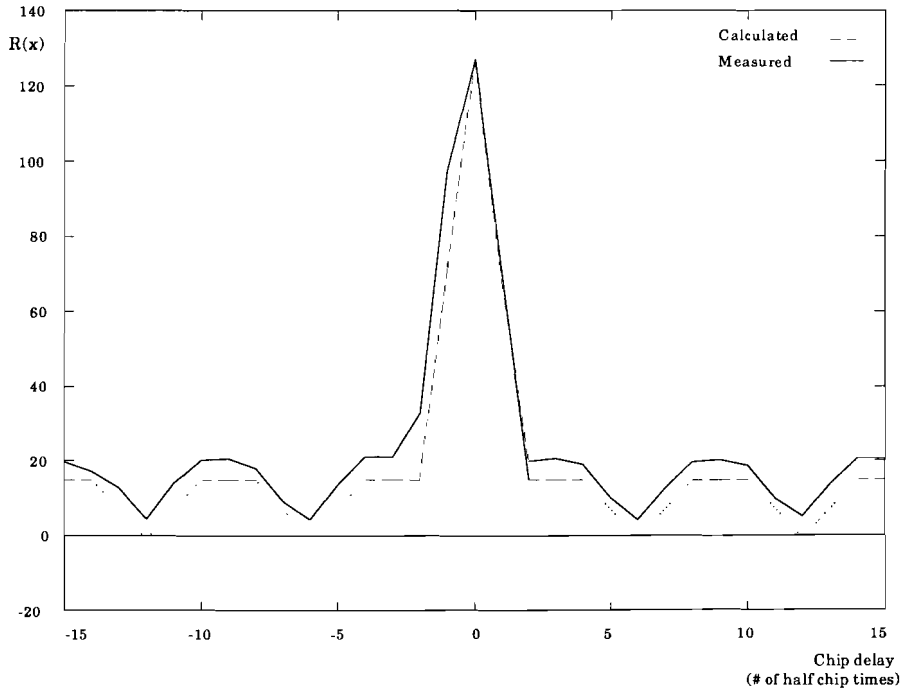


Figure 5.6: *Measurement at 50 Mchips/s (auto correlation code 1).*

According to figure 5.6 the measured auto correlation of Gold code 1 deviates from the theoretical auto correlation very much at one point i.e. $-\tau_c$. However, we must beware of the fact that this performance degradation is not necessarily caused by the correlator

itself but by the pulse shapes of the input signals. If we want to compare the measured values with the theoretical values, rectangular pulse shapes are necessary. Finally we want to get an impression of the performance of the correlator and therefore we must take care that the input signals are as rectangular as possible during the measurements. In practice these signals are never rectangular since the signals are pulse shaped, modulated, demodulated and filtered before they enter the correlator.

Figure 5.7 shows the results of the measurements of the pulse shapes. The second waveform from the top is the "reference" Gold sequence and the waveform below is the same Gold sequence a ½ chip time delayed and advanced respectively. The bottom waveform represents the output of the mixer. If we compare both mixed waveforms we observe some differences considering the areas of the signal below the baseline. The areas of the first figure are smaller than the areas in the second figure. This explains the deviation in the auto correlation measurement.

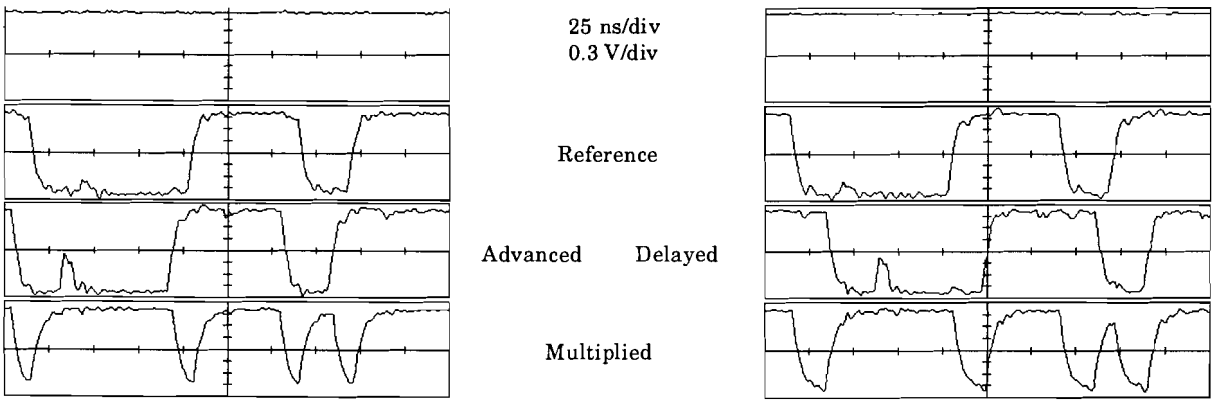


Figure 5.7: *Measurements of the pulse shapes at 50 Mchips/s.*

Figure 5.8 shows the results of the measurements of waveforms at symbol time scale. The top waveform in figure 5.8a represents the switch signal determining which I&D-circuit is selected.

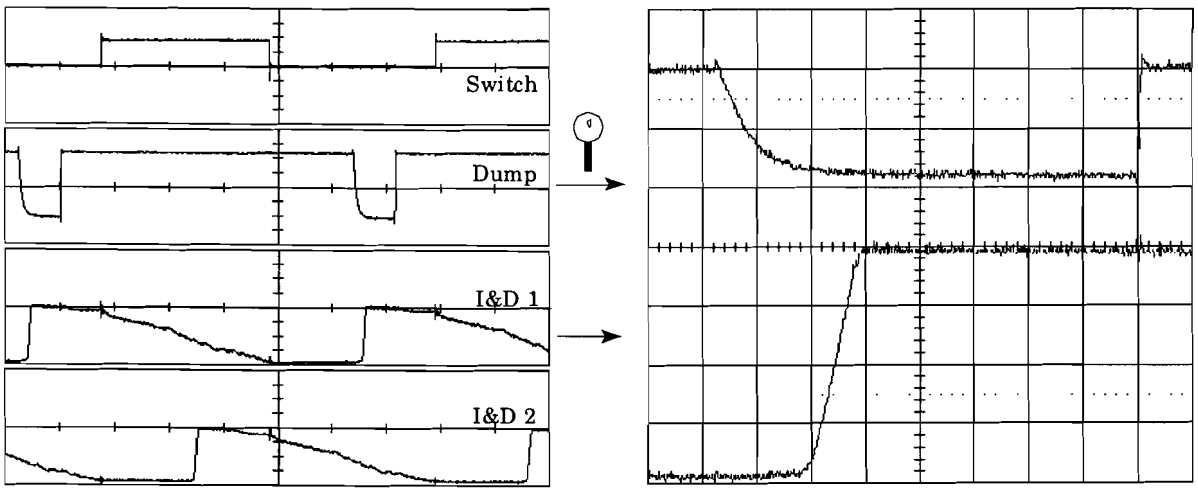


Figure 5.8: *a) Measurements at symbol time scale.*
b) Magnification of dump pulse and I&D output at the reset instant..

The second waveform from top represents the dump pulse for I&D-circuit 1. The two bottom wave-forms represent the output of I&D circuit 1 and 2 respectively. The measurements of figure 5.8a are conform figure 4.9. Figure 5.8b is a magnification of the second and third waveform to determine how much time it takes to discharge the capacitor of the I&D-circuit. The timebase of figure 5.8b was set to 100 ns/div so the dump time is approximately 100 ns. This measurement was done with zero delay so the output of the correlator before dumping reached its maximum (negative) voltage level and hence the 100 ns dump time is also the maximum dump time. From this we can conclude that the we can shorten the dump pulse and delay it to a later time instant so the hold time becomes larger and hence slower A/D-converters can be used if want to use one A/D-converter for more correlators.

The dump pulse can be reduced to at least 400 ns (fall time included). The time of one sequence at 50 Mchips/s amounts 2540 ns. If we put the dump pulse as late as possible (i.e. just before switching from I&D 1 to 2), the remaining hold time amounts 2540 - 400 = 2140 ns. If one A/D-converter has to sample four correlators, the sampling period must be at least 2140 / 4 = 635 ns and the sample rate of the A/D-converter at least 1.6 MHz. Compare this with the digital CDMA modem which operates at 4 Mchips/s using a 16 MHz A/D-converter.

The chiprate of 50 Mchips/s is the highest chiprate we can achieve at the moment since the maximum clock speed of the PLD is 100 MHz and the clock speed is twice the chiprate. The bitrate at 50 Mchips/s amounts 790 kbit/s approximately using QPSK.

5.3 BER measurements

The final measurements include the BER measurements to get an impression of the performance of the correlator. The problem here is that there is no high speed CDMA transmitter available yet. Furthermore, a lot of transmitters are needed to obtain a realistic system. The solution to these problems is solved by making use of an *arbitrary waveform generator*. This is a signal generator which can generate a user defined signal waveform which is stored on a floppy disk. With a computer program an output file can be generated containing an arbitrary number of users transmitting an arbitrary PRBS. With a BER-tester connected to the output of the correlator we can measure the BER. In figure 5.9 this measurement setup is depicted.

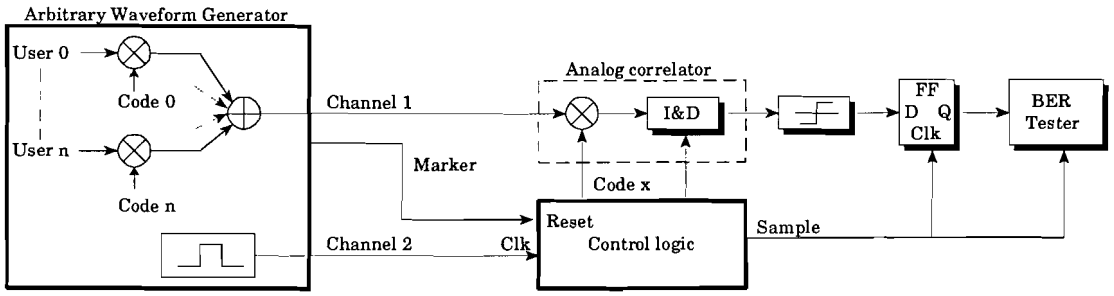


Figure 5.9: *BER measurement setup.*

Considering this measurement, we're not interested in BER's as function of SNR's but in BER's as function of the number of users because we want to test the correlator performance and not an entire CDMA-system. Since no noise is added during this measurements we expect no errors (theoretically), even if all users are active.

