NTNU
Innovation and Creativity

Evaluation of SDR-implementation of IEEE 802.15.4 Physical Layer

Roger Martinsen Koteng

Master of Science in Electronics
Submission date: July 2006
Supervisor: Lars Magne Lundheim, IET
Co-supervisor: Robin Hoel, Chipcon
Per Torstein Røine, Chipcon

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

Background:

Standardized radio interfaces are now available as one chip solutions by several manufacturers. Base band processing in these chips is done partly by programmable processors and partly by specialized units. In order to gain flexibility during development and to ease future modifications, it might be desirable that as much as possible of the signal processing is done in programmable units. Such implementations are often referred to as Software Defined Radio (SDR).

In collaboration with Chipcon AS two students will work with problems related to the development of a processor architecture for SDR development of radio interfaces of a complexity ranging from simple to medium. One of the students will be responsible for the processor architecture as such, while the other one will concentrate on a case study (the present thesis) giving input about typical algorithms and complexity for such a processor. For this case study the IEEE 802.15.4 physical layer has been chosen.

Problem specification:

Physical layer base band processing for a radio interface can be subdivided in several functionalities such as channel filter, time and frequency synchronization, correlation, demodulation, etc. Each functionality may be implemented by one or more algorithms, such as FIR filter, CORDIC computations, FFT, etc. Each of this algorithms can be described by a sequence of mathematical operations, such as multiplication, addition, absolute value etc.

Having made a mapping from functionalities to algorithms to mathematical operations, these can in turn be implemented as a sequence of instructions from the instruction set of a given processor. The instruction set relates to a programming model given by the processor architecture.

The purpose of the proposed thesis is, by means of a case study, to provide a set of algorithms described in such a way that a mapping to the instruction set given by the associated project mentioned above can be made.

Assignment given: 16. January 2006
Supervisor: Lars Magne Lundheim, IET

Abstract

The concept of software-defined radio (SDR) holds great promise. The idea behind SDR is to move software as close to the antenna as possible. This can improve flexibility, adaptability and reduce the time-to-market.

This thesis covers the evaluation of algorithms for implementing IEEE 802.15.4 physical layer. In collaboration with a digital circuit designer some of these algorithms were chosen and formed a basis for a DSP architecture optimized for low-complexity, low-power radio standards. The performance of a implementation using these algorithms were then evaluated by means of analytical computations and by simulation.

The simulations show that, if zero frequency or phase offset assumed, the 5.7×10^{-5} BER required by IEEE 802.15.4 physical layer standard can be reached for a -3.5 dB SNR at the input of the digital demodulator by using the chosen algorithms. Although, Simulations also show that the frequency and phase offset have great effect on the BER.

Preface

This report is the result of my work on the master thesis at the Norwegian University of Science and Technology, Department of Electronics and Telecommunications, in the spring of 2006.

The work was done in cooperation with another student, Hallvard Næss, who has got a background in digital hardware design. The work was mainly divided into separate area of expertize. I studied the standard and previous work done on this area and proposed different algorithms. Hallvard then analysed the feasibility of these algorithms for implementation in hardware. This was done iteratively, till a suitable solution was found. Therefore, certain parts of the work done by both members of the group had a high grade of integration. Though, the work finally led to two separate reports. I have therefore chosen to omit all details concerning hardware in this report, since it is covered in detail in Hallvard Næss' master thesis report [25].

First and foremost, I would like to thank my supervisor Lars Lundheim for great guidance along the way. I would also like to thank my team-mate, Hallvard Næss, for a constructive and good cooperation.

Roger Martinsen Koteng

Trondheim, 03.07.2006

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog-to-Digital Converter
AGC	Automatic Gain Control
ASIC	Application Specific Integrated Circuits
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CORDIC	COordinate Rotation DIgital Computer
DAC	Digital-to-Analog Converter
DCR	Cirect-Conversion Receiver
DSP	Digital Signal Processor
DSSS	Direct Sequence Spread Spectrum
ED	Energy Detection
FEC	Forward Error Correction
I	In-Phase
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, Scientific and Medical
LNA	Low-Noise Amplifier
LO	Local Oscillator
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Medium Access Control (layer)
MAP	Maximum A Posteriori
ML	Maximum Likelihood
Modem	Modulator/Demodulator
MSK	Minimum-Shift Keying
OSI	Open System Interconnection
O-QPSK	Offset Quadrature Phase Shift Keying
PER	Packet Error Rate
PHY	Physical (layer)
PN	Pseudo-random Noise
PPDU	PHY Protocol Data Unit
PPM	Parts per million
PSDU	Physical Layer Service Data Unit
Q	Quadrature-Phase

QoS	Quality of Service
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
SFD	Start-of-Frame Delimiter
SNR	Signal-to-Noise Ratio
VGA	Variable Gain Amplifier
WiFi	Wireless Fidelity
WPAN	Wireless Personal Area Networks

Chapter 1

Introduction

The ever increasing number of wireless standards have given an increase in the demand for versatility and updatability for laptops, cell phones, computer accessories, headphones etc. This may be solved by softening the hardware.

There are numerous interpretations of the words software based radio, software defined radio, software radio, adaptive intelligent software radio. In Tuttlebees book [33], software based radio is used as an overarching term for software defined radio, software radio and adaptive intelligent software radio.

Software defined radio is a radio communication system in which the characteristics of the radio is defined by digital signal processing in flexible and reconfigurable functional blocks. A software defined radio is dependent of an analog radio frequency (RF) front-end to down-convert the signal to baseband. In the future, better technology is expected to enable the digitalization to move even further toward the antenna, partly or totally eliminating the need for a analog front-end (except the antenna). This is called a software radio, in which all the processing required for the radio is performed by software. As the software radio evolve further, it will be capable of adapting to the environment in order to increase its performance. It is then called an adaptive intelligent software radio. For further details see [33, page 3-22]

The possible advantages of using software defined radio applications are that they allow flexible architectures for a wide range of communication products, the ability to provide updated, enhanced, or replaced capability via a download mechanism. From the providers perspective, this facilitates the production and helps to lower the time-to-marked. For a consumer, this means that new functionality can be installed on for example the cellular phone, without having to replace the hardware.

The fundamental problem when designing a software defined radio is that performance, power, and flexibility are conflicting goals. The general purpose processors at one extreme maximizes the flexibility, whereas application specific integrated circuits (ASIC) at the other maximizes the performance and minimizes the power.

Figure 1.1 shows a typical software defined radio. The scope of this thesis will mainly be the digital demodulator. The RF front-end and modulator will also be discussed briefly. The reader is expected to gain insight into different trade-offs when designing a digital demodulator and the influence of the choices on the performance. Thus, this thesis not intended to present the

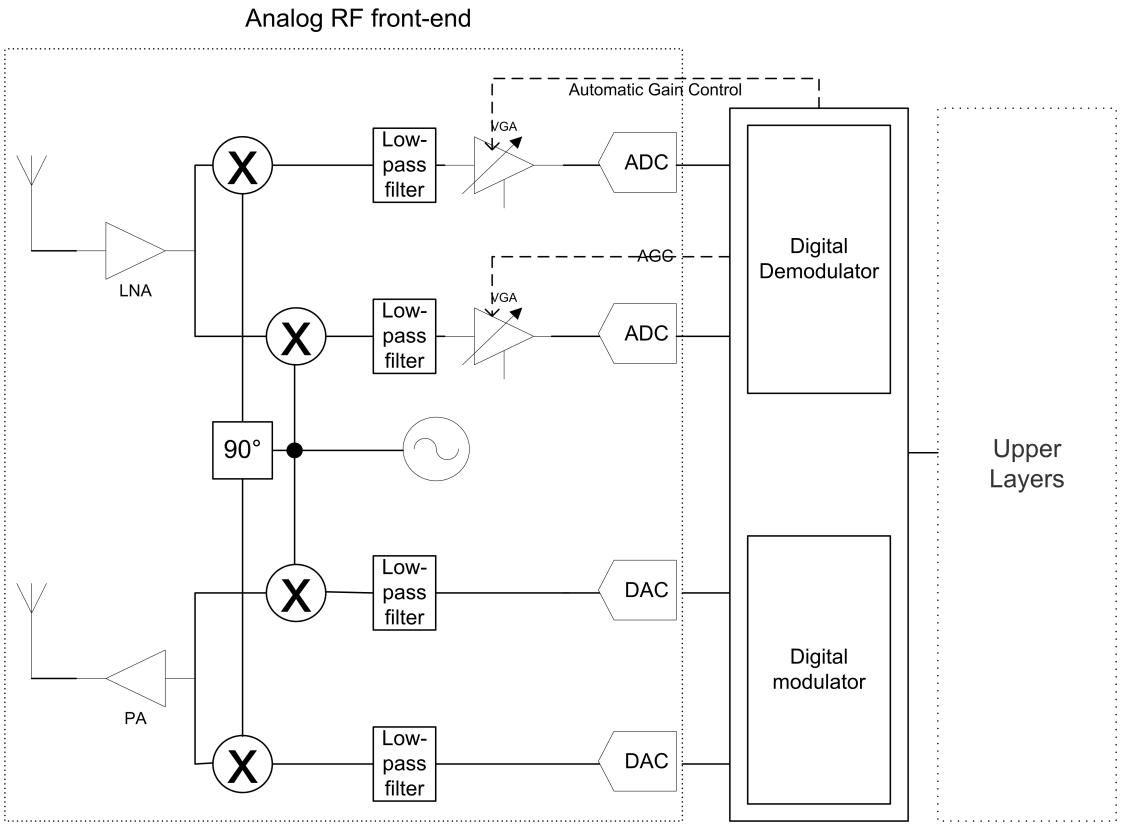


Figure 1.1: Software defined radio.

solution for an optimized demodulator.

This thesis will start by briefly presenting the IEEE 802.15.4 2.4 GHz physical layer standard. Chapter 3 will on basis of the standard propose and discuss different realizations of a software-defined radio digital demodulator. Chapter 4 will briefly present the digital modulator. Chapter 5 will give a short summary of the complexity of the different algorithms discussed in chapter 3 and 4. The described algorithms will form a basis for the development of a digital signal processor (DSP) architecture. By iteratively adjusting the proposed algorithms and DSP architecture, a suitable solution will be found. The proposed (DSP) architecture is found in [25]. Chapter 6 be will give a more thorough analysis of the digital demodulator by only looking at one chosen realization. Chapter 7 will present a simulation of different variations of the digital demodulator.

Chapter 2

Wireless Personal Area Networks (WPANs)

The focus of wireless personal area networks (WPAN) is low-cost, low power, short range and very small size. There are three classes of WPANs; high rate WPAN (HR-WPAN), medium rate WPAN (MR-WPAN) and low rate WPAN (LR-WPAN). The IEEE 802.15.3 is the high rate WPAN, IEEE 802.15.1/Bluetooth is the medium rate WPAN and IEEE 802.15.4/ZigBee is the low rate WPAN. Unlike for wireless local area network, a connection trough a WPAN can be made almost without infrastructure. This chapter will summarize the IEEE 802.15.4 standard, and clarify some of the theory necessary in order to understand the standard.

2.1 ZigBee/IEEE 802.15.4

The IEEE 802.15.4 standard is a Low-Rate Wireless Personal Area Network (LR-WPAN) standard. A LR-WPAN is a simple, low-cost communication network that allows wireless connectivity in low-power and low-throughput applications. The main goals of an LR-WPAN are ease of installation, reliable data transfer, short-range operation, extremely low cost, and a reasonable battery life, while maintaining simplicity and flexibility [12].

The ZigBee alliance was formed prior to the formation of the IEEE 802.15.4 group, but as they soon discovered, both were aiming at the same goal. Later, the ZigBee Alliance and the IEEE 802.15.4 group decided to cooperate and ZigBee is today the commercial name for this technology [36]. However, the two groups still work on different parts of the technology. The IEEE 802.15.4 group has standardized the physical- (PHY) and the medium access control (MAC) layers [12], whereas the ZigBee alliance concentrates on the higher layers.

The ZigBee stack architecture is based on the open system interconnection (OSI) standard. The OSI model is a layered abstract description for communication and computer network protocol design. It describes how the layers should interface, which dictates how the layers interacts. There are seven layers: Application, Presentation, Session, Transport, Network, Data Link, Physical [31]. The IEEE 802.15.4 standard defines the network and physical layer, whereas the ZigBee standard builds on this foundation to provide the upper layers.

The main goal of the ZigBee technology is quite different from for example Bluetooth or wireless fidelity (WiFi) applications. Instead of offering very high rates at long distances with high quality of service (QoS) requirements, it is intended to serve industrial, residential and medical applications with very low power consumption and cost requirements. This can be achieved as a result of the low data rate supported, and a battery lifetime of 6 months up to several years is achievable.

ZigBee operates in the unlicensed industrial, scientific and medical (ISM) bands, and the range is 10-75 m. One of the properties that makes this standard more power efficient is the ability to wake up from a sleep state very fast. A node that is powered down can wake up and get a packet within about 15 ms. This enables the nodes to go into a sleep state more often to conserve power. The duty cycles of a typical Zigbee/802.15.4 application is expected to be under 1%.

This chapter will give a brief overview of the IEEE 802.15.4 standard, the main scope will be on the 2,4 GHz physical layer (PHY). For further details about the IEEE 802.15.4 and the ZigBee standard see [12, 1, 36].

2.1.1 Frequency Bands

The standard specifies operation within 27 channels, distributed over 3 bands. 16 of these channels are located in the 2450 MHz band, 10 channels located in the 915 MHz band and the last channel is located in the 868 MHz. The center frequencies of the different channels are given by

$$F_c(k) = \begin{cases} 868,3 \text{ MHz} & \text{for k} = 0, \\ 906 + 2(k-1) \text{ MHz} & \text{for k} = 1,2,\ldots,10, \\ 2405 + 5(k-11) \text{ MHz} & \text{for k} = 11,12,\ldots,26, \end{cases} \tag{2.1.1}$$

where k is the channel number. The distribution of the channels in the 2450 MHz band is drawn in figure 2.1.

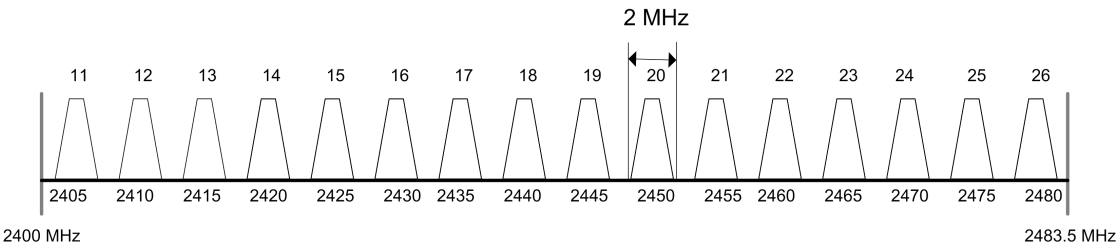


Figure 2.1: The distribution of the channels in the IEEE 802.15.4 2.4GHz physical layer.

2.1.2 Coding

In the 2.4 GHz physical layer, symbols are coded using direct sequence spread spectrum, in which each symbol is mapped to a pseudo-random noise (PN) sequence. A PN sequence will

appear as almost random, with an almost flat frequency spectrum, approximately as many 0's as 1's, narrow peak in the autocorrelation function and low cross-correlation between any two sequences. see [28, page 329-339] and [14, page 479-508] for further details about spread spectrum techniques.

Every 4 bits are grouped into symbols, which can be combined in 16 different ways. Each the 16 4-bit symbols are then mapped to a 32-chip PN sequences. The figure 2.2 shows the first and second PN-sequence, where the second symbol is given by a cyclic right-shift by 4 bits. The next 6 symbols are generated in the same fashion. Because of the narrow autocorrelation function of a PN-sequence, a sequence and the shifted version of the same sequence will stay almost uncorrelated. The last 8 symbols are given by inverting the even-indexed bits in the first 8 symbols. The even-indexed bits are marked as bold in figure 2.2.

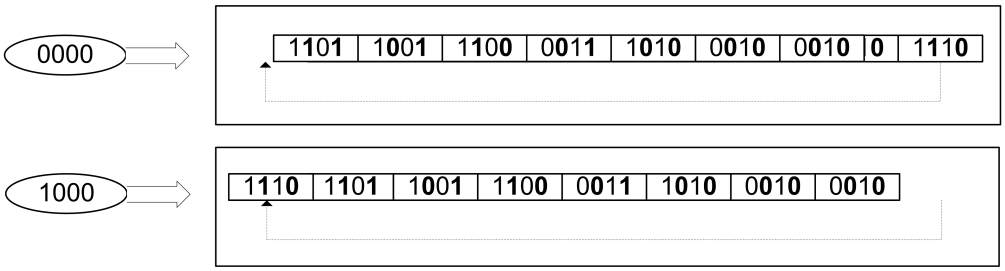


Figure 2.2: PN-sequence of symbol 0 (0000) and symbol 1 (1000)

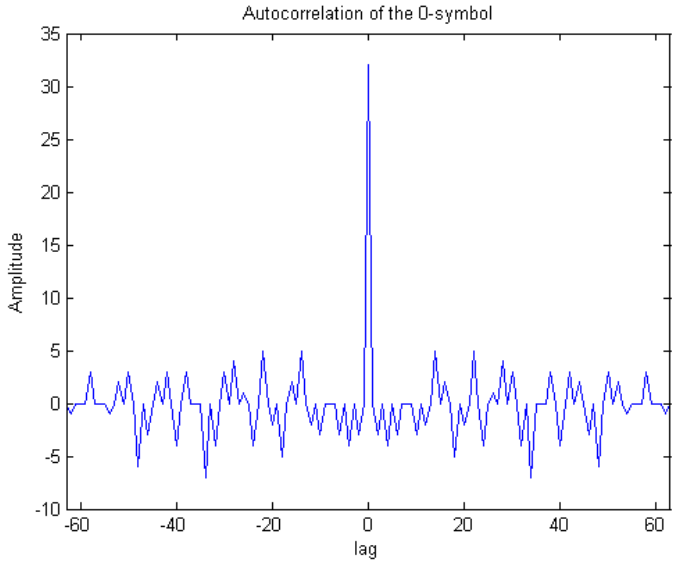


Figure 2.3: Autocorrelation of symbol 0 with half-sine pulse shape and 2 samples per chip.

2.1.3 PHY Protocol Data Unit

The general The PHY Protocol Data Unit (PPDU) packet format is given in figure 2.4.

<table>
<tr><td colspan="2">Number of bytes: 4</td><td>1</td><td colspan="2">1</td><td>Variable</td></tr>
<tr><td colspan="2">Preamble</td><td>Start-of-frame delimiter (SFD)</td><td>Frame lenght</td><td>Reserved</td><td>PSDU</td></tr>
<tr><td colspan="3">Synchronization header (SHR)</td><td colspan="2">PHR</td><td>PHY payload</td></tr>
</table>

Figure 2.4: Format of the PPDU

The preamble field is used by the transceiver to obtain chip and symbol synchronization with an incoming message. The preamble field is composed of 32 binary zeros, i.e. eight 0-symbols. The start-of-frame delimiter (SFD) is an 8 bit field indicating the end of the synchronization (preamble) field, and it consists of the bits 1110 0101. The 7 bits frame length field specifies the total number of octets (bytes) contained in the physical layer service data unit (PSDU), i.e. the physical layer payload field. The PSDU field has a variable length and carries the data of the physical layer packet.

The synchronization header, i.e the preamble and SFD, has the important task of gaining synchronization in the demodulator. This in crucial in order to achieve reliable demodulation in a coherent demodulator.

2.1.4 Modulation

The standard defines the even-indexed bits to be modulated onto the in-phase (I) carrier and the odd-indexed bits to be modulated onto the quadrature-phase (Q) carrier . The Q carrier is delayed by a half chip-period compare to the I carrier. The pulse-shaped baseband signal is shown for the 0-symbol in figure 2.5.

The standard for IEEE 802.15.4 2.4 GHz physical layer describes modulation of the bits to an offset Quadrature phase shift keying (O-QPSK) with half-sine pulse shaping, which in effect makes it a minimum-shift keying (MSK) modulated signal [30]. The main properties of a MSK signal are constant envelope, relatively narrow bandwidth and coherent detection performance equivalent to that of QPSK. The reader is referred to [14, page 345-407] and [5] for further details. Figure 2.7 shows the constellation of a MSK modulated signal. The transitions are only made on the unit circle as a consequence of the half-sine pulse shape and phase offset. The offset allows only the phase of one carrier to be changed at the time, thus only $\pi/2$ phase shifts. Figure 2.6 shows the QPSK and MSK power spectra. It can be seen that QPSK without half-sine pulse shaping falls off more rapidly than the MSK, but has higher side lobes, which results in more out-of-band interference. w2

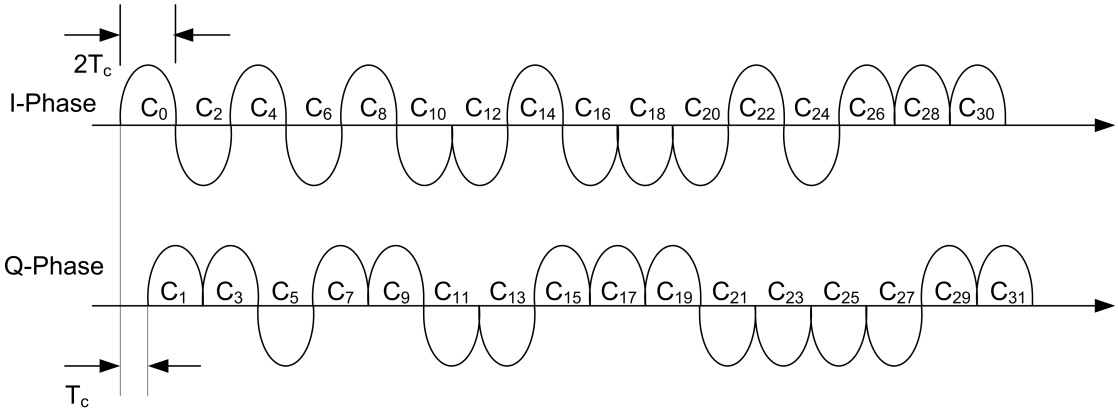


Figure 2.5: Baseband chip sequence of symbol 0

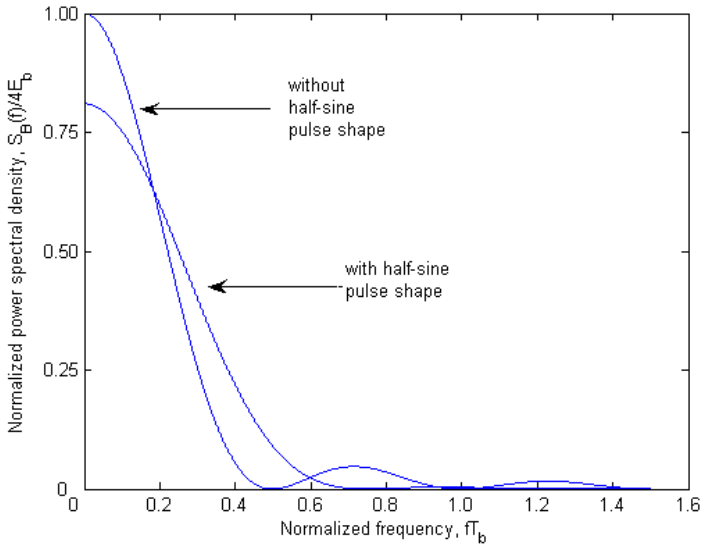


Figure 2.6: Power Spectrum of QPSK With and Without Half-sine Pulse Shape

2.1.5 Rates

The symbol rate is 62.5 ksymbols/s, where each symbol consists of 4 bits, which gives a bit rate of 250 kbit/s. Each symbol is coded with 32 chips, which gives a total chip rate of 2 Mchips/s. The chips are modulated using an O-QPSK, which in average represents 2 bits as one channel symbol.

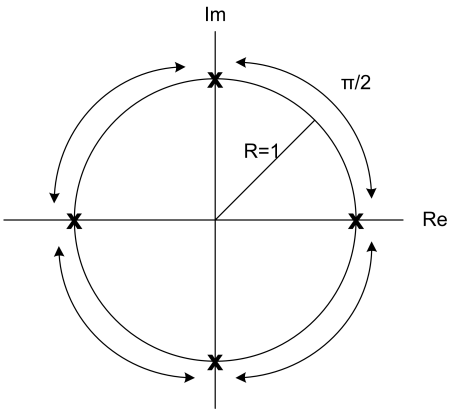


Figure 2.7: O-QPSK with half sine pulse shaping (MSK) constellation.

2.1.6 Receiver Energy Detection

'The receiver energy detection (ED) measurement is intended for use by a network layer as part of a channel selection algorithm. It is an estimate of the received signal power within the bandwidth of an IEEE 802.15.4 channel.'[12, page 53]

2.1.7 Link Quality Indicator

'The Link Quality Indicator (LQI) measurement is a characterization of the strength and/or quality of a received packet. The measurement may be implemented using receiver ED, a signal-to-noise ratio estimation, or a combination of these methods. The use of the LQI result by the network or application layers is not specified in the IEEE 802.15.4 standard.'[12, page]

Chapter 3

Receiver

The receiver of a software-defined radio will consist of an analog front-end and a digital modulator/demodulator (modem). Figure 1.1 shows the general structure of a SDR radio with direct down-conversion analog front-end. This chapter will propose possible algorithms for realization of the digital demodulator. For completeness there will also be given a brief presentation of the analog radio frequency (RF) front-end.

There are two types of demodulation, coherent and incoherent. In coherent demodulation, the phase information is required in order to reliably demodulate the signal, whereas in incoherent demodulation there is no need for phase-information. Coherent demodulation has better performance than the incoherent demodulation, but has higher complexity. Since the IEEE 802.15.4 standard uses a offset quadrature phase shift keying (O-QPSK) modulation, a coherent demodulation is used. The demodulator uses a portion of the preamble of the incoming IEEE 802.15.4 packet to correct the frequency and phase offset caused by non-perfect up- and down-conversion, Doppler shifts et cetera. When the frequency offset is corrected for, the receiver starts looking for the start-of-frame delimiter (SFD). A detection will give the correct frame synchronization and tell the demodulator to start receiving payload.

The receiver presented in this chapter is based on Chipcon's 2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver CC2420 [8]. A block diagram of the proposed receiver is shown in figure 3.1.

This chapter will discuss the proposed receiver block by block, starting left. The channel is assumed to add white Gaussian noise, with mean, $\mu = 0$ and variance $\sigma = N_0/2$.

3.1 Analog-to-Digital Converter

The purpose of the analog-to-digital converter is to digitalize the incoming analog signal. Digitalization is done by first sampling the signal in time, which gives a time-discrete and amplitude-continuous signal. The signal is done amplitude discrete by quantization. Quantization is done by approximating the continious amplitude to a finite number of amplitude levels. The number of levels is given by 2^R, where R is the number of bits per sample.

The Analog-to-Digital Converter (ADC) used in the proposed demodulator has a sampling rate of 8 MSamples/s, with a accuracy of 8 bits. It is assumed ideal, and thus disregarding offset

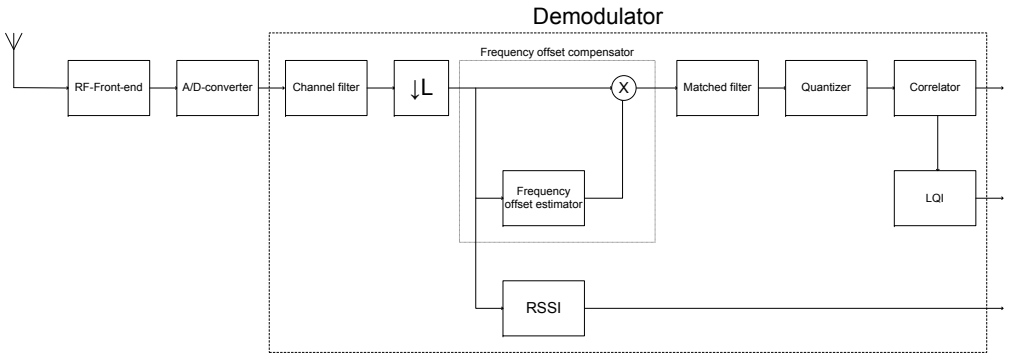


Figure 3.1: A simplified block diagram of the proposed receiver.

and gain errors, integral and diffential nonlinearities errors et cetera [17, page 454-459]. It is stated in [7] that a resolution of 8 bits is sufficient, and thus the data paths are assumed to be 8 bits wide. For further details about the choice of data path width and ADC, the reader is referred to [25].

The Nyquist sampling rate for IEEE 802.15.4 2.4 GHz physical layer is 2 MSamples/s. The 4 times oversampling used in the proposed ADC allows use of lower complexity on both the analog and digital filter. This does not give any savings in terms of the total computational requirements for the digital filter, since the number of incoming samples is 4 times higher and the number of coefficients is approximately $1/4$ of the non-oversampled case, but this reduces the memory requirements. The saving for the analog filter is big, since the analog spectrum will be repeated for the sampling frequency in the ADC. Further details about the analog and digital filter will be discussed in the following sections.

Oversampling also spreads the quantization noise over a broader bandwidth and therefore reduces the noise level.

3.2 Quantizer

As seen in the previous section, the quantizer in the ADC lowers the precision of the analog signal down to a finite number of bits. In order to reduce the complexity of the demodulator, the digital signal can be quantized again, on the expense of increasing the noise level. The noise level increases as a consequence of the reduced number of amplitude levels, and thus a coarser estimate of the original analog signal. The quantizer can be placed where it is found most suitable by the designer. In figure 3.1, the quantizer is put in front of the correlator, but this is not the only solution.

3.3 Radio Frequency Front-end

The basic function of the radio frequency (RF) front-end is to amplify the signal received at the antenna, and convert it from a carrier frequency down to baseband. Traditionally, the most common RF front-end is the superheterodyne receiver, which converts the passband signal down to baseband in two steps. This improves the channel selectivity due to sharper filters [27]. Though, harsh demands for low cost and smaller size in many applications, have given rise for the use of direct-conversion receiver, which converts the signal down to baseband in one step. The direct-conversion receiver eliminates one filter and one local oscillator (LO) and increases the level of integration compared to the superheterodyne receiver, on the expense of increasing the sensitivity towards non-linearities and noise [29]. For further details about direct-conversion receiver the reader is referred to [29] and to [24] for details about direct-conversion receiver design for IEEE 802.15.4. A direct-conversion receiver is drawn in figure 3.2.

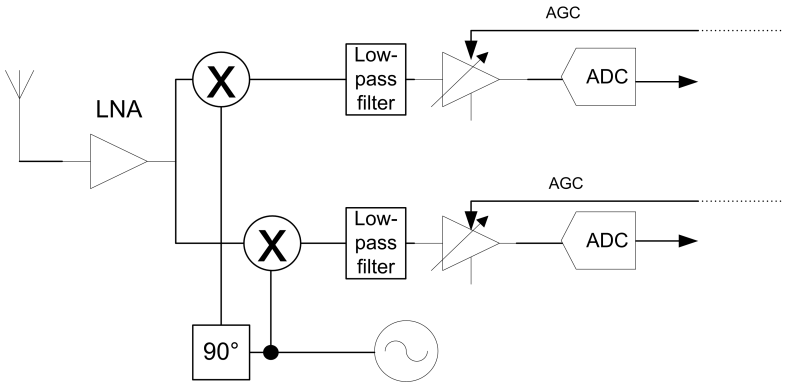
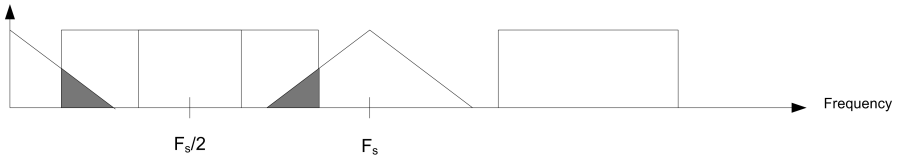


Figure 3.2: Direct conversion receiver.

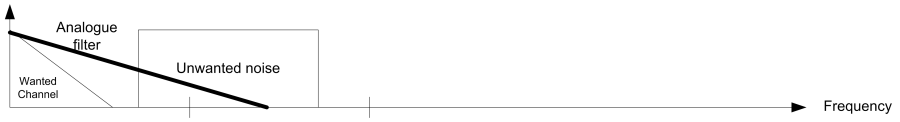
The RF front-end of the direct conversion receiver consists of a low-noise amplifier (LNA), quadrature mixer, low-pass filter and automatic gain control (AGC). The aim of the LNA is to amplify the incoming signal without introducing new noise. The purpose of quadrature mixer is to convert the signal down to baseband. The AGC aims to scale the signal level at the input of the analog-to-digital converter (ADC) to be the same irrespective of the original amplitude. This is done by sending a controll signal to the variable gain amplifier (VGA). This is to fully utilize the dynamic range of the ADC. The analog low-pass filter aims to attenuate all the unwanted signal power, e.g Gaussian noise, mirror components and other channels, above $F = F_S/2$, where F_S is the sampling frequency. This is to prevent aliasing. Aliasing is an effect that occurs when a time continuous signal is made time discrete by sampling at a rate lower than the Nyquist rate. Aliasing is impossible to remove once it has occurred, and is therefore important to prevent, and the effect of not having an anti-aliasing filter is shown in figure 3.3. Figure 3.3a and 3.3c shows the spectrum of the original analog signal and figure 3.3d and 3.3b shows the resulting sampled signal with and without using an analog anti-aliasing filter, respectively. The spectrum is repeated around the sampling frequency, F_S, so the highest frequencies of the unwanted noise

is 'folded' into the wanted spectrum. It can be seen from figure 3.3d, that in this case all of the unwanted spectrum is not removed, but it is not aliased into the wanted channel, and can therefore be removed at a later stage. The complexity of the analog filter must be high enough to prevent aliasing, but it can leave some of the noise for the digital filter to handle, as seen from figure 3.3d.

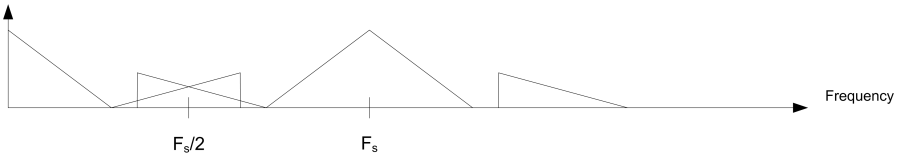
(a) The spectrum of the analog signal



(b) The spectrum of the sampled, aliased signal



(c) The spectrum of the analog signal



(d) The spectrum of the sampled signal after using a anti-aliasing filter

Figure 3.3: The effect of the anti-aliasing filter.

When implementing an IEEE 802.15.4 receiver, the cut-off frequency in stop-band can be as high as 7 MHz, if a sampling rate F_S=8 Msamples/s, because there is a 3 MHz separation between the channels. However, in this case the digital filter would have to be ideal in order to remove all the unwanted noise. The trade-off between the analog and digital filter is further discussed in the following section. For further details about aliasing and filter design, the reader is referred to [22].