The Arbitrary waveform generator (AWG) has two channels which can be programmed separately. Channel 1 outputs the CDMA signal (simulating the network output) and channel 2 outputs the clock signal so the analog correlator and the CDMA signal are clocked by the same clock. The AWG also outputs a *marker signal* indicating the beginning of the programmed sequence. The correlator will be triggered by connecting this marker signal to the reset input of the analog correlator so the code sequence of the correlator and the AWG are synchronized as well.

The BER tester can not be connected to the transmitter as usual in common BER measurements because the AWG can not calculate a CDMA signal in real time. Therefore the PRBS used in the BER tester must be known in order to program the same PRBS in the AWG (multiplied with a Gold code). The BER tester has different types of PRBS lengths. We've chosen for PRBS length 63 and can be realized by a six stage linear feedback register according to figure 5.10. The generated PRBS is multiplied with Gold code 0 and the other users get their random data from a data file. Only correlating with Gold code 0 will lead to the PRBS again which can be recognized by the BER-tester.

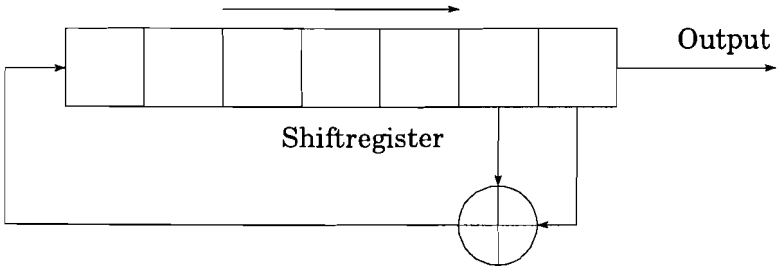


Figure 5.10: *PRBS generator (sequence length = 63).*

In order to connect the BER-tester to the correlator, a zero-detector and a flip-flop are inserted (see appendix B for circuit details). The zero-detector has to decide whether the output is positive or negative and output a "0" or a "1" respectively. The flip-flop takes care of the fact that the decision is made on the right instant and holds the data for one bit time.

The AWG was loaded with files containing the CDMA-signal with different number of users. The number of users was increased by ten users for every file. The measurements have turned out that the CDMA-signals containing user-amounts up to 80 can be correlated without errors (measurements were aborted after 10^{10} error free transmitted bits). Higher amounts of users leaded to bit errors. It is not useful to express these errors in a BER because the sequence length is 63 and the bit error occurred at the same place in the sequence every time. The zero-detector takes a wrong decision at these moments so it seems that the MAI is that large that the correlator suffers from this great MAI and the output of the correlator gets the opposite polarity. The location of the wrong bit in

the sequence was traced and the same CDMA-signal was correlated with Gold code 0 by a computer program to calculate the output of the correlator at the location of the wrong bit. The result of this computer calculation turned out that there are bits in the sequence which have a lower correlation value but were detected without errors! From this we can conclude that the correlation calculated by the computer program and by the analog correlator are not the same. But the calculations made by the analog correlator were in good agreement with the theory (see paragraph 5.2). From this we must conclude that the errors (which occur when the CDMA-signal contains more than 80 users) are caused by a little timing misalignment. So an improvement of the timing will lead to a more better performance.

6. Entire receiver circuit

In the previous chapters all aspects of the analog correlator were considered. In this chapter we will discuss the entire receiver. Some details of the receiver modem have not been discussed yet. In figure 2.4 the digital receiver was depicted. The principle of this receiver was taken as a basis for the analog receiver design.

6.1 Early/late correlator

The early/late correlator takes care of the synchronization. When the early correlation is subtracted from the late correlation, a very accurate synchronization signal can be achieved [4,5]. The early correlation is the correlation of the replica code with the received signal which is $\frac{1}{2}\tau_c$ earlier than the received signal for data detection. The late correlation is the correlation of the replica code with the received signal which is $\frac{1}{2}\tau_c$ later than the received signal for the data detection. As can be seen from figure 2.4, the early signal is not delayed, the data recovery signal is delayed by $\frac{1}{2}\tau_c$ and the late signal is delayed by τ_c.

In digital domain it's very easy to delay a signal unlike in analog domain. In analog domain this can be realized with analog delay lines which have specific delay times. Another method is not to delay the input signal of the correlator but to advance the replica codes. The late correlator gets a replica code which is not delayed, the data correlators get replica codes which are delayed by $\frac{1}{2}\tau_c$ and the early correlator gets a replica code which is delayed by τ_c.

In the digital control unit which is programmed in ALTERA's PLD, there has to be a Gold code generator and a delay line (flip-flops) to delay the I-code for the early/late correlator.

6.2 A/D-conversion of the correlator outputs

The DSP's have to process the signals from the output of the correlators. Therefore the signal has to be converted from an analog to a digital signal. This is done by the A/D-converters. But one A/D-converter for each correlator will unnecessarily drive up modem cost price so a multiplex system is proposed to save A/D-converters. After the A/D-converter a latch for each correlator is placed to allow asynchronic operation of the DSP. A part of the control unit has to be assigned to control the A/D-conversion, the multiplexer and the latches and to generate interrupt signals for the DSP indicating that data is ready.

6.3 Analog chip matched filter

Minimum BER is achieved when matched filter pairs are applied in the transmitter and in the receiver [7]. The pulse shaping filter in the transmitter limits the bandwidth of the transmitted pulses whereas the matched filter in the receiver limits the noise bandwidth and thus the noise power at the input of the receiver. This leads to a higher SNR at the input of the receiver. However, not every type of filter is suitable for these applications. Filters with a sharp cut-off frequency lead to larger ISI compared to filters with a more smoother roll-off characteristic. A trade off between bandwidth and ISI has to be made. An important issue is the shape of the received pulse $P_r(f)$ and $p_r(t)$. A pulse with a fast rate of decay and smaller values near $n.T_b$ is desirable since these properties will yield a system in which little timing errors will not cause large ISI. The commonly used characteristic of such a pulse is the raised cosine frequency characteristic and consists of a flat amplitude portion and a roll-off portion that has a sinusoidal form (formula 6.1).

$$P_r(f) = \begin{cases} T_b, & |f| \le \frac{r_b}{2} - \beta \\ T_b \cos^2 \frac{\pi}{4\beta}\left(|f| - \frac{r_b}{2} + \beta\right), & \frac{r_b}{2} - \beta < |f| \le \frac{r_b}{2} + \beta \\ 0, & |f| > \frac{r_b}{2} + \beta \end{cases} \tag{6.1}$$

where $0 < \beta < r_b/2$. When β is set to 0, the shape equals a rectangular characteristic and when β is set to $r_b/2$, the roll-off begins at $f = 0$ *Hz* and ends at r_b *Hz*.

According to [7] the transfer function of an optimal receiving filter yields

$$|H_R(f)|^2 = K\frac{|P_R(f)|}{|H_C(f)|} \tag{6.2}$$

when white noise and a flat amplitude characteristic of the transmitted pulse in the bandwidth of interest is assumed. K is a constant and $Hc(f)$ the channel transfer function which is not well known.

To describe the transfer function of the receiver, the transfer function of the channel must be known. This is very difficult because the channel characteristic of the CATV network is dependent of the location where the receiver is located. Furthermore, the transfer function of different CATV networks is always different. So in practice the receiver transfer function never equals the theoretical optimum.

A very often used raised cosine filter (RCF) is given in formula 6.3 and 6.4 [13].

$$H(f) = \frac{A}{2}\left\{1 - \sin\left(\frac{\pi}{2kf_m}(f - f_m)\right)\right\} \tag{6.3}$$

$$h(t) = A\frac{\sin(2\pi f_m t)}{2\pi f_m t} \cdot \frac{\cos(k2\pi f_m t)}{1 - \frac{8k^2 f_m t^2}{\pi}} \tag{6.4}$$

The values of *h(t)* are zero at multiples of the period time *T*. Thus, the RCF is ISI-free. However, the price of replacing an ideal filter by a RCF is the widening of the required bandwidth.

Looking back at figure 2.4 a digital RCF can be inserted between the A/D-converters and the correlators. An approximation of a digital RCF can be calculated by the software packet Digital Filter Design Package. When we use analog correlators, the filter preceding the correlator has to be analog too. In formula 6.3 the transfer function of a RCF is given but this transfer function can never be exactly realized since the filter has a finite number of poles, but an approximation of this filter can be made. Very often Butterworth filters with 2, 3 and 4 orders are used. *Bank* [13] suggests a new approximation of the RCF which is closer to the real RCF than the Butterworth filters for the case $k = 1$ and $A = 1$ in formula 6.2. The filter transfer function becomes

$$H(x) = \begin{cases} \frac{1}{2}\left(1 - \sin\frac{\pi}{2}(x-1)\right) & 0 \le x \le 2 \\ 0 & x > 2 \end{cases} \tag{6.5}$$

where $x = f / f_m$. This function is approximated by a function $F(x) = 1 / P_4(x)$ where $P_4(x)$ represents a polynomial of the 4th degree such as

$$\begin{cases} F(0) = H(0) = 1 \\ F'(0) = H'(0) = 0 \\ F(1) = H(1) = \frac{1}{2} \\ F'(x) \le 0 \text{ for } 0 \le x \le 2 \end{cases} \tag{6.5}$$

This conditions are satisfied by equation 6.6.

$$F(x) = \frac{1}{x^2 - \sqrt{2}x + 1} \cdot \frac{1}{x^2 + \sqrt{2}x + 1} = \frac{1}{1 + x^4} \tag{6.6}$$

This approximation is called a medium filter (MF). Figure 6.1 shows the transfer function of the ideal RCF and the MF.

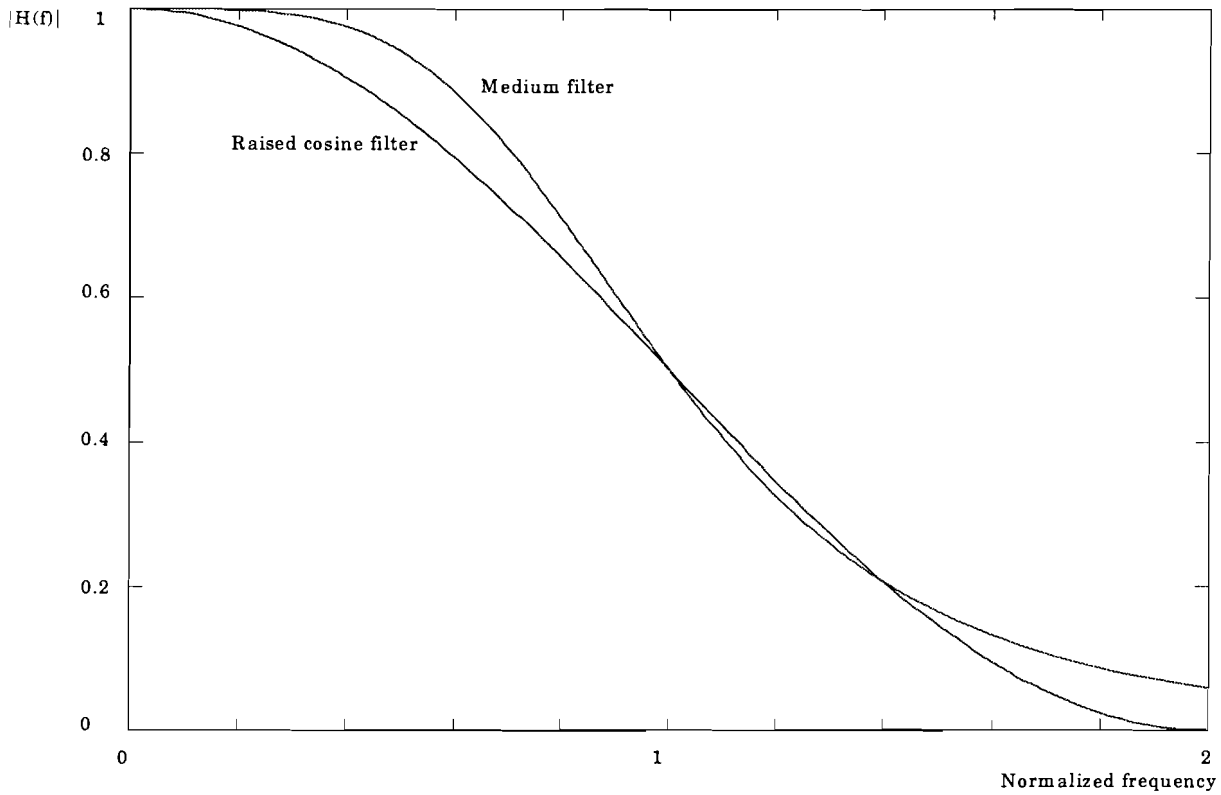


Figure 6.1: *Transfer function of a RCF and a MF. Normalized frequency 1 = ½r_b.*

To determine the MF performance, we have to compute some ISI values, i.e. the relative impulse response of the filters at *T, 2T, 3T....* with respect to the values at $t = T/2$.

The impulse response of the MF is given by equation 6.7 where x is replaced back by ω / ω_m.

$$h_{MF}(t) = \omega_m^4 \int_{-2\omega_m}^{2\omega_m} \frac{1}{\omega^4 + \omega_m^4} e^{j\omega t}\, d\omega \tag{6.7}$$

To calculate the ISI values, in equation 6.7 the variable t can be substituted by *T/2, T, 2T, 3T* etc. and h_{mf} can be calculated numerically and the ISI can be calculated using formula 6.8.

$$ISI_{MF}(k) = \left| \frac{h_m(t = kT)}{h_m(t = T/2)} \right| \tag{6.8}$$

In table 6.1 the ISI values of the MF are given for some different k as well as the ISI values of a second order and a fourth order Butterworth filter.

ISI / Filter type	t = T	t = 2T	t = 3T
RCF (ideal)	0	0	0
Butterworth n=2	0.1	0.03	0.05
Butterworth n=4	0.16	0.025	0.006
Medium filter	0.030	0.004	0.002

Table 6.1: *Computations of ISI for different filters*

Considering the ISI the medium filter has the best performance. Figure 6.2 depicts the schematic diagram of the filter where $f_m = 1 / (2\pi RC)$.

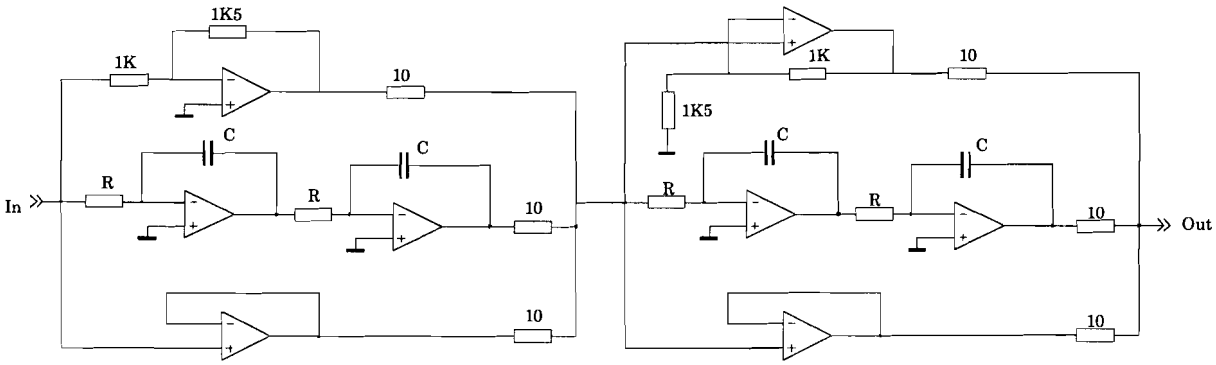


Figure 6.2: *Schematic diagram of the medium filter circuit.*

This circuit is simulated with Spice and the result is plotted in figure 6.3.

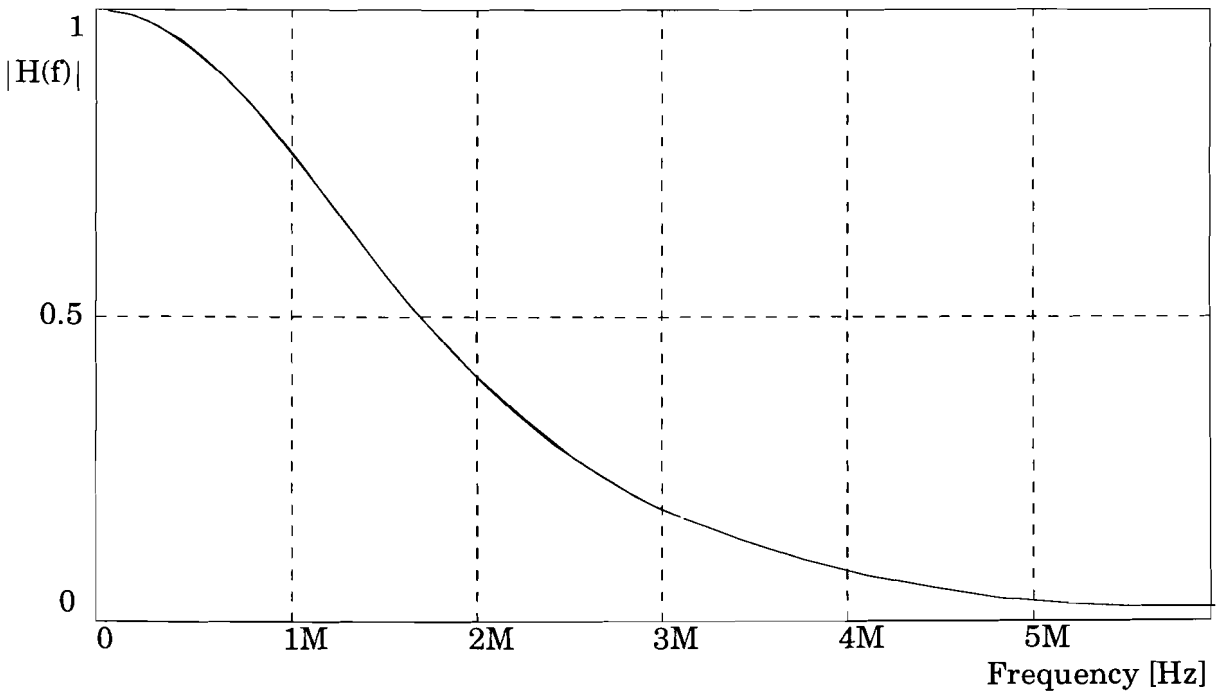


Figure 6.3: *Spice simulation of the RCF circuit (f_m = 1.5 MHz).*

More about realization of raised cosine filters can be found in [14,15,16].

6.4 Circuit description

The entire receiver circuit is depicted in figure 6.4.