3.4 Channel Filter

In the IEEE 802.15.4 2.4 GHz physical layer, the channels are required to possess a bandwidth in passband of 2 MHz each, as seen from figure 2.1. A channel is chosen by tuning the local oscillator (LO) frequency to convert the wanted channel down to baseband. Generally, the ob-

jective of the channel filter is to attenuate all the channels and noise outside the wanted channel. It also provides a fixed channel bandwidth.

The main objective of this section is to illustrate the trade-offs between analog and digital filter. In order to ease the illustration, it is assumed that the only noise present is from other IEEE 802.15.4-channels and no other noise. Three case will be presented. In case 1 a low-complexity analog filter is used, in case 2 a high complexity analog filter and case 3 will present a combination of the first two cases.

The general channel filter requirements can be derived from the conditions of the jamming resistance. The IEEE 802.15.4 2.4 GHz physical layer requires 0 dB rejection at the adjacent channel (2 MHz apart) and 30 dB rejection at an alternate channel (4 MHz apart) [12, page 48]. If adding a 10 dB margin, the stop-band attenuation will be 40 dB. The analog filter can be designed as a Butterworth-filter and the digital filter as linear phase equiripple FIR-filter using Parks-McClellan algorithm.

Case 1: low-complexity analog filter The low complexity analog filter can be designed with 1-dB cutoff-frequency at 1.1 MHz and a minimum attenuation of 40 dB at 7 MHz. The minimum order, N, of the Butterworth is computed as shown in [22, page 315-317] to be $N = 3$. The low order of the analog filter has to be compensated for by the digital filter. Assuming sampling rate, F_S=8 MSamples/s, the channel filter can be designed with cutoff frequencies in pass-band and stop-band, F_p=1.1 MHz and F_S=2 MHz, minimum stop-band attenuation A_S=40 dB and ripple in passband, A_S=1 dB. The order in this case will then be 13. Figure 3.4 shows the analog spectrum (top), where the bold line shows the magnitude response of the analog filter , the filtered analog signal (mid) and the digital spectrum (low), where the bold line is the magnitude response of the digital filter.

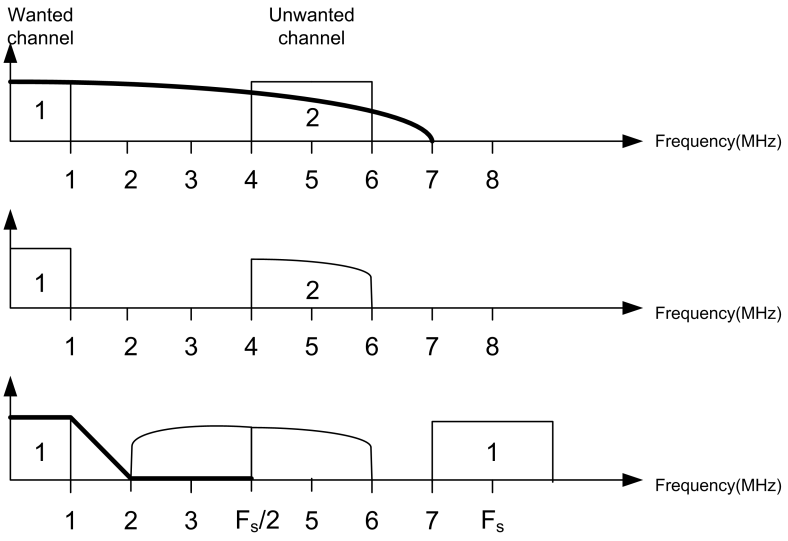


Figure 3.4: Low complexity analog filter.

Case 2: High-complexity analog filter The high complexity analog filter can be designed with 1-dB cutoff-frequency at 1.1 MHz and a minimum attenuation of 40 dB at 4 MHz. The minimum order, N, of the Butterworth is computed to be N=5. The digital channel filter is designed with the same specifications as in case 1 except for the cutoff in stop-band, F_p=4MHz. The specifications can then be fulfilled by using a filter with N=3. Figure 3.5 shows the analog spectrum (top), where the bold line shows the magnitude respons of the analog filter , the filtered analog signal (mid) and the digital spectrum (low), where the bold line is the magnitude response of the digital filter.

The complexity of the digital channel filter can be lowered some by designing a half-band FIR filter. The complexity reduction is a consequence of the fact that almost half of the coefficients are zero. This filter has a 6-dB cutoff fixed at $F_C = F_S/4$ and can therefore not fill the requirements of case 1. The lowest order of this filter is N=6, so it will not result in any complexity reduction in case 2 either, but a compromise between the two cases can give complexity reduction for the whole system. The actual savings in using a half-band filter depends on the digital signal processor (DSP) architecture. For further reading about half-band filters see [22, page 700-705].

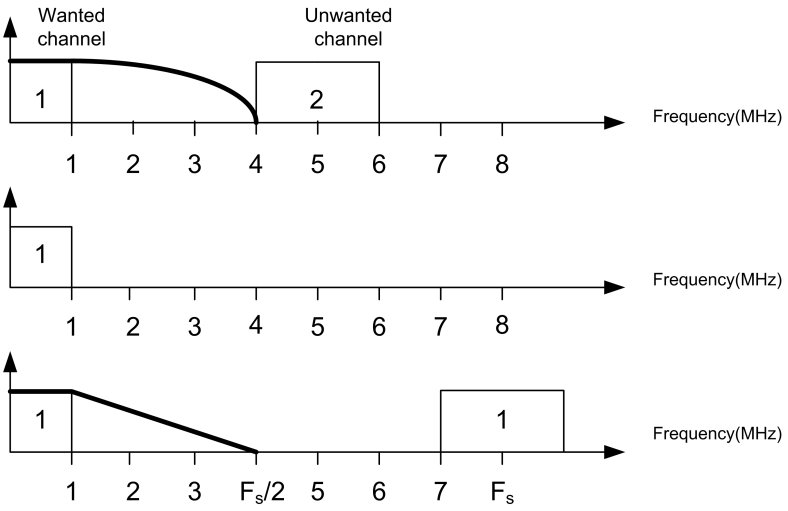


Figure 3.5: High complexity analog filter.

Case 3: Compromise between the analog and digital filter As a compromise between case 1 and case 2, one can concider a specification where the 1-dB cutoff-frequency is at 1.1 MHz and a minimum attenuation of 40 dB at 5.3 MHz. The minimum order, N, of the Butterworth is computed as to be N=4. A channel filter designed with the same specifications as in case 1 except for the cutoff-frequency in stop-band, F_S=2.7 MHz. The specifications can then be fulfilled by using a FIR filter with N=11. Figure 3.6 shows the analog spectrum (top), where the bold line shows the magnitude respons of the analog filter , the filtered analog signal (mid) and the digital spectrum (low), where the bold line is the magnitude response of the digital filter.

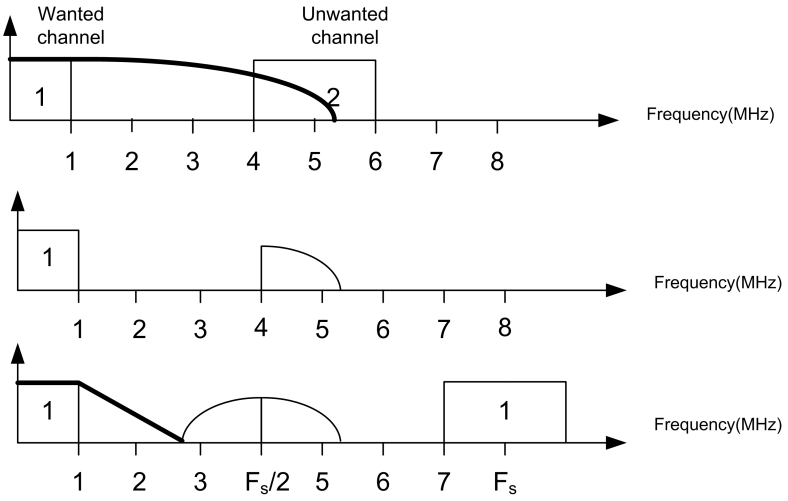


Figure 3.6: Compromise between the analog and digital filter.

The general idea in software radio is that as much as possible should be put into the digital part of the receiver. This means that case 1 generally is the one most suited for this purpose. Though, it was chosen to focus on case 3. The reason for this is that the digital filter would never the able to remove all the unwanted noise, so a compromise will help maximizing the performance in terms of noise.

3.4.1 Realizations of the Channel Filter

Some possible realizations of the channel filter are direct form, transposed form or a polyphase realization of these two. In addition to the mentioned realizations, realizations can also be designed to take advantage of the inherent symmetry in the linear phase FIR-filters. A linear phase FIR filter will always be symmetric or anti-symmetric around N/2, where N is the filter order [22, page 226-231]. This means that the number of multiplications will be approximately half of what could be obtained with a straight forward realization. The actual advantage of these simplifications depends on the architecture of the DSP. Three candidates for realization of the 11^{th} order linear phase FIR channel filter are presented in the following.

Candidate 1: Direct form A direct form FIR implementation of the channel filter is a straight forward implementation of the correlation between the incoming sequence and the time-reversed channel filter coefficients, i.e a convolution. This filter is shown in figure 3.7.

Candidate 2: Transposed form A transposed form FIR filter channel filter is found by reversing all paths, replacing the pick-off nodes by adders, and vice versa, and interchange the input and output nodes. This filter is shown in figure 3.8.

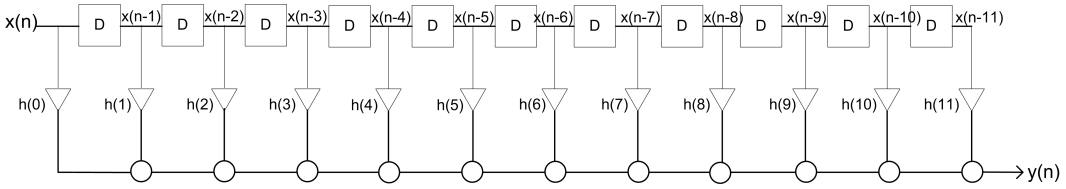


Figure 3.7: Direct form implementation of the channel filter.

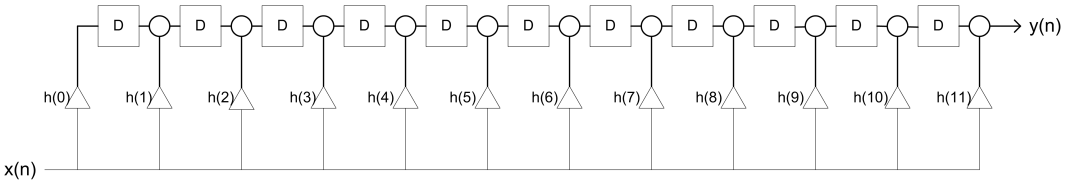


Figure 3.8: Transposed form implementation of the channel filter.

Candidate 3: Exploiting the symmetry In a linear phase FIR filter of order 11, i.e length 12, there will be symmetry around the 6^{th} coefficient, i.e $h(0) = h(11)$, $h(1) = h(10)$, $h(2) = h(9)$, $h(3) = h(8)$, $h(4) = h(7)$ and $h(5) = h(6)$. A FIR-filter realization which exploits this symmetry in the structure shown in figure 3.9.

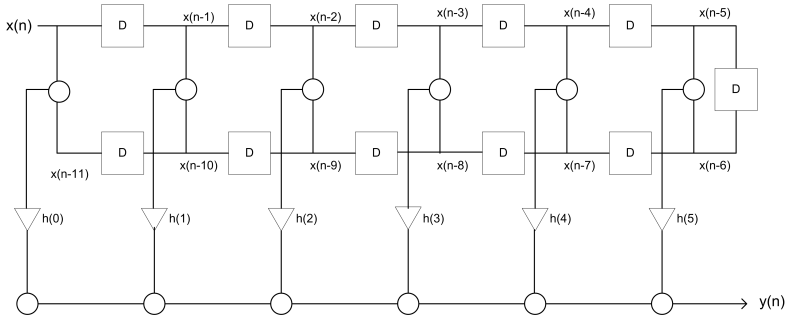


Figure 3.9: Exploiting the symmetry of the channel filter

For further details about filter design see [22].

3.5 Down Sampler

The down sampler lowers the sampling rate to be 4 times the chip-rate for the I- and Q-branch, i.e 2 times oversampling. This means that the analog spectrum of the signal is repeated around the new sampling frequency, $\hat{F}_s = F_s/2$. Which gives a sampling rate of 4 Msamples/s for IEEE 802.15.4 2.4 GHz physical layer.

The realization of the down sampler depends on the DSP architecture.

3.6 Received Signal Strength Indicator

The receiver energy detection (ED) is an estimate of the received signal power within the bandwidth of an IEEE 802.15.4 channel. The Received Signal Strength Indicator (RSSI) is the implementation of the receiver energy detection, and can be implemented by measuring the accumulated energy over a period of N samples. The power is found by dividing the energy by the number of samples. This gives two possible algorithms for the RSSI

$$RSSI_1 = \sum_{n=0}^{N-1} |m(n)|^2 = \sum_{n=0}^{N-1} s_I^2(n) + s_Q^2(n), \tag{3.6.1}$$

and

$$RSSI_2 = \frac{1}{N} \sum_{n=0}^{N-1} |m(n)|^2 = \frac{1}{N} \sum_{n=0}^{N-1} s_I^2(n) + s_Q^2(n), \tag{3.6.2}$$

where N is the length of the data word over which the accumulation or averaging is done.

3.6.1 Realizations of the Received Signal Strength Indicator

Two possible realizations of the RSSI are a direct Pythagoras-based multiply-accumulate realization or realization by making use of the COordinate Rotation DIgital Computer (CORDIC) in vectoring mode. The CORDIC-algorithm is a trigonometric algorithm, and it can operate in two different modes; rotation and vectoring. In rotation mode the incoming I- and Q- samples are rotated by a preassigned angle. In vectoring mode, the CORDIC rotates the input vector till it is as aligned to the x-axis as the precision allows it. Some of the applications of the CORDIC are to compute sine and cosine, convert from polar to rectangular coordinates and vice versa, finding the vector magnitude et cetera [10, 4]. From figure 3.10 it can be seen how $|m(n)|$ can be found by rotating the sample point down to the x-axis and returning the amplitude along the x-axis.

3.7 Frequency and Phase Offset Compensator

In a typical wireless communication system, imperfect up- and down-conversion caused by non-idealities in the transmitter and receiver local oscillators result in a carrier offset at the receiver. This offset causes a continuous rotation of the signal constellation, and must be corrected for in order to achieve reliable demodulation of the received signal. A frequency and phase offset compensator must first estimate this frequency and phase offset and then compensate for it. The frequency and phase offset estimation can be done in two steps, where the first step is to make a coarse estimation directly on the incoming sequence. This estimate is then used to compensate for the rotation of the constellation. The samples go through the system and are fed to the correlator where the second estimate of the frequency is done. The code and processing

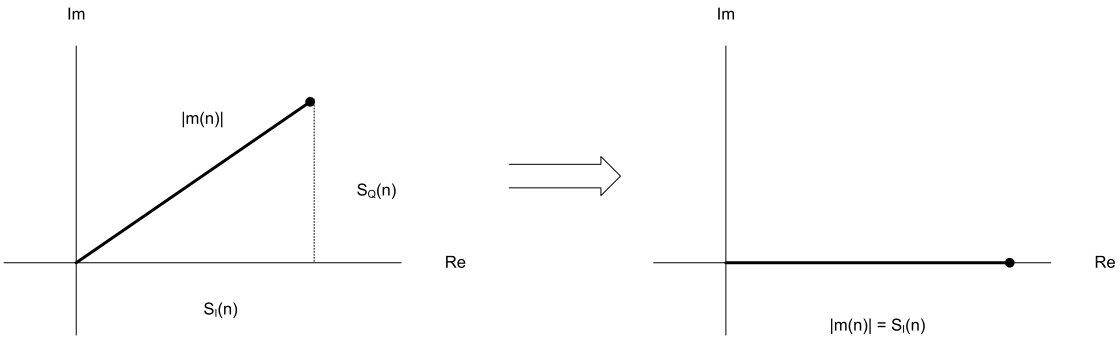


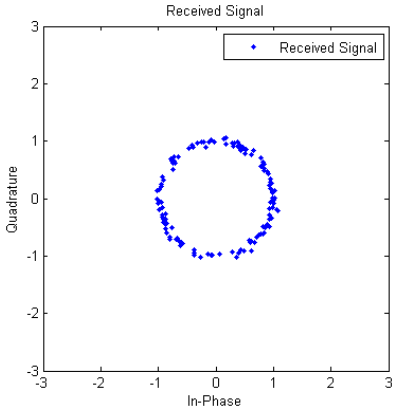
Figure 3.10: Geometric Representation of a Sample in Signal Space.

gain is added to signal to noise ratio (SNR) in the correlator, so the fine estimator is based on a signal with a higher signal to noise ratio (SNR) and much slower rotation than the coarse estimator. It is also data-aided, meaning that it knows what to expect and therefore can use this information to get a clear reference. This is not the case for the coarse frequency offset estimator, where the variance of the signal add uncertainty to the estimation. This leads to higher precision in the final estimate using a fine data-aided frequency and phase estimator. The computed phase angles used in the frequency offset estimator can be used to compensate for the constant phase offset. This is done by initializing the phase offset of the phase accumulator. Figure 3.11 shows the signal constellation at the receiver when 256 random bits are modulated using O-QPSK and sent through a channel with a 25 dB SNR. $\Delta\omega$ is the frequency offset and θ_0 is the phase offset. Figure 3.11a shows the received signal with no frequency offset compensation or synchronization. Figure 3.11b and 3.11c shows the signal constellation after the course and fine frequency offset compensation. Lastly, in figure 3.11d the phase error is corrected for.