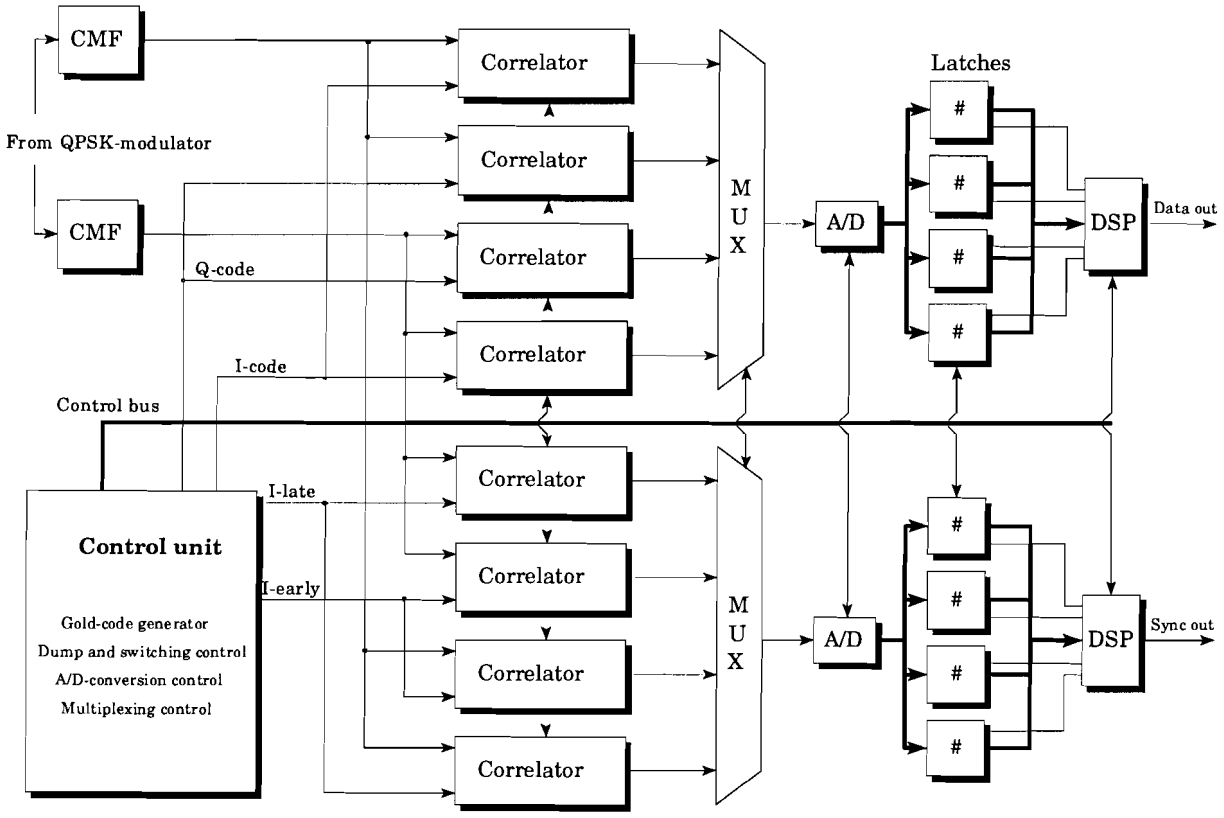


Figure 6.4: *Circuit diagram of the receiver.*

If we compare figure 6.4 with figure 2.4 (the digital receiver) we observe that the circuit structure hasn't changed. A remarkable different is the position of the A/D-converter and an analog multiplexer to share the A/D-converter by four correlators. It's also possible to use one A/D-converter for eight correlators or one A/D converter for each correlator. Also the signal delay units are disappeared.

The control unit is all digital and can be programmed in a PLD. The control unit contains:

1. Goldcode generators and a delay circuit to provide an early and late code sequence.
2. Dump and switch logic for the I&D-filters.
3. Control logic to trigger the A/D-converter.
4. Control logic to drive the multiplexer.
5. Control logic to clock the latches when an A/D-conversion is finished.
6. Control logic to generate an interrupt pulse for the DSP indicating valid data is present at its input.

Recent research has turned out that each user can operate with one Gold code for both the I and Q channel. Therefore the software in the DSP has to be changed. Consequence of this is that each user needs one Gold code generator but also two out of the four

correlators for the data recovery can be dropped because the first and the second correlator and the third and the fourth correlator are fed by the same input signals if the I and Q codes are the same. This also means a reduction of the hardware complexity.

7. Conclusion and recommendations

In this report the realization of an analog correlator for a high speed CDMA modem is discussed as well as some theoretical simulations of BER's and filter types. The correlator was realized with discrete hardware components and measurements at several chiprates were done.

The measurements of the auto correlation functions were in good agreement with the theoretical values of the auto correlation functions. Therefore it can be concluded that the realized correlator operates well also at higher frequencies. So it can be concluded that an analog correlator can speed up the conventional digital modem. Measurements turned out that a factor of at least ten (50 Mchips/s) can be achieved.

The measurements of the analog correlator in a multi-user system, where an arbitrary waveform generator generates a CDMA-signal containing an arbitrary number of users, have turned out that a CDMA-signal containing 80 users can be correlated error free. A higher amount of users leaded to bit errors but a better synchronization must improve these results. These measurements where done at a chiprate of 25 Mchips/s and this corresponds to a bitrate of approximately 400 kbit/s using a QPSK modulation scheme.

When the modem is equipped with an analog correlator the modem will require a lot of discrete analog hardware since the modem consists of eight correlators. Furthermore, each individual receiver needs its own A/D-converters so the cost price will increase. However, recent research has resulted in the possibility of using one code sequence for each user by adapting the algorithm of the complex phase rotator. This implies that the modem needs less correlators so in this case the hardware complexity reduces and makes the analog correlator more attractive.

References

[1] R.P.C. Wolters
Bi-directional transmission in hybrid fibre-coax CATV-networks; a framework.
Technical report, Eindhoven University of Technology, 1995.

[2] J.K. Holmes
Coherent spread spectrum systems.
John Wiley & Sons, New York, 1982.

[3] M.K. Simon and J.K. Omura and R.A. Scholtz and B.K. Levitt
Spread spectrum communications volume III.
Computer Science Press, Rockville, Maryland 20850.

[4] A.W.L. Janssen
Realization of a CDMA based modem for upstream access in CATV networks.
Technical report, Eindhoven University of Technology, 1996.

[5] Y.L.C. de Jong
A CDMA based bidirectional system for CATV networks.
Technical report, Eindhoven University of Technology, 1996.

[6] W. van Etten
Stochastische Signaaltheorie, 12e editie januari 1996.
Collegedictaat no. 5632 of EUT.

[7] K.S. Shanmugam
Digital and analog communication systems
John Wiley & Sons, New York, 1985.

[8] A.M. Street and D.J. Edwards
Analysis of an analog correlator for fibre optic sensor multiplexing applications.
IEE colloquium on progress in fibre optic sensors & their applications.
No. 194 (1995) p. 6/1 - 6/6.

[9] P.C.T van der Laan et al.
Electromagnetic compatibility, november 1994.
Collegedictaat no. 5785 of EUT.

[10] C.R. Giles and J.M. Kahn
1 Gbit/s integrate and dump filter for digital communication systems.
Electronic letters, Vol. 25 (1989), No. 3, p. 212-214.

[11] B. Enning and E. Bodtker and G. Jacobsen and B. Jensen
Design and test of novel integrate and dump filter for optical Gbit/s system application
Electronic letters, Vol. 27 (1991), No. 24, p. 2286-2288.

[12] Max 2 Plus Users Guide.
Altera

[13] M. Bank and J. Gavan
Pratical realization of a raised-cosine filter
Electronic letters, Vol. 32 (1996), No. 5, p. 438-440.

[14] L.F. Lind
On the approximation of raised-cosine filters
European Conference on circuit theory and design, sept 1-4 1987, Paris,
p. 657 - 662.

[15] L.F. Lind and T. Labarre
Realizable 100% raised cosine filters
Electronic letters, Vol. 17 (1981), No. 5, p. 197-198

[16] S.E. Nader and L.F. Lind
Optimal data transmission filters
IEEE transactions on circuits and systems, Vol. 26 (1979), No. 1, p. 37-45

[17] P. Sommen et al.
Realisering van digitale signaalberwerkende systemen
Collegedictaat no. 5750 of EUT.

[18] R.A Iltis and L. Mailaender
Multiuser detection of quasisynchronous CDMA signals using linear
decorrelators.
IEEE transactions on communications, Vol. 44 (1996), No. 11, p. 1561-1570.

[19] D.E. Johnson and J.R. Johnson and H.P. Moore
A handbook of active filters
Prentice-Hall, Englewood Cliffs, New Jersey 1980.

[20] M.C. Jeruchim and P. Balaban and K.S. Shanmugam
Simulation of communication systems.
Plenum Press, New York and London, 1992.

[21] F. Tarico
Fast bipolar dump switches has low offset
EDN Magazine, november 1974. p. 66.

Appendix A: Simulation program sources

A.1 Simulation of LPF and I&D-filter

To get an impression of the correlator output using a LPF or an I&D-filter, a C-program source was written. The simulation program consists of three main programs: one transmitter program which generates an output file containing the output signal values of 128 users (one code each) and two receiver programs which detect the data of one particular user with a LPF and an I&D-filter respectively. In this simulation perfect synchronization of the codes is assumed and the received signal is free of noise and modulation.

The data for all users was generated by Matlab's command RAND (random) and stored in the file USERDAT.DAT. This file contains the data of all successive users (-1's and 1's). Each user transmits 256 data symbols. The Gold codes are generated by the program itself. This is done by modulo-2 adding of the outputs of two shiftregisters. These shiftregisters get their new input by a feedback of some modulo-2 adders tapped somewhere from the shiftregisters. For detailed information about the Gold code generator read [2,4,5] or see the circuit in appendix B.

The filter of the correlator is calculated by Matlab's command BUTTER. With the command [a,b,c,d] = butter(5,1/127) a state space matrix of a five pole Butterworth filter with cut-off frequency 1/127 returns to a,b,c and d. The receiver program calculates the output by an IIR-filter with the state space matrices [17]. Here an IIR-filter is used instead of a FIR-filter because the FIR-filter requires a huge amount of taps and thus long simulation run times. However we must beware of the fact that IIR-filters never have a linear phase transfer function which can lead to different group delays.

The second receiver program is much easier. Here an adder is used instead of a filter and simulates an I&D-filter.

The first 12 symbols and thus 12*127 output values of both receivers are used in figure 4.4. In this figure, all detected bits are error free but the remaining simulation results in some errors at the LPF output.

```
/* Test correlator with LPF and I&D-filter          */
/* Transmitter Program                              */
/* Written by Jim van der Heijden                   */

#include <stdio.h>

#define NUSERS        127
#define DATALEN       256
#define SEQLEN        127

int i,n,l,k;
int data[NUSERS][DATALEN];
int code[NUSERS][SEQLEN];
int x,y,reg_1,reg_2,seq_out,a;
int out;
float input;
FILE *fdata,*fout;

void main(){

 if ((fdata=fopen("userdata.dat","r"))==NULL)
    printf("Error opening file\n");
 if ((fout=fopen("out.dat","w"))==NULL)
    printf("Error opening file\n");

 for (i=0;i<=NUSERS-1;i++){                          /* Read userdata */
   for (n=0;n<=DATALEN-1;n++){
     fscanf(fdata,"%f",&input);
     data[i][n]=(int) input;
   }
 }

/*************************************************************************/
/* Generate all code sequences for transmitting users                    */
/*************************************************************************/

 for (a = 0; a<=NUSERS-1; a++){
    reg_1 = a;                                       /* init shiftreg 1 */
    reg_2 = 127;                                     /* init shiftreg 2 */
    for (i = 0; i<=126;i++){
       x = ((reg_1 & 64)>>6)^((reg_1 & 8)>>3);
       y = ((reg_2 & 64)>>6)^((reg_2 & 32)>>5)^((reg_2 & 16)>>4)
          ^((reg_2 & 8)>>3)^((reg_2 & 4)>>2)^((reg_2 & 2)>>1);
       reg_1 = reg_1 << 1;
       reg_2 = reg_2 << 1;
       seq_out = ((reg_1 & 128)^(reg_2 & 128))>>7;
       reg_1 = ((reg_1 | x) & 255);
       reg_2 = ((reg_2 | y) & 255);
       if (seq_out == 0) seq_out = -1;               /* Make a bipolar signal */
       code[a][i]=seq_out;
    }
  }

/*************************************************************************/
/* Start transmitting                                                    */
/*************************************************************************/

 for (l=0;l<=DATALEN-1;l++){
   for (n=0;n<=SEQLEN-1;n++){
     out =0;
     for (i=0;i<=NUSERS-1;i++){
       out += data[i][l]*code[i][n];                 /* calculate output */
     }
     fprintf(fout,"%d\n",out);                       /* Output data to output file */
   }
 }

fclose(fdata);
fclose(fout);
}
```

```
/* Test correlator with LPF (IIR)  5 pole butterworth f0=1/127      */
/* Receiver program                                                 */
/* Written by Jim van der Heijden                                   */

#include <stdio.h>

#define NUSERS        127
#define USER          0                    /* Receive only user 0 */
#define DATALEN       256
#define SEQLEN 127

int inp;
int error;
int chip;
int i,n,k,p;
int x,y,reg_1,reg_2,seq_out;
long int l;
int data_s[DATALEN];
int code[SEQLEN];
int rec[DATALEN];
float input;
float out;
float a[5][5];
float b[5];
float x0[5],x1[5];

FILE *fdata,*fout,*fout2;

void main(){

 if ((fdata=fopen("userdata.dat","r"))==NULL)
    printf("Error opening file\n");
 if ((fout=fopen("out.dat","r"))==NULL)
    printf("Error opening file\n");
 if ((fout2=fopen("out2.dat","w"))==NULL)
    printf("Error opening file\n");

/*************************************************************************/
/* Define filter parameters (Matlab)                                     */
/*************************************************************************/

 a[0][0]=0.9742;
 a[0][1]=0;
 a[0][2]=0;
 a[0][3]=0;
 a[0][4]=0;
 a[1][0]=0.0253;
 a[1][1]=0.9582;
 a[1][2]=-0.0256;
 a[1][3]=0;
 a[1][4]=0;
 a[2][0]=0.0003;
 a[2][1]=0.0256;
 a[2][2]=0.9997;
 a[2][3]=0;
 a[2][4]=0;
 a[3][0]=0.0000;
 a[3][1]=0.0003;
 a[3][2]=0.0260;
 a[3][3]=0.9836;
 a[3][4]=-0.0260;
 a[4][0]=0;
 a[4][1]=0;
 a[4][2]=0.0003;
 a[4][3]=0.0260;
 a[4][4]=0.9997;

 b[0]=0.0365;
 b[1]=0.0005;
 b[2]=0.0000;
 b[3]=0;
 b[4]=0;

/*************************************************************************/
/* Generate all code sequences for receiving users                       */
/*************************************************************************/

    reg_1 = USER;                                  /* init shiftreg 1 */
    reg_2 = 127;                                   /* init shiftreg 2 */
```

```
    for (i = 0; i<=126;i++){
       x = ((reg_1 & 64)>>6)^((reg_1 & 8)>>3);
       y = ((reg_2 & 64)>>6)^((reg_2 & 32)>>5)^((reg_2 & 16)>>4)
          ^((reg_2 & 8)>>3)^((reg_2 & 4)>>2)^((reg_2 & 2)>>1);
       reg_1 = reg_1 << 1;
       reg_2 = reg_2 << 1;
       seq_out = ((reg_1 & 128)^(reg_2 & 128))>>7;
       reg_1 = ((reg_1 | x) & 255);
       reg_2 = ((reg_2 | y) & 255);
       if (seq_out == 0) seq_out = -1;
       code[i]=seq_out;
     }

/*****************************************************************************/
/* Load userdata to compare with detected symbols                            */
/*****************************************************************************/

for (i=0;i<=NUSERS-1;i++){
   for (n=0;n<=DATALEN-1;n++){
     fscanf(fdata,"%f",&input);
     if (i==USER) data_s[n]=(int) input;
   }
 }

/*****************************************************************************/
/* Start receiver                                                            */
/*****************************************************************************/
 chip=0;
 n=-1;
 out=0;

 for (l=0;l<=SEQLEN*DATALEN-1;l++){
   fscanf (fout,"%d",&inp);
   inp = inp*code[chip];

   out = x0[2]*0.0001 + x0[3]*0.0092 + x0[4]*0.7070;       /* filter output */

   x1[0]=0;
   x1[1]=0;
   x1[2]=0;
   x1[3]=0;
   x1[4]=0;
   for (i=0;i<=4;i++){                                   /* Calculate new states */
     for (k=0;k<=4;k++){
     x1[i] +=a[i][k]*x0[k];
     }
     x1[i] +=b[i]*inp;
   }
   for (i=0;i<=4;i++) x0[i]=x1[i];                       /* copy new states in next states */

   chip++;
   if (chip == 127) {                                    /* End of sequence ? */
     chip =0;
   }
   if (chip==80){                        /* Delay of 80 chips compensates for filter delay */
     if (out>0) rec[n]=1;
     if (out<0) rec[n]=-1;
     n++;
   }
   fprintf(fout2,"%f\n",out*150);                        /* output to file and scale    */
 }

 error=0;
 for (i=0;i<=DATALEN-1;i++){
   if (data_s[i] != rec[i]) error++;
 }
 printf ("Aantal fouten: %d \n",error);

fclose(fdata);
fclose(fout);
fclose(fout2);
}
```

```
/* Test correlator with I&D-filter                                    */
/* Receiver program                                                   */
/* Written by Jim van der Heijden                                     */

#include <stdio.h>

#define NUSERS        127
#define USER          0
#define DATALEN       256
#define SEQLEN 127

int i,n,k,p;
int x,y,reg_1,reg_2,seq_out;
long int l;
int code[SEQLEN];
float input;
int out;
int inp;
int chip;

FILE *fout,*fout2;

void main(){

 if ((fout=fopen("out.dat","r"))==NULL)
    printf("Error opening file\n");
 if ((fout2=fopen("out3.dat","w"))==NULL)
    printf("Error opening file\n");

/*************************************************************************/
/* Generate all code sequences for receiving users                       */
/*************************************************************************/

    reg_1 = USER;                                        /* init shiftreg 1 */
    reg_2 = 127;                                         /* init shiftreg 2 */
    for (i = 0; i<=126;i++){
       x = ((reg_1 & 64)>>6)^((reg_1 & 8)>>3);
       y = ((reg_2 & 64)>>6)^((reg_2 & 32)>>5)^((reg_2 & 16)>>4)
          ^((reg_2 & 8)>>3)^((reg_2 & 4)>>2)^((reg_2 & 2)>>1);
       reg_1 = reg_1 << 1;
       reg_2 = reg_2 << 1;
       seq_out = ((reg_1 & 128)^(reg_2 & 128))>>7;
       reg_1 = ((reg_1 | x) & 255);
       reg_2 = ((reg_2 | y) & 255);
       if (seq_out == 0) seq_out = -1;
       code[i]=seq_out;
     }

/*************************************************************************/
/* Start receiver                                                        */
/*************************************************************************/

 chip=0;
 out=0;

 for (l=0;l<=SEQLEN*DATALEN-1;l++){
   fscanf (fout,"%d",&inp);
   inp = inp*code[chip];
   out +=inp;
   chip++;

   if (chip==127}{                          /* end of sequence */
     chip =0;
     out =0;
   }
    fprintf(fout2,"%d\n",out);
 }
fclose(fout);
fclose(fout2);
}
```

A.2 Simulation of imperfect integration

To determine the effect of imperfect integration, a simulation program was written. The simulation consists of two program sources: a transmitter program which generates an output file and a receiver program which adds noise to the received signal and counts the errors of the received symbols.

In this simulation each user uses two codes and the maximum number of users is 64. Each user transmits 2 * 256 symbols which are QPSK-modulated on a carrier. In this simulation carrier and code synchronization are assumed and no filters are considered because of long simulation run times.