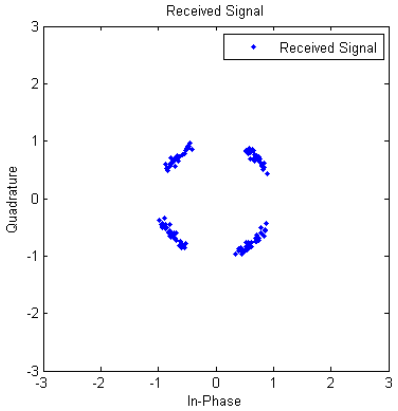
This section will first present different estimators suitable for making both blind, coarse estimates and finer data-aided estimates of the offset in the frequency and phase. Then, different possible realizations of the frequency and phase offset compensator, which includes both estimator and complex rotation, will be proposed.

3.7.1 Coarse Frequency Offset Estimator

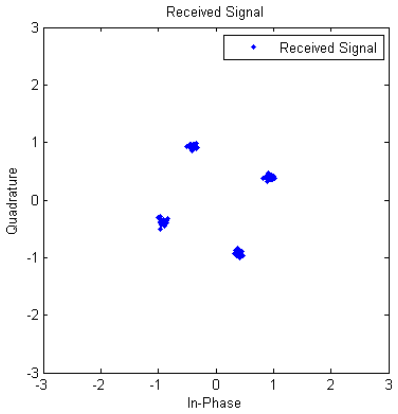
There are numerous ways to estimate the frequency. One method for obtaining good estimates is to find the peak of the Periodogram. This can be implemented efficiently by using a FFT of the discrete incoming signal followed by a search, or a interpolation for the spectral maxima. These methods are not suited when the spikes in the periodogram is closely spaced or if the constraints in complexity are strict [34]. For the IEEE 802.15.4 the latter applies. In this case, either data based or correlation based frequency estimator are best suited [34]. The phase averager and weighted phase averager belongs to the former class, and weighted linear prediction estimator and linear prediction estimator belongs to the latter [18]. The phase averager using lag, $D = 1$ was originally presented by Steven Kay [18], and will therefore be termed Kay estimator. The estimator is termed the generalized Kay estimator when $D \leq 2$ is used. The same applies for the



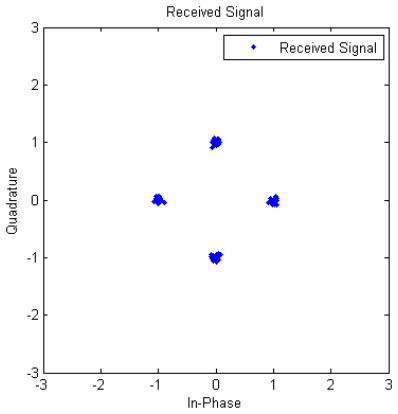
(a) $\Delta\omega = 0.156$, $\theta_0 = \pi/8$, $SNR = 25\,dB$



(b) $\Delta\omega = 5 \times 10^{-3}$, $\theta_0 = \pi/8$, $SNR = 25\,dB$



(c) $\Delta\omega = 10^{-5}$, $\theta_0 = \pi/8$, $SNR = 25\,dB$



(d) $\Delta\omega = 10^{-5}$, $\theta_0 = 0$, $SNR = 25\,dB$

Figure 3.11: The effect of frequency and phase offset compensation.

linear prediction estimator, which was first introduced by Ferdinand Classen and Heinrich Meyr [9], and will be termed the generalized Meyr estimator in the case when $D \leq 2$. The estimated frequency by using a Kay estimator is

$$\hat{\omega} = \frac{1}{D} \sum_{t=0}^{N-2} w_t \angle(x_t^* x_{t+D}), \tag{3.7.1}$$

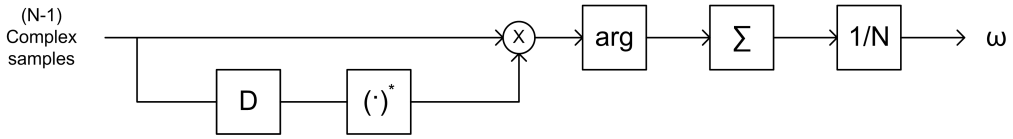
and by using Meyr estimaor is

$$\hat{\omega} = \frac{1}{D} \angle\left(\sum_{t=0}^{N-2} w_t x_t^* x_{t+D} \right), \tag{3.7.2}$$

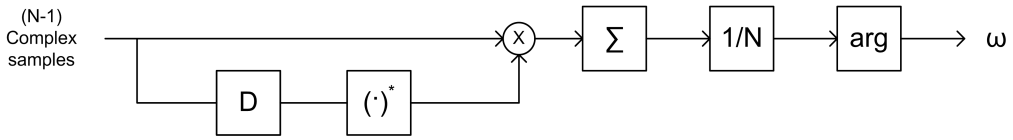
where N is the number of samples used to obtain the estimate. D is the lag between the samples in the incoming signal x_t and

$$w_t = \frac{\frac{3}{2}N}{N^2-1}\{1-[\frac{t-(\frac{N}{2}-1)}{\frac{N}{2}}]^2\} \tag{3.7.3}$$

is the weight function for the weighted generalized Kay and Meyr. The weight function for the unweighted generalized Kay and Meyr estimator is $1/(N-1)$. The block diagrams of the generalized Kay and Meyr estimator are shown in figure 3.12, where the blocks have the same meaning as in equation 3.7.1 and 3.7.2.



(a) block diagram of the generalized Kay estimator



(b) block diagram of the the generalized Meyr estimator

Figure 3.12: The block diagrams of the coarse frequency offset estimators.

The general idea of the estimators is that speed is given by the displacement divided by time, i.e $\omega = \frac{\theta_D - \theta_0}{D}$, where θ_D and θ_0 is the phase at time instant D and 0, respectively, and D is the lag and thus the time difference. This is shown in figure 3.13.

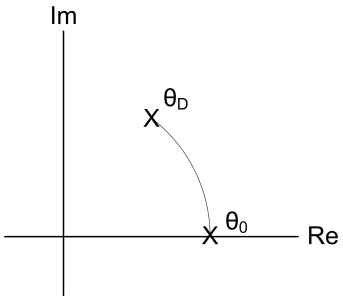


Figure 3.13: The principle used in the frequency offset estimator.

It is shown in [18] that the ratio between the variances for the weighted and unweighted estimators is approximately $N/6$, showing that the improvement in using a window function is substantial for large data records, but this will cause an increase in complexity. The complexity will be discussed further in the later chapter.

The choice of lag, D, depends on the signal to noise ratio (SNR). In general, higher lag gives better performance, but has a higher SNR threshold. SNR threshold under which the phase starts being uncorrelated and uniformly distributed over $[0, 2\pi)$. This means that this threshold has to be exceeded in order to determine the frequency offset. If a frequency estimator is being designed for the low SNR case, the correlation lag D has to be chosen to give a low SNR threshold. This is, in general, achieved for small lags, D [15]. In order to lower the convergence and variance, several estimators with different lags can be put in parallel. This is called a Fitz-estimator [11]. For further details on correlation based frequency offset estimators the reader is referred to [35, 3, 2].

In low-rate systems, where area and power consumption are the main optimization criteria, it is natural to use a Kay or Meyr estimator. It is desirable that the frequency offset estimator works at low SNR. But when operating at low SNR, the estimator is dependent on many samples in order to get a desirably low variance on the estimate.

Because the frequency offset estimators do not require phase unwrapping, they are limited to frequency offsets that obey [18]

$$|\omega D| < \pi, \tag{3.7.4}$$

where $\omega = \Delta\Omega T_S$ is the normalized frequency, where the T_S is the sampling period.

The IEEE 802.15.4 standard requires at least $\pm$ 40 ppm accuracy in both the up and down-conversion. This gives a maximum frequency offset at the receiver side of $\pm$ 80 ppm. With the highest frequency being 2.4835 GHz, the maximum frequency offset is 198.7 KHz. With sampling at a rate 4 Msamples/sec, the normalised frequency ω is approximately 0.312. The maximum D in this case is 10. The higher lag sample autocorrelation functions are less affected by noise and give a better estimate [11], but the SNR threshold increases and this would also require a bigger buffer. It is shown in [25] that the size of the memory has great consequences on the area and current consumption of the DSP architecture. It is therefore chosen to focus on the case where $D = 1$.

3.7.2 Fine Frequency and Phase Offset Estimator

The coarse estimator is first used to find an estimate of the frequency and phase offset. Then, the frequency and phase offset of the samples are corrected and used as the input to the fine frequency and phase offset estimation. This means that the course estimate has to be precise enough for the SNR during one symbol to stay high, in order to get a good final estimate of the frequency and phase offset. In this subsection, two different data-aided algorithms for obtaining a fine frequency and phase estimate will be proposed. It is assumed that the incoming sequence contains a preamble which is known and is meant for synchronization purposes. This is the case for IEEE 802.15.4 2.4 GHz physical layer, as described in chapter 2.

Synchronization

In order for the frequency offset estimator to make use of the known information in the preamble, frame synchronization must first be gained. This can be done by correlating a known sequence

with the incoming sequence. This will give a peak value when the sequences are aligned. The time instant of this peak will give the right frame synchronization.

Algorithms

The correlator performs a complex correlation between the complex signal at the input and the known sequence. The net-rotation per sample is then found by taking the argument of the output correlation for each symbol and divide by the total number of samples.

This estimator is expected to have a much lower variance than the coarse frequency offset estimator, because it uses the information about each complex sample. This means that each complex sample is compared to the wanted complex sample, i.e the reference phase stands still. This is not the case for the blind estimator used in the course frequency and phase offset estimator, since the phase information changes every sample, which adds the signal variance to the estimator. The data aided frequency and phase offset estimator also operates on a higher SNR. Thus, the data aided frequency and phase offset estimator has much higher performance than the blind frequency and phase offset estimator. If the n^{th} input symbol to the correlator is a complex vector

$$\hat{\mathbf{c}}(n) = \hat{\mathbf{c}}_R(n) + j\hat{\mathbf{c}}_I(n) = (\mathbf{c}_R(n) + \mathbf{n}_R(n)) + j(\mathbf{c}_I(n) + \mathbf{n}_I(n)), \tag{3.7.5}$$

where $\mathbf{n}_Q$ and $\mathbf{n}_I$ are the real and imaginary part of the complex noise and $\mathbf{c}_Q$ and $\mathbf{c}_I$ are the real and imaginary part of the complex signal.

Then, the complex correlation for one symbol is then given by

$$\hat{\mathbf{c}}^H(n)\mathbf{c}(n) = \sum_{i=1}^{N} \hat{c}_i(n)^* c_i(n), \tag{3.7.6}$$

where N is the number of chips in a symbol.
Thus, the phase offset of the output of the correlator is given by

$$\theta(n) = \angle(\sum_{i=1}^{N} \hat{c}_i(n)^* c_i(n)). \tag{3.7.7}$$

The random noise added by the channel, analog to digital converter (ADC), CORDIC etc can be approximated by additive white Gaussian noise (AWGN) with a average value of 0. Thus, taking the average phase change between consecutive symbols will give the net-rotation of the constellation. The speed of rotation is given by dividing the rotation by the time. Thus, the fine frequency offset estimator is given by

$$\hat{\omega} = \frac{1}{KN} \sum_{n=1}^{K} \theta(n), \tag{3.7.8}$$

where K is the number of symbols the average is taken over and N is the number of chips in each symbol. This gives two candidates for the fine frequency offset estimator.

Candidate 1: Taking the Argument at the End of each Symbol. This algorithm is obtained by directly using the result above

$$\hat{\omega} = \frac{1}{KN} \sum_{n=1}^{K} \left(\angle \sum_{i=1}^{N} \hat{c}_i^*(n) c_i(n) \right). \tag{3.7.9}$$

Candidate 2: Taking the Argument First. Because the estimator operates on high SNR, the noise vector only contribute with a phase change. Thus, the angle and summation operation can be interchanged as shown in [18], and 3.7.7 can be rewritten as

$$\theta(n) = \sum_{i=1}^{N} \angle(\hat{c}_i(n)^* c_i(n)) \tag{3.7.10}$$

$$= \sum_{i=1}^{N} \angle c_i(n) - \angle \hat{c}_i(n) \tag{3.7.11}$$

$$= \sum_{i=1}^{N} \theta_i(n) - \hat{\theta}_i(n). \tag{3.7.12}$$

Thus, the proposed frequency offset estimator is

$$\hat{\omega} = \frac{1}{KN} \sum_{n=1}^{K} \left(\sum_{i=1}^{N} \theta_i(n) - \hat{\theta}_i(n) \right). \tag{3.7.13}$$

3.7.3 Phase Offset

The total phase offset at discrete time instant n is $\theta[n] = \theta_0 + \omega n$, where θ_0 is the initial phase offset and ω is the frequency offset. The initial phase offset of the complex signal is found by taking the argument at the time instant before the compensator starts rotating the samples.

3.7.4 Complex Rotation

The time-varying phase offset described in the previous subsection needs to be compensated for. This is done by rotating the samples back to their estimated original position. This operation is given by

$$Q(n) = m(n)e^{-j(\hat{\omega}n+\hat{\theta_0})} = (S_I(n) + jS_Q(n))e^{j((\omega-\hat{\omega})n+(\theta_0-\hat{\theta}))} = (S_I(n) + jS_Q(n))e^{j((\Delta\omega)n+(\Delta\theta))}, \tag{3.7.14}$$

where S_I(n) S_I(n) are the real and imaginary samples, respectively. $\hat{\omega}$ and $\hat{\theta}$ is the estimated frequency and phase and ω and θ the actual frequency and phase. $\Delta\omega$ and $\Delta\theta$ are the frequency and phase offset after compensation, which ideally are 0. This is seldom the case since the estimators have non-zero variance. The constellation for the ideal situation, where there is no frequency and phase offset after compensation, is shown in figure 3.14. The constellation before and after compensation is shown in gray and black, respectively.

The complex rotation can be implemented by using the CORDIC algorithm in the rotation mode or alternatively perform a complex multiplication.

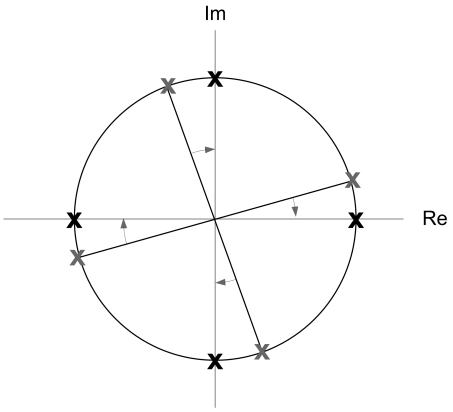


Figure 3.14: Complex rotation

3.7.5 Realization of the Frequency and Phase Offset Compensator

This subsection will present some possible realizations of the frequency and phase offset estimators.

The Coarse Frequency offset Estimator

A direct implementation of the unweighted Kay estimator is shown in figure 3.15. The CORDIC in vectoring mode is used to find the argument and the CORDIC in rotary mode is used for the complex rotation. A phase accumulator is used to give the phase at time instant n, $\theta[n] = \theta_0 + \omega n$. The accumulator is initialized by θ_0. This value will be increased by ω for every sample. The output of the phase accumulator will thus give the total phase offset at discrete time instant n. This value is put into the CORDIC to update the phase rotation. How often this updating is done, depends on the size of the frequency offset and the designers wishes concerning the performance. The actual computational savings depends on the architecture.

The algorithm for the unweighted Kay estimator in equation 3.7.1, where $D = 1$ can equivalently be written as

$$\hat{\omega}_0 = \frac{1}{N-1}\sum_{t=0}^{N-2}\angle(x_t^* x_{t+1}) = \frac{1}{N-1}\sum_{t=0}^{N-2}\angle(x_{t+1}) - \angle(x_t), \tag{3.7.15}$$

where N the number of samples over which the estimate is taken, i.e the length of the in the incoming signal, x_t. This can be implemented as shown in figure 3.16.

The Meyr estimator can be realized the same way as the Kay estimator, but the argument is taken after the averaging. Thus, the CORDIC (vectoring) block has to be moved at the output of the frequency offset estimator, i.e in front of the phase accumulator. This excludes the alternative

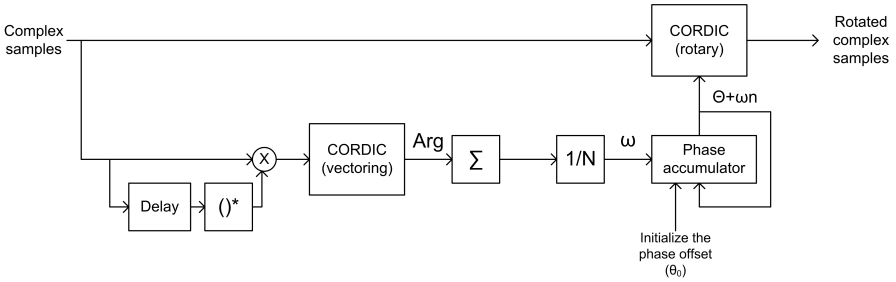


Figure 3.15: Direct realization of the unweighted Kay estimator.

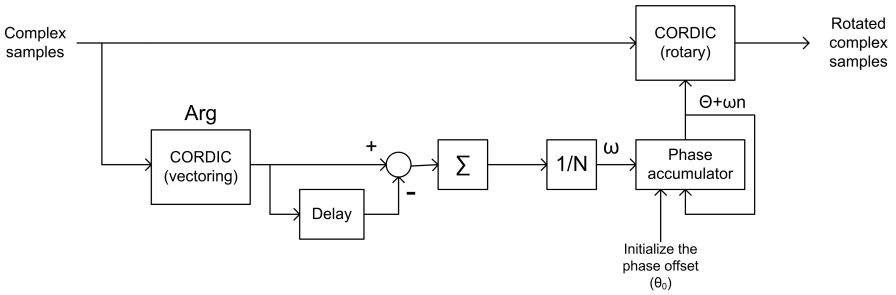


Figure 3.16: Alternative realization of the unweighted Kay estimator.

realization for the Meyr estimator as the one shown in figure 3.16 for the Kay estimator. A realization of the generalized unweighted Meyr estimator is shown in figure 3.17.

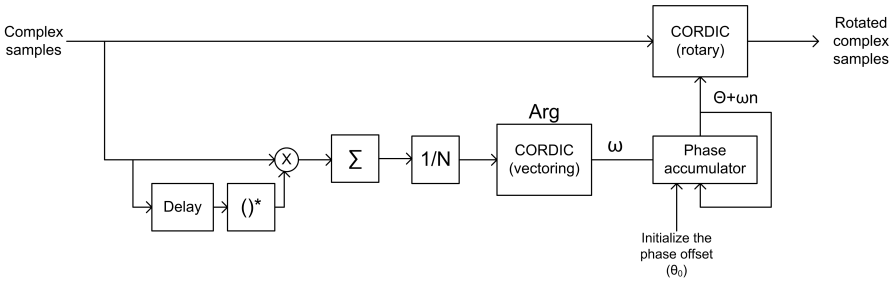


Figure 3.17: Direct realization of the generalized unweighted Meyr estimator.

Fine Frequency offset estimator

The fine frequency candidate 1 described in section 3.7.2 can be realized as shown in figure 3.18. $C_I + N_I$ and $C_Q + N_Q$ are the real and imaginary parts of the noise influenced signal, respectively. A is the correlation between $\hat{I}$ and I, B is the correlation between $\hat{Q}$ and Q, C is the correlation

between $\hat{I}$ and Q and D is the correlation between $\hat{Q}$ and I. Input X to the CORDIC is the real part of the signal and input Y is the imaginary part, and the output Z is the arctan value. The $\sum_i$ block accumulates over a symbol (16 complex chips for the IEEE 802.15.4 2.4 GHz PHY). The last summation block accumulates total phase offset and the last block divided the total phase offset by the numbers of samples. This gives the estimated increment in phase offset per sample, i.e the estimated frequency offset, $\hat{\omega}$.

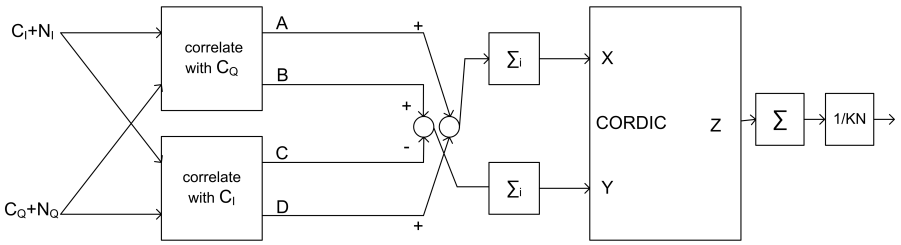


Figure 3.18: A realization of the fine frequency offset estimator where the argument is taken at the end.

Candidate 2 can be realized as shown in figure 3.19.

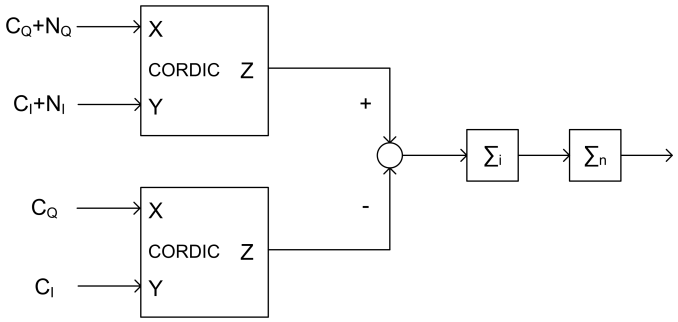


Figure 3.19: A realization of the fine frequency offset estimator where the argument is taken before the correlator.

Phase offset

The phase at a time instant is given by the arctan value of the complex sample. Since the information in the signal is in the phase, the signal variance makes a phase offset estimate unreliable, in addition to the noise. The phase offset can therefore be found by averaging the arctan value of the output of the complex rotation. After obtaining an estimate of the phase offset, the the phase accumulator in the frequency offset compensator is initialized. Then, the frequency and timing offset compensator starts rotating the incoming samples. The fine frequency offset estimator at the correlator will then find a more exact estimate of the frequency and the arctan value of the

last fed back to the phase accumulator to initialize the phase. The latency must in this case be taken into account.

3.8 Matched Filter

The matched filter is the optimum system for detecting a known signal in additive Gaussian noise [14]. The objective of the matched filter is to maximize the signal noise ratio. This is done by correlating the signal with itself, i.e convolving it with a time-reversed version of the pulse shape. Ideally, the matched filter should be matched to the signal at the input of the filter without noise. This would require the coefficient to change according to the changes in the channel, which would require higher complexity. The matched filter is therefore chosen to match the pulse shape from the transmitter.

The IEEE 802.15.4 2.4GHz-band PHY is required to use a half-sine pulse shape given by

$$p(t) = \begin{cases} \sin(\pi \frac{t}{2T_c}) & ,0 \leq t \leq 2T_c \\ 0 & ,\text{otherwise}, \end{cases} \tag{3.8.1}$$

where $2T_C$ is one chip period.

The coefficients of the matched filter, if a 4 MHz sampling rate, i.e 2 times oversampling, is assumed :

$$h(nT_s) = h_n = \begin{cases} \sin(\pi - \pi \frac{n}{4}) & ,n = 0,1,2,3 \\ 0 & ,\text{otherwise}, \end{cases} \tag{3.8.2}$$

which gives the filter coefficients $h_1 = 0$, $h_2 = \frac{1}{\sqrt{2}}$, $h_3 = 1$, $h_4 = \frac{1}{\sqrt{2}}$. Figure 3.20 shows the received signal before and after the matched filter after passing through a additive white Gaussian noise (AWGN) channel with a signal to noise ratio (SNR) of 4 dB. It can be seen that the constellation is wider for the matched filter case. The spreading of the received signal points is approximately the same, thus the SNR has been improved.

3.8.1 Realization of the Matched Filter

The 3 candidates for realization of the matched filter are all designed as FIR filters and are decribed in the following.

Candidate 1: Direct form The direct form FIR filter implementation of the matched filter is a straight forward implementation of the convolution between the incoming sequence and the matched filter coefficients. This is shown in figure 3.21.

Candidate 2: Transposed form The transposed FIR filter is found by reversing all paths, then to replace the pick-off nodes by adders, and vice versa, and interchange the input and output nodes.

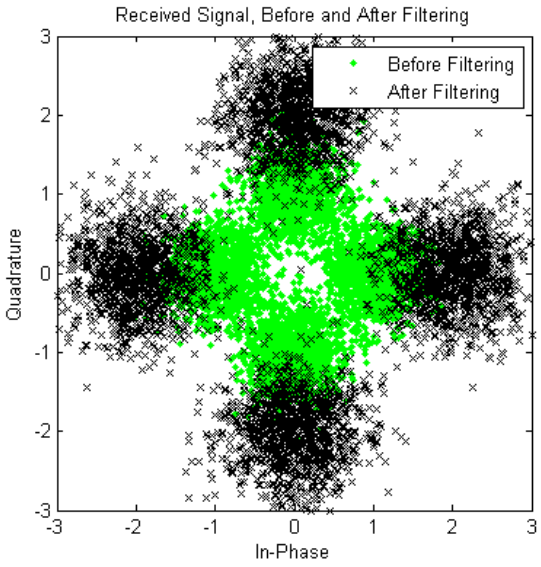


Figure 3.20: The received signal before and after the matched filter.

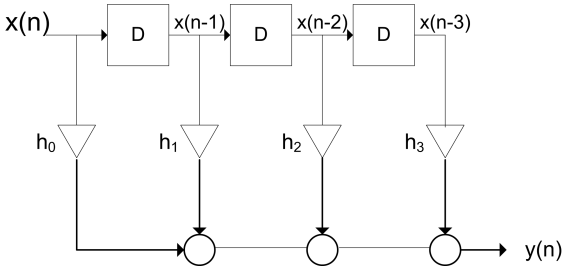


Figure 3.21: Direct form implementation of the matched filter.

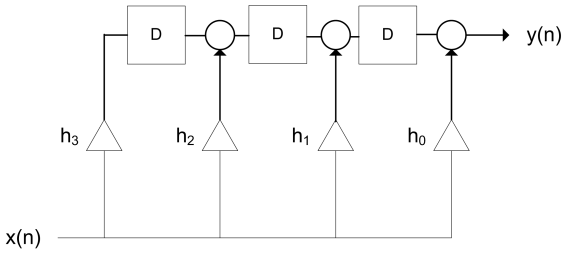


Figure 3.22: Transposed form implementation of the matched filter.

Candidate 3: Exploiting the symmetry In the matched filter, two of the coefficients are similar and one coefficient is zero. This is exploited in the structure shown in figure 3.23

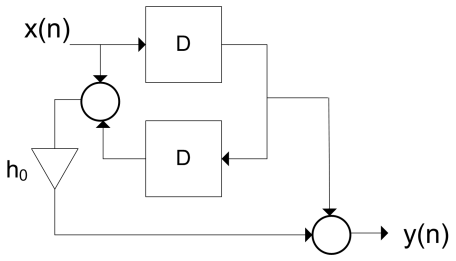


Figure 3.23: Exploiting the symmetry of the matched filter

3.9 Correlator

This section will start by presenting some background theory on the detection method and the assumptions made. Then some different realizations will be presented.

As described in chapter 2, the first 5 bytes received in a packet is the synchronization header, which consists of a preamble and start-of-frame delimiter (SFD). The detector uses the preamble and SFD for correcting the frequency and phase offset and to find the right frame synchronization, respectively. The correlator operates therefore in two modes. In mode 1, the coarse frequency offset estimator uses the first symbols of the preamble to obtain an estimate of the frequency and phase offset. When this estimate is done, the correlator starts looking for a 0-symbol and a detection indicates the right timing. It is then possible to make the fine frequency estimation. When this is done, the correlator starts looking for the start-of-frame delimiter, which gives the right frame synchronization and tells the correlator to switch to the second mode. The second mode is when the actual payload is being received.

There will also be a short presentation of direct-sequence spread spectrum (DSSS) and block coding, since the gain from using these techniques appears at the correlator. These techniques effect on the performance of the system will be presented later in this Thesis.

3.9.1 Detection

The two most used detection methods are maximum likelihood (ML) and the maximum a-posteriori (MAP). The MAP detector is optimal in the sense that it minimizes the probability of error. Thus, it has a lower probability of error than the ML detector, but it is shown in [5] that MAP reduces to ML in the case of equal probability for all symbols, which is the case for the IEEE 802.15.4 physical layer. If there is no information about the probability of the different symbols, equal probability is usually assumed. The channel is assumed to be AWGN, so the ML detection is equivalent to a minimum mean square detection [5]. The received discrete-time signal can be describes as

$$Y_k = s_k^{(a)} + N_k \quad , 0 \leq k \leq L, \tag{3.9.1}$$

where $s_k^{(a)}$ is the a^{th} known signal of the set consisting of M, N_k is the zero mean AWGN and L is the number of components in the vector.

The ML detector calculates the euclidean distance in the signal space between the observed signal and the known signal, and then finds the smallest. Thus, it calculates [5, page 297]

$$J_a = \sum_{k=0}^{L} |Y_k - s_k^{(a)}|, \tag{3.9.2}$$

where Y_k is the kth sample of the received signal, $s_k^{(a)}$ is the kth sample of the ath known sequence. The detector chooses the a for which J_a is at its minimum. This is equivalent to maximizing

$$R_a = \mathrm{Re}\{\sum_{k=0}^{L} Y_k s_k^{(a)}\} - \frac{1}{2}E_a, \tag{3.9.3}$$

where E_a is the energy of the ath signal and R_a is the correlation value between the incoming signal and the ath known signal. If the signal is represented using antipodal signaling and the same length for all the code words, the energy will be the same for all R_a, and E_a can therefore be disregarded when maximizing R_a. It is therefore sufficient to maximize

$$R_a = \mathrm{Re}\{\sum_{k=0}^{L} Y_k s_k^{(a)}\}, \tag{3.9.4}$$

which is the discrete time correlation receiver. A geometric representation of the presented simplification is shown is figure 3.24. It shows that finding the point in space with the shortest euclidean distance is equivalent to finding the vector in space which is the most correlated. The correlation receiver is shown in figure 3.25. The observation vector, $\mathbf{x}$ is correlated with the M different signal vectors, $\mathbf{s}_i$. The estimated received signal, $\hat{a}$, is given by the largest accumulated value, i.e the largest inner product.

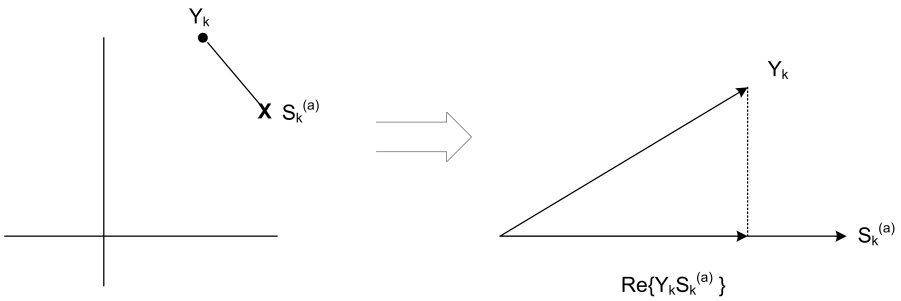


Figure 3.24: The Simplification of the ML-detector.

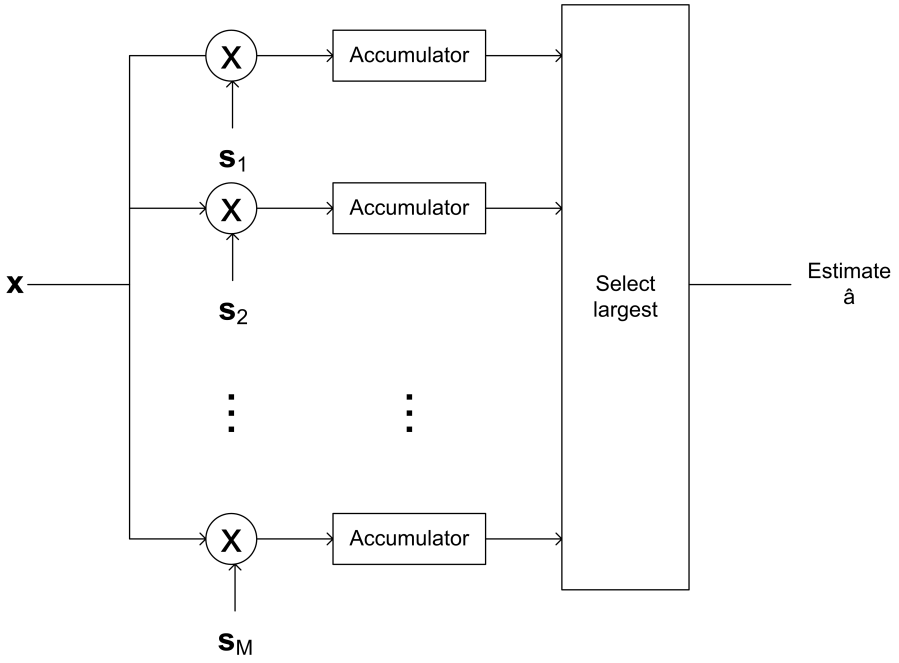


Figure 3.25: The correlation receiver.

The correlator can equivalently be seen as using M discrete filters which are matched to the M different known signal.

This chapter will describe different realizations of the correlation receiver, all based on using the described correlation receiver. Thus, they will all be based on the optimal ML detector shown in figure 3.25. The difference in realizations is the ways of choosing the largest value out from the accumulators and what strategy to use in order to utilize the properties of the I and Q information.

Since the IEEE 802.15.4 2.4 physical layer standard defines the modulation to be O-QPSK, the information is contained in both the I and Q carrier. These are chosen to be sent trough two separate branches, since this helps utilizing the properties of the chip sequences described in chapter 2.1.2. The realizations are all based on using 4 samples per chip, which is used to find the correct timing as described in the sequel. The frequency and phase offset is found during the preamble and the frame synchronization is found by using the SFD. Then, the receiver starts decoding the actual information, i.e the payload.

3.9.2 Timing

Because of the frequency offset, there will be a change in the amplitude for the different chips, since the sampling is not taken at the highest point. By oversampling the chips and correlating them with the same sequence at all the paths, the highest correlation value is the one taken at the most ideal sampling instant. This means that the correlation value for each chip is given

by the biggest of the 4 correlation values. In order to reduce the complexity of the search for the biggest value, an early-late detection algorithm can be used [5, 19]. This algorithm is based on the principle that by comparing the sample before and after the ideal sampling instant. The deviation can then be used to decide if and what way the ideal sampling instant should be moved. This algorithm is dependent of a symmetric signal shape. The deviation is given by

$$e_n = (y_n - y_{n-2}), \tag{3.9.5}$$

where y_{n-2} and y_n are, respectively, the sampling instants before and after the ideal sampling instant as shown in figure 3.26. A positive error means that the timing is early and a negative error means that the timing is late. When the error exceeds a predefined threshold, the timing must be changed to equalize the error as much as possible, as seen in figure 3.26. Thus, the early-late detection algorithm adjusts the timing instant so that the early sample y_{n-2} is as similar to the late sample y_n as possible.