A noise file with variance 1 (the noise power of this file is normalized to 1) and average 0 is generated with Matlab. In the source code of the receiver, the noise is multiplied by a constant which depends on the E_b/N_0 ratio. The receiver program receives all users for different amounts of integrated chips and outputs the number of errors. From this the BER's are calculated and a least square fit method is used to obtain smooth BER characteristics. The results are plotted in figure 4.7.

```
/*********************************************************/
/*  Transmitter with QPSK modulation without CMF         */
/*  Written by J. v.d. Heijden   jan 1998                */
/*********************************************************/


#include <stdio.h>
#include <math.h>

#define DATALEN       256           /* Length of data sequence    */
#define SEQLEN        127           /* Lenght of one sequence     */
#define SAMPLESPCHIP  4             /* Number of samples per chip */
#define NUSERS        63            /* Number of active users     */
#define IF            4.1E+6        /* IF modulation frequency    */
#define FSAMPLE       1.6256e+7     /* Simulation sample frequency*/

void main()
{
  int userdata[2*NUSERS][DATALEN];
  int code[NUSERS*2][SEQLEN];
  float theta,tx_out;
  float output_i, output_q;
  float tx_output_i,tx_output_q;
  int reg_1,reg_2,seq_out,a,x,y;
  int i, j, k, l, n,p;
  FILE *fptr_out, *fptr_dat;

 if((fptr_out = fopen("out_if_63.dat", "w")) == NULL) {
    printf("Error in opening file 'out_if_63.dat'.\n");
  }
 if((fptr_dat = fopen("userdata.dat", "r")) == NULL) {
    printf("Error in opening file 'userdata.dat'.\n");
  }

/*************************************************************************/
/*      Goldcode generation routine                                      */
/*************************************************************************/

  for (a = 0; a<=NUSERS*2-1; a++){
    reg_1 = a;                                  /* init shiftreg 1 */
    reg_2 = 127;                                /* init shiftreg 2 */
    for (i = 0; i<=126;i++){
      x = ((reg_1 & 64)>>6)^((reg_1 & 8)>>3);
      y = ((reg_2 & 64)>>6)^((reg_2 & 32)>>5)^((reg_2 & 16)>>4)
         ^((reg_2 & 8)>>3)^((reg_2 & 4)>>2)^((reg_2 & 2)>>1);
          reg_1 = reg_1 << 1;
          reg_2 = reg_2 << 1;
          seq_out = ((reg_1 & 128)^(reg_2 & 128))>>7;
          reg_1 = ((reg_1 | x) & 255);
          reg_2 = ((reg_2 | y) & 255);
          if (seq_out == 0) seq_out = -1;
          code[a][i] = seq_out;
    }
  }

/*************************************************************************/
/*  Get userdata from file                                               */
/*************************************************************************/

  for (i=0;i<=2*NUSERS-1;i++){
    for (k=0;k<=DATALEN-1;k++){
      fscanf(fptr_dat, "%d",&userdata[i][k]);
    }
  }

/*************************************************************************/
/* Main program  transmitter                                             */
/*************************************************************************/

  theta =0.0;
  for(k = 0; k <= DATALEN-1; k++) {
    printf("Current symbol %d...\n", k);
    for(n = 0; n <= SEQLEN-1; n++) {
      for(l=0;l<=SAMPLESPCHIP-1;l++){
        tx_output_i = 0.0;
        tx_output_q = 0.0;
          for(i = 0; i <= NUSERS-1; i++) {
            output_i=(float) (userdata[2*i][k]*code[2*i][n]);
            output_q=(float) (userdata[(2*i)+1][k]*code[(2*i)+1][n]);
            tx_output_i += output_i;                    /* add all users */
```

```
            tx_output_q += output_q;
          }
        tx_out = tx_output_i*cos(theta) + tx_output_q*sin(theta);
        theta +=(2*PI*IF)/FSAMPLE;
         fprintf(fptr_out, "%f\n",tx_out);
      }
    } /* end of sequence loop */
  } /* end of databit loop */

  fclose(fptr_out);
  fclose(fptr_dat);
}
```

```
/*****************************************************************************/
/*  Receiver with demodulation without Chip matched filter                   */
/*  Simulation of imperfect correlation                                      */
/*  Written by J. v.d. Heijden  jan 1998                                     */
/*****************************************************************************/

#include <stdio.h>
#include <math.h>

#define NUSERS        63        /* number of active users              */
#define SEQLEN        127       /* code sequence length                */
#define DATALEN       256       /* number of symbols to be processed   */
#define IF            4.1E6     /* IF frequency                        */
#define FSAMPLE       16.256E6  /* Simulation sample frequency         */

void main(){

  int code[2*NUSERS][SEQLEN];                 /* code sequences                  */
  int USER;
  int l,p;
  int i;
  int chip = 0;                                /* chip counter                    */
  int reg_1;
  int reg_2;
  int seq_out;
  int a,x,y;
  int error;
  int noc;                                  /* Number of chips to be integrated */
  float data_i[NUSERS][DATALEN];            /* transmitted data                 */
  float data_q[NUSERS][DATALEN];
  float sample;                              /* received samples                */
  float sym_i[NUSERS];                      /* detected symbols                 */
  float sym_q[NUSERS];
  float corr_inp_i;
  float corr_inp_q;
  float noise;
  float data;
  float theta;
  float tot_i,tot_q;
  float flt_i[10],flt_q[10];
  FILE *fptr_i, *fptr_q;                     /* file pointers                   */
  FILE *fptr_sym, *fptr_noise, *fptr_rep;

  /*****************************************************************************/
  /* open files for input and output data                                      */
  /*****************************************************************************/

  if((fptr_sym = fopen("userdata.dat", "r")) == NULL) {
    printf("Error in opening file 'userdata.dat'.\n");
  }
  if((fptr_rep = fopen("report63.dat", "w")) == NULL) {
    printf("Error in opening file 'report.dat'.\n");
  }

/*****************************************************************************/
/*  Get transmitted userdata                                                 */
/*****************************************************************************/

  for(USER=0;USER<=NUSERS-1;USER++){
    for(i=0;i<=DATALEN-1;i++) fscanf(fptr_sym,"%f",&data_i[USER][i]);
    for(i=0;i<=DATALEN-1;i++) fscanf(fptr_sym,"%f",&data_q[USER][i]);
  }
  fclose(fptr_sym);

/*****************************************************************************/
/* Generate all code sequences for receiving users                           */
/*****************************************************************************/

 for (a = 0; a<=2*NUSERS-1; a++){
    reg_1 = a;                                          /* init shiftreg 1 */
    reg_2 = 127;                                        /* init shiftreg 2 */
    for (i = 0; i<=126;i++){                            /* calculate input functies */
       x = ((reg_1 & 64)>>6)^((reg_1 & 8)>>3);
       y = ((reg_2 & 64)>>6)^((reg_2 & 32)>>5)^((reg_2 & 16)>>4)
          ^((reg_2 & 8)>>3)^((reg_2 & 4)>>2)^((reg_2 & 2)>>1);
       reg_1 = reg_1 << 1;                               /* shift registers 1 position */
       reg_2 = reg_2 << 1;
       seq_out = ((reg_1 & 128)^(reg_2 & 128))>>7;       /* modulo 2 sum of 2 registers */
       reg_1 = ((reg_1 | x) & 255);                          /* bit 1 gets its new value */
       reg_2 = ((reg_2 | y) & 255);
```

```
        if (seq_out == 0) seq_out = -1;
        code[a][i]=seq_out;
      }
    }

/*****************************************************************************/
/* start reception                                                           */
/*****************************************************************************/

  for(noc=126;noc>=64;noc--){                    /* simulate 127 to 65 integration chips */
    printf("processing %d \n",noc);
    if((fptr_noise = fopen("noise.dat", "r")) == NULL) {
      printf("Error in opening file 'noise.dat'.\n");
    }
    if((fptr_i = fopen("out_if_63.dat", "r")) == NULL) {        /* received signal   */
      printf("Error occurred while opening file for I symbol output.\n");
    }

    theta=0.0;
    error=0;
    i=0;
    chip=0;
    for(p=0;p<=DATALEN-1;p++){
      for(l = 0; l <= (SEQLEN * 4) - 1; l++) {
      i++;

      /*************************************************************************/
      /* load baseband input signal                                            */
      /*************************************************************************/

      fscanf(fptr_i, "%f", &sample);
      fscanf(fptr_noise, "%f", &noise);
      sample +=2.51*noise;                    /*  add noise   Eb/N0 = 5 dB  */

      corr_inp_i =sample*cos(theta);
      corr_inp_q =sample*sin(theta);
      theta += (2 * PI * IF)/FSAMPLE;

      /*************************************************************************/
      /* Correlator                                                            */
      /*************************************************************************/

      if(chip <=noc) {                       /* simulate imperfect integration */
       for(USER=0;USER<=NUSERS-1;USER++){
          sym_i[USER] += corr_inp_i * code[2*USER][chip];
         sym_q[USER] += corr_inp_q * code[2*USER+1][chip];
        }
      }
      if (i==4) {
       chip++;                                        /* increment chip counter   */
       i=0;
       if(chip == SEQLEN) {                           /* end of code sequence     */
         chip = 0;                                    /* reset chip counter       */

         /***********************************************************************/
         /* Count number of errors                                              */
         /***********************************************************************/

         for(USER=0;USER<=NUSERS-1;USER++){
           if ((sym_i[USER]<=0)&&(data_i[USER][p]>0)) error++;
           if ((sym_i[USER]>=0)&&(data_i[USER][p]<0)) error++;
           if ((sym_q[USER]<=0)&&(data_q[USER][p]>0)) error++;
           if ((sym_q[USER]>=0)&&(data_q[USER][p]<0)) error++;
            sym_i[USER] = 0.0;
           sym_q[USER] = 0.0;
         }
       }
      }
    }
  }

  fprintf(fptr_rep,"%d \n",error);          /* output number of errors to report file */
  printf("fout %d \n",error);
  fclose(fptr_i);
  fclose(fptr_noise);
  }

  fclose(fptr_rep);
}
```

Appendix B: Circuits

B.1 Digital control circuit

The control logic to control the analog correlator and the Gold-code generators are housed in one programmable logic device of ALTERA. In table 4.1 the activities are listed as well as the time instants. Also a delay circuit is developed to obtain phase shifts between two Gold-code sequences. Figure B.1 shows the schematic diagram of the entire control circuit for measurement purposes.