A plot of the variance of the delay versus SNR by using the early-late detection algorithm is shown in [16]. And examples of two other algorithms for timing recovery are Mueller and Muller algorithm [23] and Gardner Algorithm [13]. These are not described here because both are more suitable for signals where there transitions around zero occur regularly. This is not the case for IEEE 802.15.4 2.4 GHz PHY.

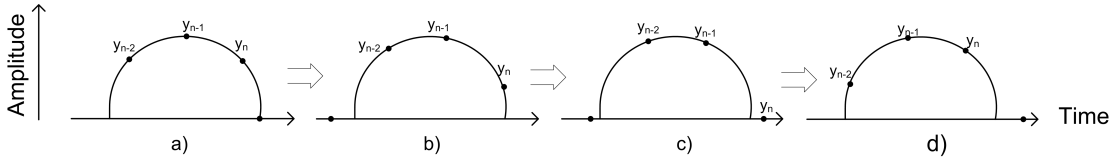


Figure 3.26: Timing adjustment using early-late detection

3.9.3 Realization of the Correlator for Reception of Preamble and SFD

A block diagram of the reception during the preamble and SFD is shown in figure 3.27. First, the right frame synchronization must be found in order to make use of the fact that the preamble is known. This is done by correlating the incoming sequence with the 0-symbol and detecting the peak. The fine frequency offset estimator in now ready to receive. The realizations of the estimator is shown in subsection 3.7.5 . This estimator uses K symbols to obtain an estimate of the frequency and pases the the average phase and frequency to the phase accumulator to update the parameters. $Delta\omega$ indicates the estimated frequency offset and $Theta_0$ the estimated phase offset. It is up to the designer to decide how ofter this update should be done, but it should in general be done more for systems working on low SNR, since this increases the uncertainty of the estimators. When the parameters have been updated in the compensator, the correlator starts looking for the start-of-frame delimiter (SFD). This can be done by correlating the incoming sequence with the SFD and detecting the peak.

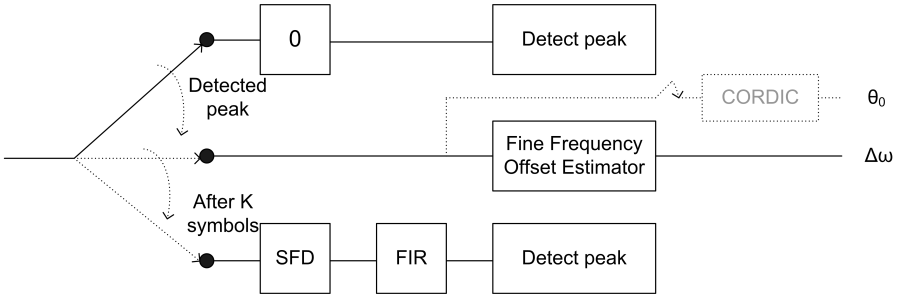


Figure 3.27: Reception During the Preamble and SDF.

3.9.4 Realization of the Correlator for Reception of Payload

This subsection will first discuss the general form of the correlator, then strategies for utilizing the qualities of the I- and Q-part of the chips and algorithms for finding the most correlated sequence. Different algorithms for finding the best timing instant were discussed under the previous subsection.

There are several different options for the general form of the correlator. The two main options are whether the matched filter and the correlator should be two separate units or not. If the matched filtering is done first, the precision can be lowered before the correlation. It also enables the correlator to be implemented as a FIR filter where all the coefficients are ± 1, which in effect is a accumulation. Using a joint unit for the matched filter and correlator also increases the latency. Thus, this thesis will only have focus on the case where the matched filter and the correlator are seperate units.

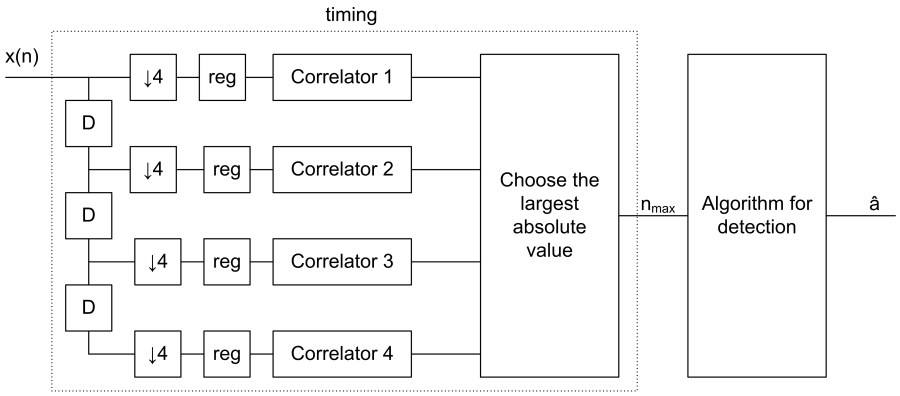


Figure 3.28: Proposed correlator.

The simplified block diagram of the correlator is shown in figure 3.28. This first part takes care of the timing, $n_{\max}$ is the largest correlator value and $\hat{a}$ is the detected symbol.

The IEEE 802.15.4 2.4 GHz physical layer uses 16 quasi-orthogonal symbols, each of length

32 chips. This means that, ideally, 16 different correlators of length 32 should be used to detect the received signal. It is described in chapter 2.1 that symbol number i, for $i = [0,7]$ is created by shifting the 0-symbol $i \times 4$ times. Symbol $i+8$ is created the same way, only inverting every other bit. The even indexed bits are modulated onto the I-carrier and the odd indexed bits are modulated onto the Q-carrier. The index starts at 0. It is therefore natural keep the I- and Q-information divided in separate branches.

In order to get an unambiguous detection of a symbol, both branches or only the Q-branch has to be detected. A detection in the I-branch can only decide for symbol i or $i+8$, since the first 8 symbols are similar to the last 8. But by using the Q-path for detection, there are 16 unique symbols for detection, where the last 8 symbols are inverted versions of the first 8. Thus, a detection in only the Q branch is sufficient to decide on which symbol most probably were sent. These characteristic leads to different options for strategies for utilizing this in different ways. It will in the sequel be proposed 3 different strategies.

Strategy 1: Correlate for all the values in both the I- and Q-branch This is the most robust and intuitively satisfying solution. A detection is made on the basis of the sum of the 8 different I-branch correlation values and the 16 different Q-branch values. Thus, a direct realization of the ML-detector in figure 3.25 with $M = 8$ for the I-branch and $M = 16$ for the Q-branch. Then the I-branch is added together with the corresponding Q-branch correlation values and a detection is made on the basis of this value.

Strategy 2: Correlate for all the values in the I-branch and then check the sign of the Q-branch After a detection is made in the I-branch correlator, the incoming sequence at the Q-branch is correlated with the corresponding symbol in the Q-branch correlator. Since the 8 last symbols in the Q-branch are inverted versions of the first 8, and the 8 first symbols in the I-branch are similar to the last 8, the sign at the output of the correlator gives the detected symbol. Thus, symbol i or i+8 is first detected by making an exhaustive search through the 8 correlation values in the I-branch and then correlating symbol i in the q-branch. Positive sign at the output of the correlator means a detection on symbol i, while a negative sign means a detection on symbol i+8. This is shown in figure 3.29

Strategy 3: Correlate for all the values in the Q-branch Since there are 16 unique symbols for the Q-branch, it is possible to make a detection on a symbol based on only the Q-branch values. This is a direct realization of the ML-detector in figure 3.25 with $M = 16$.

Once a appropriate strategy is found, the designer has to decide on how the detection should be made, i.e detection the signal vector which is the most correlated with the observation vector in figure. This is done, as seen from figure 3.25, by choosing the largest correlation value. It will now be proposed 3 different candidates for how this detection can be done.

Candidate 1: Find the largest The most robust implementation is to make an exhaustive search to find the largest correlation value. This is the straight forward implementation of the correlation receiver; all the 8 correlation values, R_{m}, where m=[1,8], are stored and compared to find the largest. Thus, this is a robust, but computationally demanding candidate for detection.

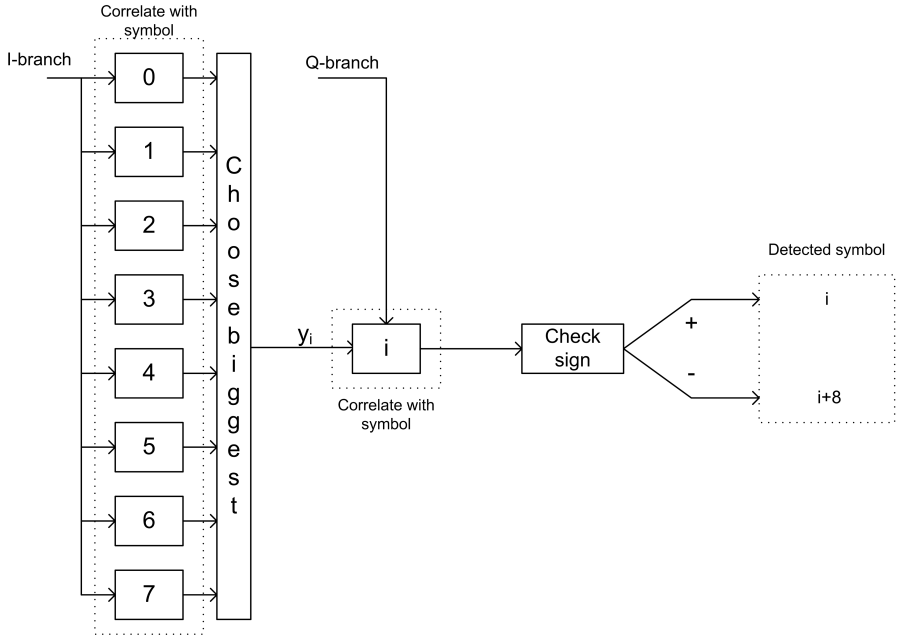


Figure 3.29: Block diagram of the correlator using strategy 2 .

Candidate 2: One threshold The correlations are done sequentially until one of the correlation values exceeds the defined threshold. This means that each correlator in general only needs to do 4,5 correlations in average, if a equal probability for doing 1 to 8 correlations is assumed. The main problem is to find a suitable threshold, because a too low threshold will cause a detection too easily, whereas a too high threshold will cause the correlator to make no detection. In this case, a default value must be set as output if there is made no detection. The noise variance will in general vary, so it is hard to set the right threshold. Thus, this candidate has lower computational requirements than candidate 1, but is also less robust.

Candidate 3: Combinations of threshold and finding the largest As a compromise between robustness and complexity , the detection can be made as a combination of candidate 1 and 2. In addition to the threshold, the values can be stored, so they can be used to find the largest correlation value if the threshold is not exceeded. This gives the flexibility between robustness and complexity; if the threshold is set high, it is never exceeded and therefore 8 correlation values must be computed. If the threshold is set low, noise can easily cause values to exceed the threshold, and thus make wrong detections, but the complexity will be low.

3.9.5 Direct Sequence Spread Spectrum (DSSS) and Forward Error Correction (FEC)

The direct sequence spread spectrum (DSSS) is a technique in where the original signal is spread over a bandwidth much bigger than the bandwidth required by the Nyquist criterion. The processing gain (PG) is a measure of how much the signal-to-noise ratio (SNR) can be lowered and still achieve a certain bit error rate (BER). It is shown in [14, page 488-497] that the processing gain for DSSS is

$$PG = \frac{R_{\mathrm{C}}}{R_{\mathrm{b}}}, \tag{3.9.6}$$

where R_{C} is the chip rate and R_{b} is the bitrate.

The direct sequence spread spectrum (DSSS) technique used in the IEEE 802.15.4 standard is mapping from source symbols to channel symbols, and can therefore also be seen as a forward error correction (FEC) block channel code. The objective of a channel code is to increase the minimum hamming distance, $d_{\min}$ between the symbols and thus be able to correct $m = \lfloor (d_{\min} - 1)/2 \rfloor$ error. Thus, the BER for a certain SNR goes down. The amount the SNR can be lowered in order to get the same BER as in the uncoded case is called the code gain.

3.10 Link Quality Indicator

The link quality indication (LQI) measurement is a characterization of the strength and/or quality of a received packet. The 802.15.4 standard requires the the LQI number to be represented by a 8 bit integer, ranging from 0 through 255, in which at least 8 unique values must be defined.

3.10.1 Realization

$$LQI = \frac{1}{N} \sum_{n=0}^{N} R_n + A, \tag{3.10.1}$$

where N is the number of samples over which the LQI is found, R_n is the correlation value of sample at discrete time n and A is value which moves the average correlation from range $[-128, 127]$ to $[0, 255]$.

The LGI can also be implemented by using the RSSI value directly or together with the correlation value. Using the RSSI value directly to estimate the LQI value has the disadvantage of not giving any info about the actual quality, only the strength of the incoming signal. In CC2420[8, page 50], a semi-empirical model is used, where the correlation values of the 8 first symbols after the SFD is used as an indication of the 'chip error rate'.

Chapter 4

Transmitter

This chapter will give a brief presentation of a possible implementation of a IEEE 802.15.4 2.4 GHz physical layer transmitter. The proposed transmitter is shown in figure 4.1.

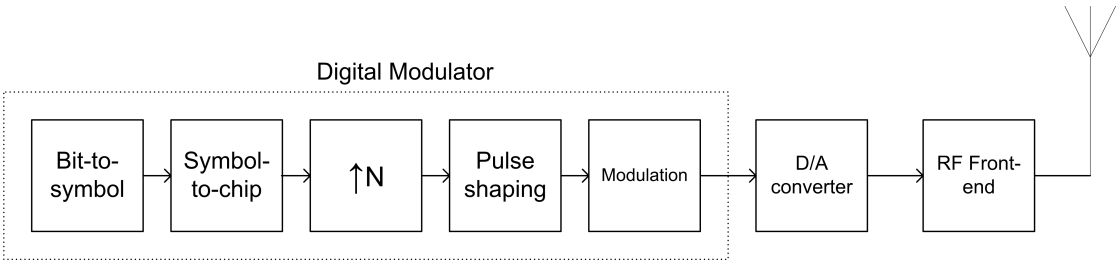


Figure 4.1: A simplified block diagram of the transmitter.

4.1 Bit-to-symbol Mapping

The bit-to-symbol mapping is done by grouping 4 bits together. Giving 16 possible symbols.

4.2 Symbol-to-Chip Mapping

The symbol-to-chip mapping is done in separate I and Q branches. The chip sequence for symbol 0 is stored in memory, and the next 7 symbols are given by cyclic shifts of this sequence. The last 8 values for the I branch is found the same way, whereas the last 8 values for the Q branch is found the same way as well, only inverting the bits.

4.3 Pulse Shaping

The pulse shaping used in the IEEE 802.15.4 2.4 GHz physical layer, as described earlier, is half sine. The pulse shaping is done by first upsampling the sequence by the number of taps in the

pulse shaping filter and then convolving it with the digital pulse shaping filter. The pulse shaping filter is the same as the matched filter and can be implemented the same way.

4.4 Modulation

The signal is divided into I and Q values already in the symbol-to-chip mapper. Since the IEEE 802.15.4 2.4 GHz physical layer standard defines a O-QPSK modulation, where the Q values are one half chip period delayed compared to the I. This can be done by stuffing $N/2$ zeros at the beginning of the Q-sequence and at the end of the I-sequence.

4.5 Radio Frequency Front-end

The radio frequency front-end is assumed to be the same as for the receiver, but all the components are flipped. The direct conversion transmitter is shown in figure 4.2. The digital signal is converted to an analog form by the digital to analog converter (DAC), then passed trough a low-pass filter to smooth out the effects of the digital-to-analog conversion, converted up on the carrier frequency, amplified and sent out on the antenna.

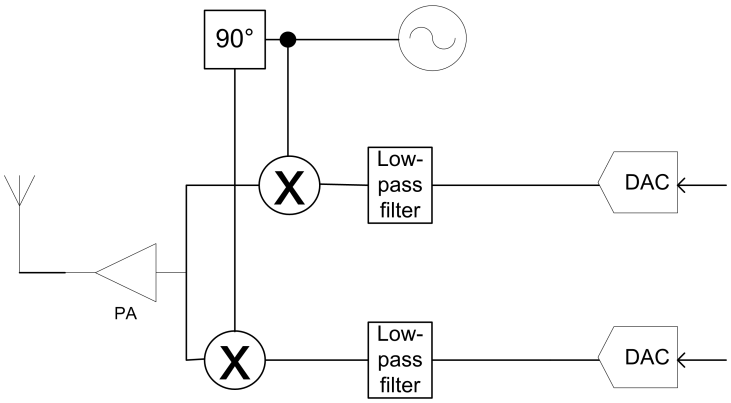


Figure 4.2: Direct conversion Transmitter.

4.6 Realization of the Modulator

The main trade-off in the design of the modulator is between memory and complexity/current consumption. The whole modulator can be implemented as a look-up table, which minimizes the complexity and thus the current consumption. The modulator can conversely be implemented directly as seen from figure 4.1. The only look-up table is the symbol-to-chip mapper. Then, the signal is upsampled by a factor N and filtered by the N-tap FIR filter. At last, the offset between the two carriers is created by stuffing $N/2$ zeros at the beginning of the Q-sequence and at the

end of the I-sequence. This implementation has higher computational requirements, but requires less memory.

Chapter 5

Complexity

Chapter 3 and 4 presented various algorithms for realizing a SDR modem. This chapter will present a summary of complexity of the different functional blocks in the modem. The complexity was considered by the number of additions, complex and real multiplications, multiplications by constants, complex conjugations etc. The real savings in complexity depends on the architecture and was investigated in [25]. Some of the results of this is summarized in chapter 6.

5.1 Demodulatior

The complexity of the demodulator shown in figure 3.1 will in this section be summarized, block by block, starting left.

5.1.1 Channel Filter

The candidates described in section 3.4 were:

- **Candidate 1:** Direct form realization (figure 3.7)
- **Candidate 2:** Transposed form realization (figure 3.8)
- **Candidate 3.1:** Symmetric realization type 1 linear phase
- **Candidate 3.2:** Symmetric realization type 2 linear phase (figure 3.9)

The complexity of the different realizations of a M^{th} order FIR filter with a incoming sequence of N samples is summarized in table 5.1.

Candidate 3 is divided into two candidates for the general case since type 1 and type 2 linear phase FIR filters have different computational requirements. Candidate 3 has clearly lower complexity than the other two, since the number of multiplications is approximately half of the two others.

Candidate	**1**	**2**	**3.1**	**3.2**
Additions	$M \times N$	$M \times N$	$M+1 \times N$	$M \times N$
Delays	$M \times N$	$M \times N$	$M+1 \times N$	$M \times N$
Multiplications	$M+1 \times N$	$M+1 \times N$	$(\frac{M}{2}+1) \times N$	$(\frac{M}{2}+1) \times N$

Table 5.1: Complexity of the different channel filter realizations.

5.1.2 Down Sampler

The down sampler reduces the number of samples per second, thus also the total number of samples which needs to be processed.

5.1.3 Received Signal Strength Indicator

The two candidates described in section 3.6 were:
Candidate 1:

$$RSSI_1 = \sum_{n=0}^{N-1} s_I^2(n) + s_Q^2(n). \tag{5.1.1}$$

Candidate 2:

$$RSSI_2 = \frac{1}{N} \sum_{n=0}^{N-1} s_I^2(n) + s_Q^2(n). \tag{5.1.2}$$

The complexity of these two realizations of the received signal strength indicator is summarized in table 5.2.

Candidate	**1**	**2**
Additions	2N	2N
Multiplications	2N	2N
Multiplications by a constant	0	1

Table 5.2: Complexity of the RSSI for different realizations.

It is also possible to find the absolute value by using the COordinate Rotation DIgital Computer (CORDIC) algorithm in vectoring mode and then output the real value. This is generally less effective than the straight forward implementation analyzed here, since the absolute value has to be squared, but can be used if the work load on the multiply-and-accumulate unit in the DSP is much higher than for the CORDIC. The complexity of this realization depends on the implementation of the CORDIC algorithm.

5.1.4 Frequency and Phase Offset compensator

The frequency offset compensator consists of the fine and coarse frequency and phase offset estimator and the complex rotation.

Coarse Frequency Offset Estimator

This section will sum the complexity of the coarse frequency offset estimators described in section 3.7. For reasons explained in chapter 3, D is set to be 1.
Candidate 1: the weighted Kay estimator

$$\hat{\omega}_0 = \sum_{t=0}^{N-2} \angle(w_t x_t^* x_{t+1}). \tag{5.1.3}$$

Candidate 2: the weighted Meyr estimator

$$\hat{\omega}_0 = \angle(\sum_{t=0}^{N-2} w_t x_t^* x_{t+1}). \tag{5.1.4}$$

where

$$w_t = \frac{\frac{3}{2}N}{N^2-1}\{1-[\frac{t-(\frac{N}{2}-1)}{\frac{N}{2}}]^2\}. \tag{5.1.5}$$

Candidate 3: the unweighted Kay estimator

$$\hat{\omega}_0 = \frac{1}{N-1}\sum_{t=0}^{N-2} \angle(x_t^* x_{t+1}). \tag{5.1.6}$$

Candidate 4: the unweighted Meyr estimaror

$$\hat{\omega}_0 = \angle(\frac{1}{N-1}\sum_{t=0}^{N-2} x_t^* x_{t+1}). \tag{5.1.7}$$

Candidate 5: the simplified unweighted Kay estimator

$$\hat{\omega}_0 = \frac{1}{N-1}\sum_{t=0}^{N-2} \angle(x_{t+1}) - \angle(x_t), \tag{5.1.8}$$

where N is the number of samples used to get an estimate of the frequency offset.

The complexity of these 5 realizations of the coarse frequency offset estimators is summarized in table 5.3.

It can be seen from table 5.3 that the two weighted estimators has much higher complexity than the two unweighted estimators. And also that the simplified unweighted Kay estimator has much lower complexity than all the other estimators, because it has substituted the complex conjugation and multiplication by subtractions.

Candidate	**1**	**2**	**3**	**4**	**5**
Additions	N-1	N-1	N-1	N-1	N-1
Complex conjugates	N-1	N-1	N-1	N-1	0
Complex multiplications	N-1	N-1	N-1	N-1	0
CORDIC operations	N-1	1	N-1	1	N
Multiplications by a constant	$N-1$	$N-1$	1	1	1
squares	N	N	0	0	0
Subtractions	4N	4N	0	0	N-1

Table 5.3: Complexity of the different coarse frequency offset estimator realizations

Fine Frequency Offset Estimator

This section will sum the complexity of the fine frequency offset estimators described in section 3.7. The length of each symbol is N and the number of symbols the average is taken over is K.

Candidate 1: Taking the Argument at the End.

$$\hat{\omega} = \sum_{n=1}^{K} \left(\angle \sum_{i=0}^{N} \hat{C}_i^*(n) C_i(n) \right). \tag{5.1.9}$$

Candidate 2: Taking the Argument First.

$$\hat{\omega} = \sum_{n=1}^{K} \left(\sum_{i=1}^{N} \theta_i(n) - \hat{\theta}_i(n) \right), \tag{5.1.10}$$

where K is the number of symbols the average is taken over and N is the number of samples per symbol.

The complexity of these 2 realizations of the fine frequency offset estimators is summarized in table 5.4.

Candidate	**1**	**2**
Additions	$N \times K$	$N \times K$
Complex conjugates	$N \times K$	0
Complex multiplications	$N \times K$	0
CORDIC	K	$2N \times K$
subtractions	0	$N \times K$

Table 5.4: Complexity of the differents fine frequency offset estimator realizations.

It can be seen from table 5.4 that the $N \times K$ complex multiplications for candidate 2 has been substituted by increase of factor $2N$ in the number of CORDIC operations and $N \times K$ subtractions. Thus, candidate 2 has lower complexity than candidate 1.

Phase Offset

For the coarse frequency and phase offset estimator, the phase offset estimate needs to accumulate the output of the compensator to eliminate some of the signal phase variance. Then, a CORDIC operation is needed to find the phase offset.

For the fine frequency and phase offset estimator, the reference points stands still and it is therefore sufficient to use the phase offset of the average over a low number of outputs, maybe as low as 1 since the SNR is quite high at the correlator. For candidate 2 for the fine frequency offset estimator, the the phase of the last incoming sample is already found and can therefore be used directly to initialize the phase at the phase accumulator. If one of the other realizations is used, the phase of the last sample used in the estimate has to be found and will therefore add 1 CORDIC operation to the complexity.

Complex Rotation

The complex rotation is done by the CORDIC and the complexity will therefore depend on the implementation of the CORDIC. The complex rotation must by done one time per sample after the coarse frequency and phase offset estimation is done.

5.1.5 Matched Filter

The time-discrete matched filter with 2 times oversampling will consist of 4 coefficients. This can be implemented as a 3^{rd} order FIR filter. The candidates described in section 3.8 were:

- **Candidate 1:** Direct form realization (figure 3.21)
- **Candidate 2:** Transposed form realization (figure 3.22)
- **Candidate 3:** Symmetric realization (figure 3.23)

The complexity of the different realizations of the 3^{rd} order FIR filter with N samples input sequence is summarized in table 5.5.

Candidate	**1**	**2**	**3**
Additions	3N	3N	2N
Delays	3N	3N	3N
Multiplications	4N	4N	1N

Table 5.5: Complexity of the matched filter for different realizations.

5.1.6 Quantizer

The complexity of the quantization depends on the architecture. Disregarding the complexity of the actual quantization, a quantization by L times reduces the complexity L times. For example going from 8 bits to 2 bits will reduce the complexity 4 times on all the subsequent blocks.

5.1.7 Correlator During Preamble and Start-of-Frame Delimiter (SFD)

How the correlator operates during the Preamble and start-of-frame delimiter (SFD), i.e mode 1 is shown in figure 3.27. The 0-symbol and SFD correlation are both implemented as FIR-filter. The complexity of these filters can be seen from the table for the complexity of the channel filter(5.1), where $M = 31$ for the 0-symbol and $M = 63$ for the SFD. The complexity of the fine frequency and phase estimator is shown in subsection 5.1.4.

If the the input of the correlator is quantized to 1 or 2 bits, all the filter coefficients will be ± 1. This means that the multiplications can be substituted with additions and subtractions. This can give big complexity reductions.

The detection of the peak is done by comparing the correlation value to a predefined threshold. This comparison can be done by subtracting the threshold amplitude from the correlation value and then checking the sign.

5.1.8 Correlator During Payload

When the correlator starts receiving during the payload, the system is assumed to be synchronized. Thus, it is no longer necessary to correlate for all time instances. The correlator takes one chip sequence at the time.

Different Strategies

- Strategy 1: Correlate for all the values in both the I- and Q-branch
- Strategy 2: Correlate for all the values in the I-branch and then check the sign of the Q-branch
- Strategy 3: Correlate for all the values in the Q-branch

The complexity for the different strategies for chips of length K is shown in table 5.6.

Candidate	**1**	**2**	**3**
Additions	$8K + 16K$	$9K$	$16K$
Multiplications	$8K + 16K$	$9K$	$16K$

Table 5.6: Complexity of the correlator in mode 2 for different strategies

It can be seen from table 5.6 that strategy 1 has much higher complexity than the two other. Strategy 3 has lower complexity than strategy 1, but trows away all the information in the I branch. Strategy 2 has lowest complexity and does not trow information away. Strategy 2 is therefore the most natural choice for a low complexity correlation receiver.

Timing

The described implementation of the correlator is based on 4 samples per chip. This means that for chips with a max amplitude of 1, the worst case amplitude of the sample with the biggest amplitude of the 4 is 0.89. If the rate is reduced to 2 samples per chip, the worst case amplitude is 0.5, which will give a big degradation of the robustness of the system. Increasing the rate to 8 samples per chip, this amplitude will be 0.9808. Using 2 samples per chip will halve the complexity, whereas using 8 will double the complexity of the correlator compared to using 4.

Different Candidates for Detection

In section 3.9 there were described 3 candidate algorithms for detection:

- Candidate 1: Find the largest
- Candidate 2: One threshold
- Candidate 3: Combinations of threshold and finding the largest

Candidate 3 will be omitted in the complexity table since it is a flexible compromise between the two other options. It is assumed that a comparator is used for finding the largest correlation value.

The complexity of the different realizations is summarized in table 5.7

Candidate	**1**	**2**
Additions	8N	4.5N
compare	24N	13.5N
delay	1.5N	1.5N
Multiplications	8N	4.5N

Table 5.7: Complexity of the correlator for different detecton strategies.

Candidate 1 has higher complexity, but is more robust than candidate 2.

5.2 Modulator

One of the possible implementations of the modulator is table look-up. The complexity of this implementation depends on the DSP architecture.

The other possible implementation, is a direct implementation as shown in figure 4.1. The complexity of the of the pulse shaping filter will be equivalent to the values is table 5.1, where $M+1$ is the number taps in the filter and N is the number of samples which is filtered. The complexity of the rest of the modulator depends on the DSP architecture.

Chapter 6

Performance Analysis

This chapter will propose some suitable algorithms for implementation of the different functionalities in the demodulator. These are chosen from the algorithms described in chapter 3. Then, the signal-to-noise improvement through the digital demodulator will be investigated by analytic computations. The effect of phase and frequency offset on the performance of the demodulator will also be investigated analyticly. Lastly, some of the performance metrics for the DSP architecture proposed by Hallvard Næss [25] will be presented.

A simplified block diagram of the receiver analyzed in this chapter is shown in figure 6.1. It was decided to set the downsampling factor to 2. By oversampling at the ADC by a factor of 4 and then running the signal trough the digital low-pass channel filter, the requirement of both the analog and digital low-pass filter could be eased. The quantizer in front of the correlator is set to reduce the accuracy from 8 to 2 bits. This was to reduce the complexity of the correlator. The 2 bits accuracy enabled the correlator to be implemented by only using addition or subtraction and delays. Chapter 7 will show by simulation the effect of 2 bits quantization before the correlator had on the performance in terms of bit error rate (BER).

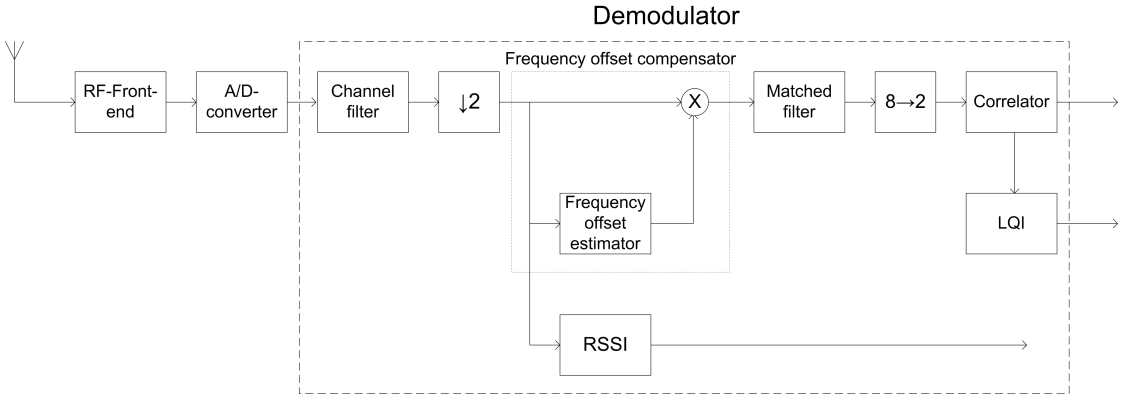


Figure 6.1: A simplified block diagram of the receiver analyzed in this chapter.

6.1 Algorithms

This section will present the chosen algorithms.

It was found that a straight forward implementation of the FIR-filters in the channel filter and matched filter was most suitable for the DSP architecture in [25], because of the overhead of controlling the memory accesses would be bigger than the savings in making use of the symmetry as shown in figure 3.9 and 3.23. This is further described in [25].

6.1.1 Channel Filter

The channel filter used in this analysis is implemented with a sampling frequency of 8 Msamples/s, passband edge frequency of 0.820 MHz, Stopband edge frequency of 2.5 MHz, minimum attenuation in stop-band of -40 dB and maximum pass-band ripple of 0.085 dB. The impulse and magnitude response of this 11-tap filter is shown in figure 6.2.

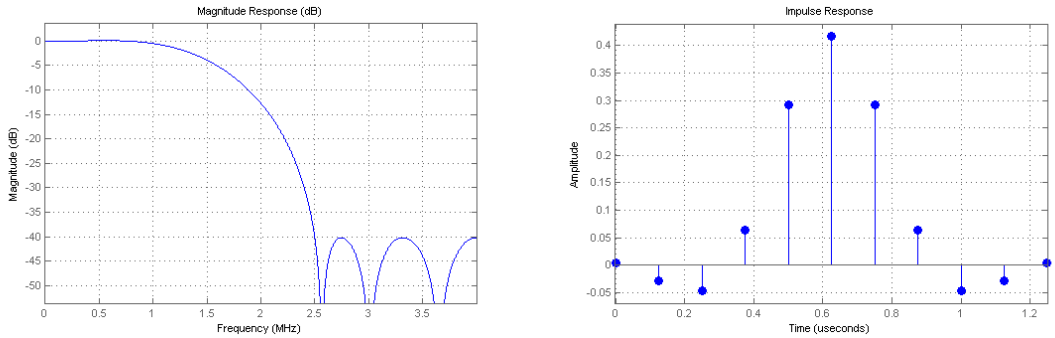


Figure 6.2: Channel Filter

6.1.2 Matched Filter

The matched filter is implemented as a 4-tap FIR filter. The impulse and magnitude response of the matched filter is shown in figure 6.3.

6.1.3 Coarse Frequency and phase Offset Compensator

The coarse frequency offset estimator was implemented as the simplified unweighted Kay estimator. This realization is shown in figure 6.4. The COordinate Rotation DIgital Computer(CORDIC) in vectoring mode was used to find the argument and in rotation mode to perform the complex rotation. The estimator uses the first complex samples of the preamble to obtain an coarse estimate of the frequency offset. The average value of the argument of the next samples is used to initialize the phase accumulator. The number of samples needed for the phase and frequency estimate will be discussed in section 6.2.