From figure B.1 it becomes clear that one Gold code generator gets its reset from the external master reset whereas a second Gold code generator gets its reset from another circuit. A comparator compares the counter output (which counts the clock pulses) with an external forced delay value. When these values are equal, the second Gold code generator will be resetted. Also the comparator will be disabled because this procedure must happen only once after a master reset. Another counter counts from 1 to 253. The output of this counter corresponds with the chip number of the two successive sequences. A combination of logical gates determine the output signals like the switch pulses and the dump pulses according to table 4.1.

Obviously the control logic for the CDMA-modem will be slight different. The delay circuit for instance exists only for test purposes and will be eliminated in the control logic for the modem.

Appendix C contains the simulations of the digital control circuit.

B.2 Entire correlator test circuit

Figure B.2 shows the entire test circuit of the analog correlator. The A/D-converter is not included yet but the control signals for the A/D-converter are present already. There are two types of control signals for the A/D-converter: a clock signal and a short pulse indicating the start of a conversion. Depending on what type of A/D-converter is used one of the two signals can be used.

Considering figure B.2 the integrate and dump circuits seem to be the most complex parts of the entire circuit. The mixer used is an AD835 of Analog Devices and is a four quadrant multiplier and operates at frequencies up to 250 MHz. The mixer gets its two bipolar sequences from the PLD (EPF8636 of ALTERA). The input of the I&D circuit can be switched between the analog mixer and a digital mixed signal by a jumper. The I&D circuit consists of two identical I&D sub-circuits. Two anti-parallel connected transistors act as a switch for bipolar signals. The opamps used are MAX477's of Maxim. The specified maximum power supply voltage amounts +/- 5 volts but a slightly lower supply voltage gives much better performance considering unwanted oscillations. On the other hand, low power supply voltages induce a smaller dynamic range.

Connector J1 is used for programming the PLD, connector J3 is the clock input, switch sw4 is the master reset, sw2 and sw3 are the Gold code initial registers and sw1 corresponds with the delay value of the second Gold code generator.

The output signals of the PLD are uni-polar (0 and 5 volts) and have to be converted to bipolar signals. Therefore a negative reference voltage source is applied. With resistors (R13-R15 and R17-R19) the output of the PLD can be converted to a bipolar signal. A very stringent condition is that the bipolar signal is symmetrical, otherwise great errors in the correlation can be expected. On one hand these resistors must be kept small because of fast rise and fall times, on the other hand they may not be too small because of maximum currents (20 mA per output) supplied by the PLD. Concluded: small rise- and fall times lead to a greater power dissipation. A trade off has to made between power dissipation and small rise times. Here 240 ohm was chosen. The reference source can be tuned to optimize the symmetry of the bipolar signal.

The correlation values were measured right after the I&D circuit (pin 15 of IC3). For the BER measurements a zero-detector and a flip-flop are connected to the output of the I&D circuit. The output of the comparator becomes -5 volts if the input is positive and +5 volts if the input is negative. This inversion compensates the inversion of the I&D-circuit. The flip-flop samples the output on the right instant; just after the end of a sequence.

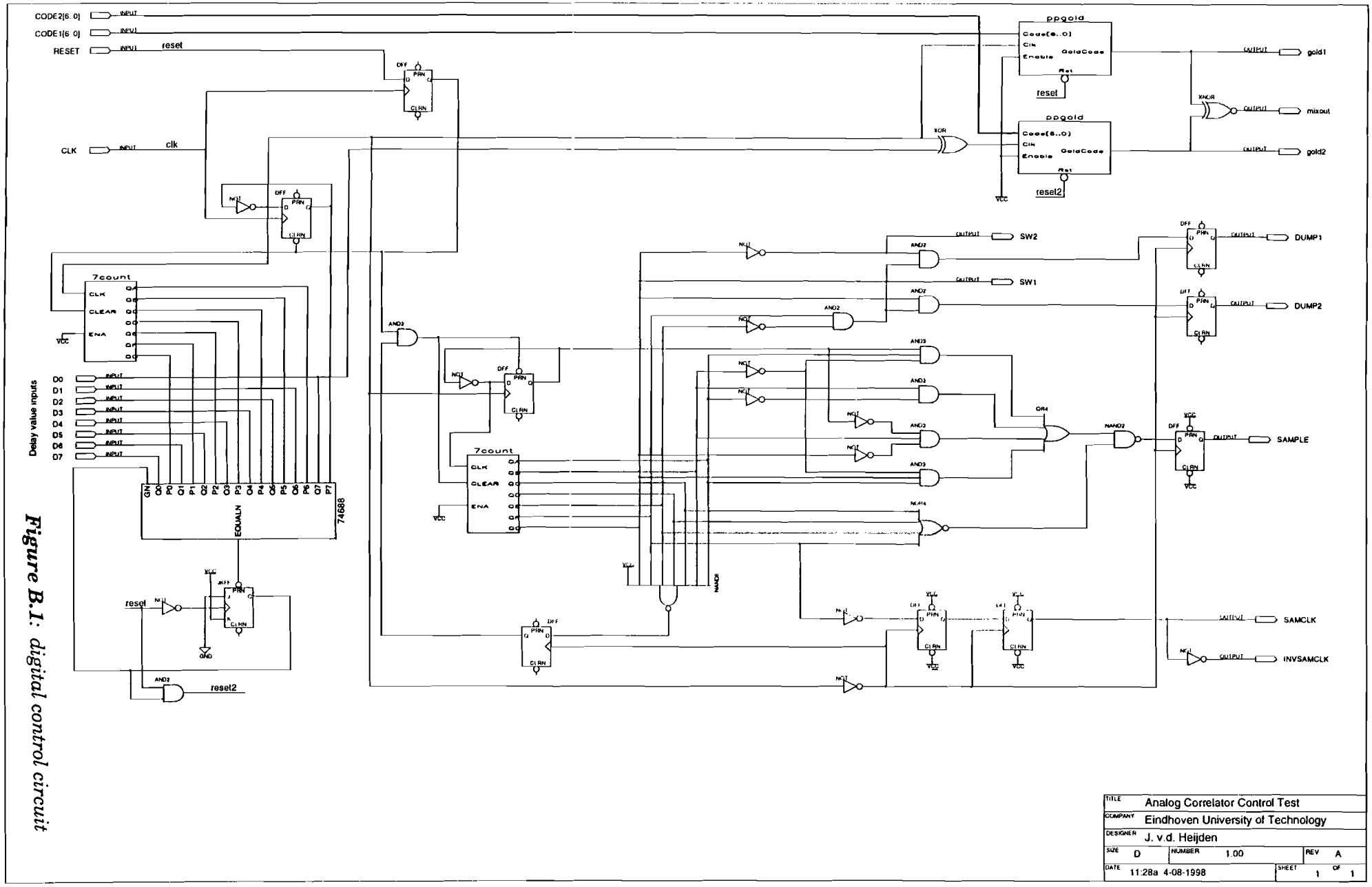


Figure B.1: *digital control circuit*

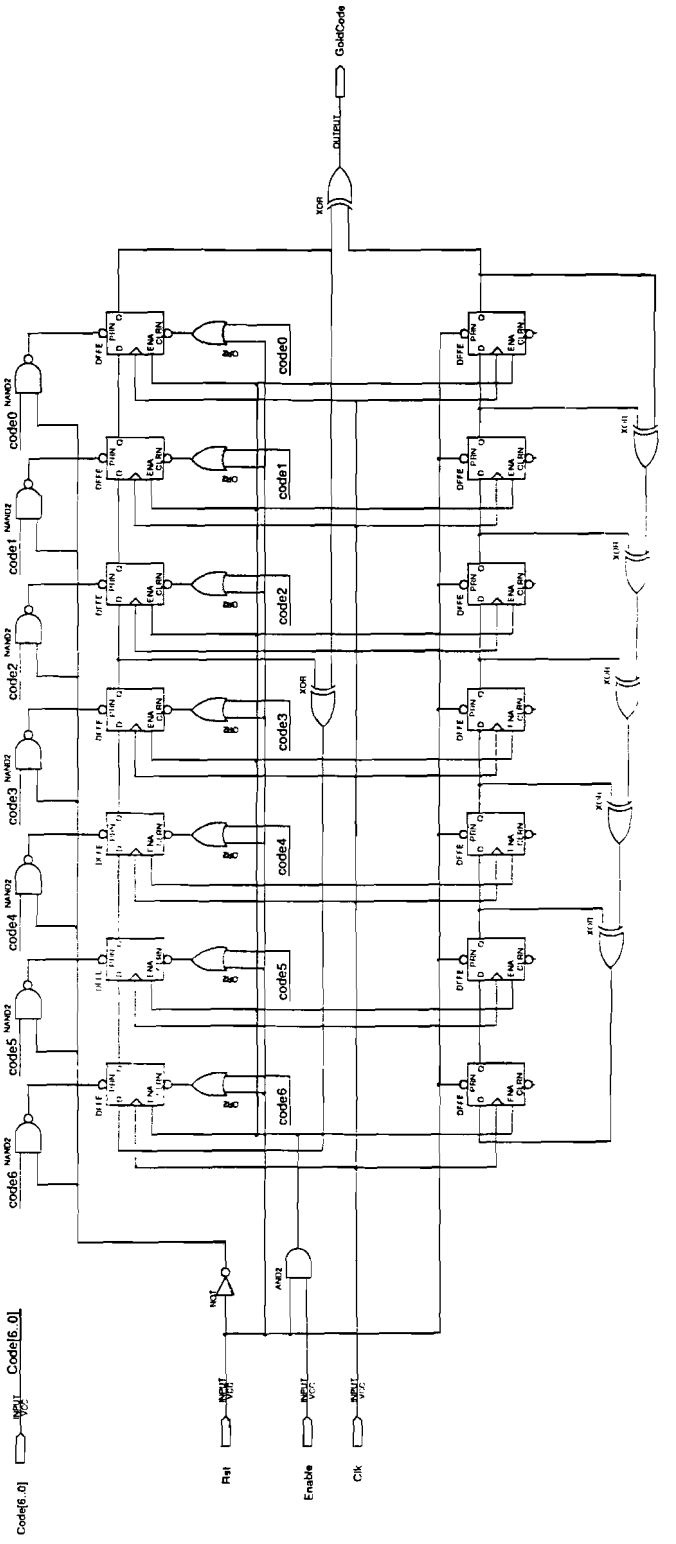


Figure B.2: *Gold-code generator (subcircuit)*

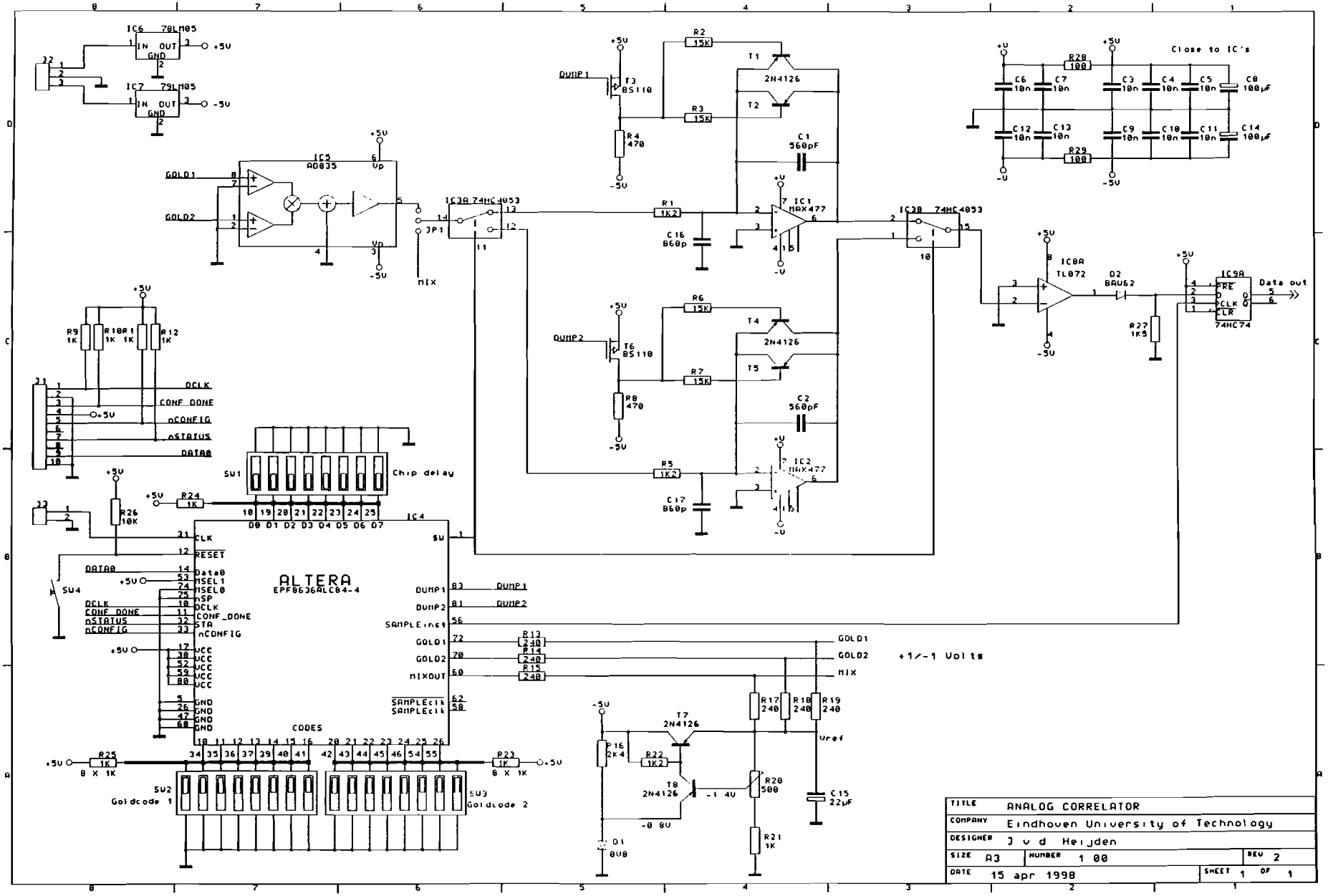
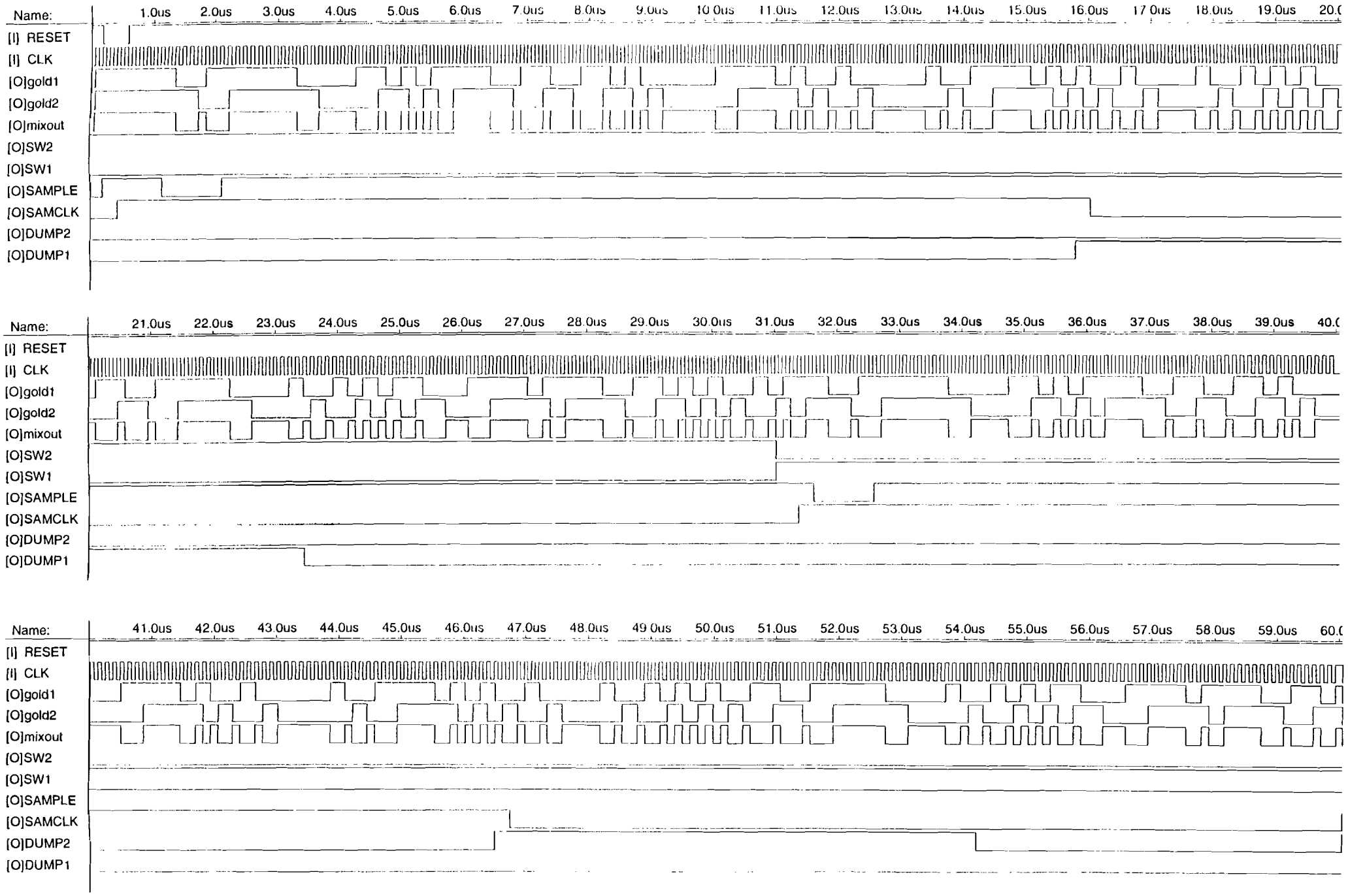


***Figure B.3**: Total circuit of the analog correlator*

Appendix C: Simulation of control circuit

This simulation is done by the software packet *"Max2Plus"*. The delay is set to "3" so a delay of 1.5 chiptime is configured. This can be seen in the simulation. The clock frequency is twice the chiprate. All output signals function as was proposed in table 5.1 so the digital hardware function well according to this simulation. The timescale of this simulation is arbitrary.


Name:
1.0us 2.0us 3.0us 4.0us 5.0us 6.0us 7.0us 8.0us 9.0us 10.0us 11.0us 12.0us 13.0us 14.0us 15.0us 16.0us 17.0us 18.0us 19.0us
[I] RESET
[I] CLK
[O]gold1
[O]gold2
[O]mixout
[O]SW2
[O]SW1
[O]SAMPLE
[O]SAMCLK
[O]DUMP2
[O]DUMP1
Name:
21.0us 22.0us 23.0us 24.0us 25.0us 26.0us 27.0us 28.0us 29.0us 30.0us 31.0us 32.0us 33.0us 34.0us 35.0us 36.0us 37.0us 38.0us 39.0us
[I] RESET
[I] CLK
[O]gold1
[O]gold2
[O]mixout
[O]SW2
[O]SW1
[O]SAMPLE
[O]SAMCLK
[O]DUMP2
[O]DUMP1
Name:
41.0us 42.0us 43.0us 44.0us 45.0us 46.0us 47.0us 48.0us 49.0us 50.0us 51.0us 52.0us 53.0us 54.0us 55.0us 56.0us 57.0us 58.0us 59.0us
[I] RESET
[I] CLK
[O]gold1
[O]gold2
[O]mixout
[O]SW2
[O]SW1
[O]SAMPLE
[O]SAMCLK
[O]DUMP2
[O]DUMP1