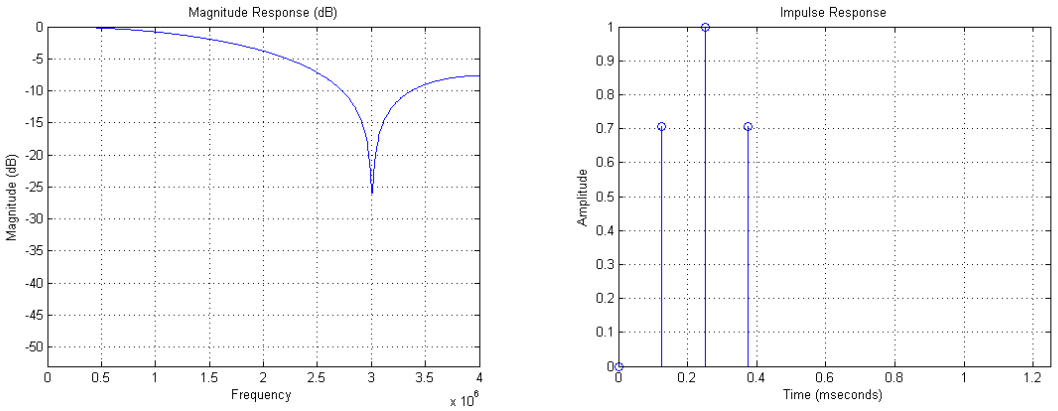


Figure 6.3: Matched Filter

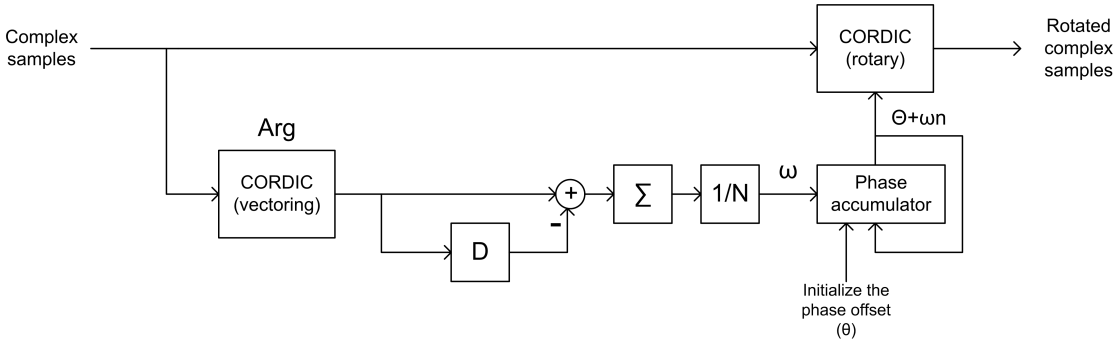


Figure 6.4: freqency offset compensator used in the analysis

6.1.4 Received Signal Strength Indicator

The received signal strength indicator (RSSI) was implemented by straight forward using the algorithm

$$RSSI = \frac{1}{N}\sum_{n=0}^{N-1}|m(n)|^2 = \frac{1}{N}\sum_{n=0}^{N-1}s_I^2(n)+s_Q^2(n), \tag{6.1.1}$$

where N is the number of complex samples over which the average is taken. The IEEE 802.15.4 2.4 GHz physical layer standard defines the average to be taken over 8 symbol periods. A sampling rate of 4 Msamples/sec gives 512 complex samples. In order to lower the complexity the RSSI is computed for every 8^{th} sample, and the result is multiplied by 8. This gives an unbiased estimator for the RSSI value

$$RSSI_1 = \frac{8}{N}\sum_{n=i}^{i+N-1}|m(8n)|^2 = \frac{8}{N}\sum_{n=i}^{i+N-1}s_I^2(8n)+s_Q^2(8n), \tag{6.1.2}$$

where N=512 and thus the average is taken over 64 samples. The complexity is reduced on the expense of increased variance as a consequence of the lost information due to the downsampling.

The IEEE 802.15.4 2.4 GHz physical layer standard defines the RSSI to have a dynamic range of 40 dB, represented as a 8 bit integer. This means that a linear to logarithmic conversion must be implemented. mapping from the received power in decibels to ED value shall be linear with an accuracy of ± 6 dB [12].

6.1.5 Correlator

For mode 1, the correlator was set up as described in section 3.9. This is shown in figure 6.5.

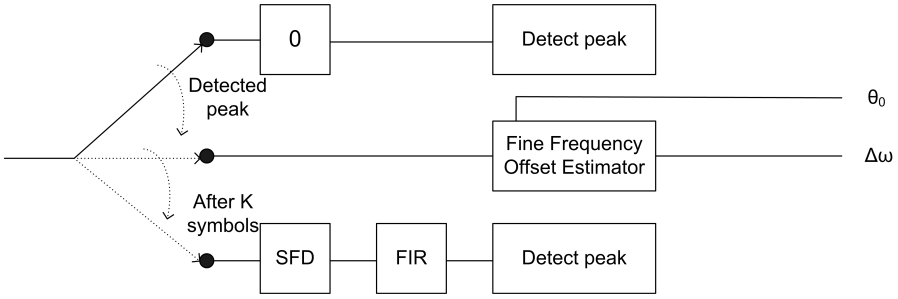


Figure 6.5: The proposed correlator in mode 1: Reception During the Preamble and SFD.

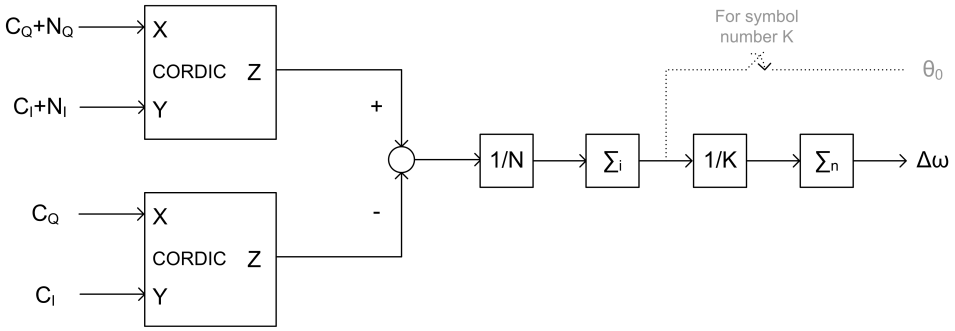


Figure 6.6: The realization of the fine frequency offset estimator used in the analysis.

When the SFD is detected and the right timing is found, the correlator switches into mode 2. It was chosen to use strategy 2, i.e correlate for all the values in the I-branch and then check the sign of the Q-branch. The reason for choosing strategy 2 was that the it had the lowest complexity and was not expected to perform much poorer than strategy 1, which is the direct

implementation of the correlation receiver. The implementation of the correlator in mode 2 is shown in figure 6.7. Candidate 1 for detection of the I-branch was chosen, i.e exhaustive search for the biggest, since this is much more robust than using a threshold and the penalty in complexity is small. The sign of the correlation value in the Q-branch was checked directly, without using a threshold. The reason was that in the case of no detection, the next biggest I-branch value would have to be checked, and the next, and so on. This would add much to the complexity, but does not necessarily improve the performance.

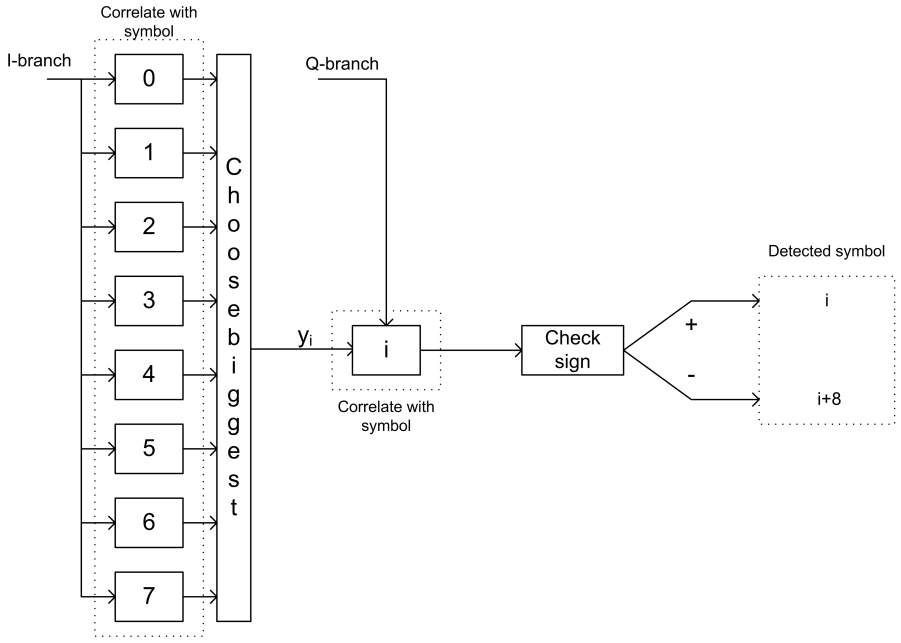


Figure 6.7: The proposed correlator in Mode 2: Reception of payload.

6.1.6 Link quality indicator

$$LQI = \frac{1}{N}\sum_{n=0}^{N} R_n + A, \tag{6.1.3}$$

where N is the number of samples over which the LQI is found, R_n is the correlation value of sample at discrete time n and A is value which moves the average correlation from range $[-128, 127]$ to $[0, 255]$.

6.2 Noise Considerations

This section will present computations done on the theoretical performance of the demodulator. Figure 6.9 show the proposed signal model used in the computations.

This section will present a signal model, which was analyze the theoretical performance of the demodulator

As in chapter 3, the channel noise, quantization noise and CORDIC noise was modeled as additive, white, Gaussian noise (AWGN) with mean value, $\mu_n = 0$ and variance, $\sigma_n^2 = N_0/2$. The noise before the frequency offset estimator is independent of the phase offset in the constellation, so the error in the frequency offset estimator was modeled by introducing a rotation in the constellation. The coding and processing gain and the gain from using coherent demodulation was introduced at the correlator.

With these assumptions, the probability of making errors in the minimum distance receiver, i.e the correlation receiver, could be evaluated. By starting at the end of the signal path in the demodulator, the required E_b/N_0 was found. By assuming an SNR at the input and considering all noise to be AWGN, an estimate of the improvement of the SNR trough the demodulator could be obtained. In order to get a better indication of the total theoretical performance of the demodulator, the effect of frequency and phase offset on the SNR will also be discussed.

6.2.1 Requirements

The IEEE 802.15.4 standard defines a 1% maximum packet error rate (PER). If an uniform distribution of the errors is assumed, the maximum bit error rate for a IEEE 802.15.4 receiver will be

$$BER = 1 - (1 - PER)^{\frac{1}{N}} = 5.71 \times 10^{-5}, \tag{6.2.1}$$

where it is assumed, as in [12], an average packet length of 22 bytes, i.e the total number of bits per packet $N = 176$.

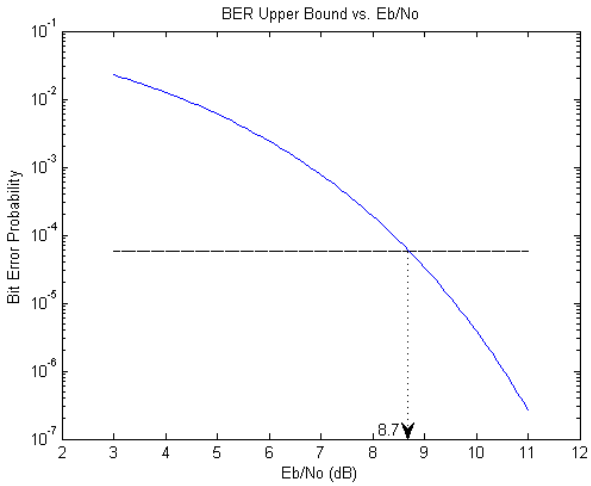


Figure 6.8: BER vs Eb/No curve for a coherently detected uncoded MSK modulation over an AWGN channel using nondifferential data encoding.

The curve for BER vs E_b/N_0 curve for a coherently detected uncoded MSK modulation over an AWGN channel using non-differential data encoding is shown in figure 6.8. The horizontal line shows the minimum bit error rate required from the IEEE 802.15.4 standard. It can be seen from this figure that the demodulator needs E_b/N_0 to be higher than 8.7 dB, at the output of the correlator in order to achieve the required BER.

It was decided to assume 0 dB SNR in the channel. This was based on the 3 dB co-channel rejection of Chipcons ZigBee-ready RF chip [8], which is an good indication of the SNR requirements. Adding a 3 dB implementation loss margin to this gives a requirement for SNR $= 0$ dB at the input of the demodulator.

The signal model is shown in figure 6.9.

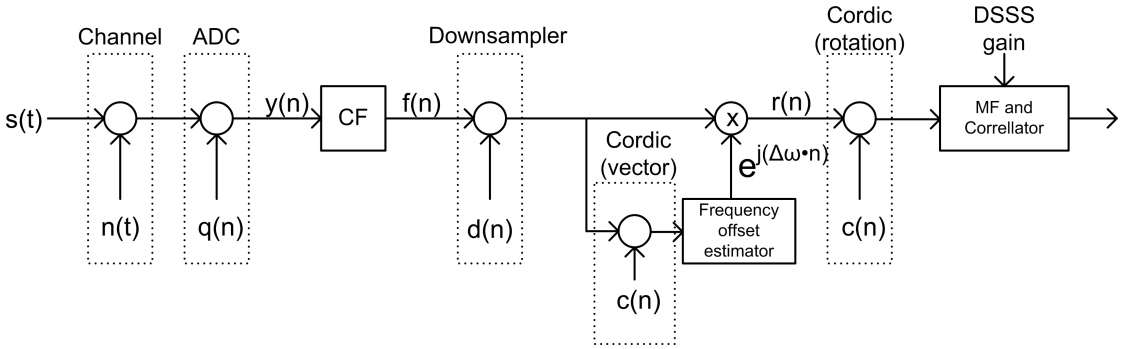


Figure 6.9: Signal model of the demodulator.

6.2.2 Channel

All noise added between the digital modulator and demodulator is in this model referred to as channel noise. This includes atmospheric noise, thermal noise, shot noise, et cetera.

The computations presented in this section assumes a signal to noise ratio (SNR) of 0 dB at the input of the ADC, which means that the signal power, σ_s^2, equals the channel noise power, σ_n^2.

The automatic gain control (AGC) system is assumed to be ideal in the sense that the variable gain amplifier (VGA) makes sure the signal level is scaled to the whole dynamic range of the analog to digital converter (ADC). This minimizes the noise. In this model, the amplitude range of the ADC is set to ± 1. The pseudo-random noise (PN) sequences consists of as many 0's as 1's, so the average value, μ_s=0. Thus, the average signal power at the input of the ADC is given by

$$\sigma_s^2 = \frac{1}{2T_c}\int_0^{2T_c} sin^2(\frac{\pi t}{2t_c})dt = \frac{1}{2}, \tag{6.2.2}$$

where $2T_c$ is the chip period. Which gives the noise power

$$\sigma_n^2 = \frac{1}{2} = \int_0^B N_0 dF. \tag{6.2.3}$$

where $B = F_s/2 = 4$ MHz. Thus, the power spectral density of the white noise is given by

$$N_0 = \frac{1}{2B}. \tag{6.2.4}$$

6.2.3 Analog-to-Digital Converter

The quantization noise is defined as the difference between the input and output signal to a quantizer. It is assumed a uniform midrise type quantizer with range $(-1,1)$ and sufficient number of levels for the quantization error to be a uniformly distributed random value with mean, $\mu_Q = 0$. In this case, the quantization noise variance is given by [14]

$$\sigma_q^2 = \frac{1}{3} m_{max}^2 2^{-2R}, \tag{6.2.5}$$

where m_{max} is the maximal amplitude of the quantizer and R is the number of bits used. The ADC has a 8 bits resolution and the VGA scales the maximum amplitude of the signal to be 1, which gives $R = 8$ and $m_{max} = 1$ into 6.2.5

$$\sigma_q^2 = \frac{1}{3} 1^2 2^{-16} = 5.09 \times 10^{-6}, \tag{6.2.6}$$

which gives the average quantization noise power, since $\mu_q = 0$.

6.2.4 Channel Filter

The channel filter attenuates the noise at the high frequency, whereas the signal is left unattenuated. This improves the SNR. The quantization noise compared to the channel noise is

$$\log(\frac{\sigma_q^2}{\sigma_n^2}) \approx -50\,\text{dB}. \tag{6.2.7}$$

The quantization noise is approximately 50 dB smaller than the channel noise and can therefore be ignored. This gives the power spectral density at the input of the channel filter

$$S_y(F) \approx S_n(F) = N_0 = \frac{1}{2B}, \tag{6.2.8}$$

where $B = F_s/2 = 4$ MHz. This means that the noise variance at the output of the filter is

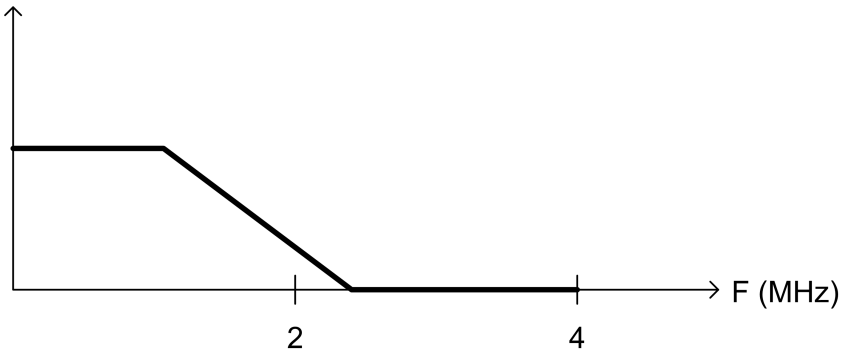
$$\sigma_f^2 = \int_0^{F_s/2} |H(F)|^2 S_y(F) dF = \int_0^B |H(F)|^2 N_0 dF = 0.1794, \tag{6.2.9}$$

where $|H(F)|^2$ is the squared magnitude spectrum of the channel filter described in section 6.1.

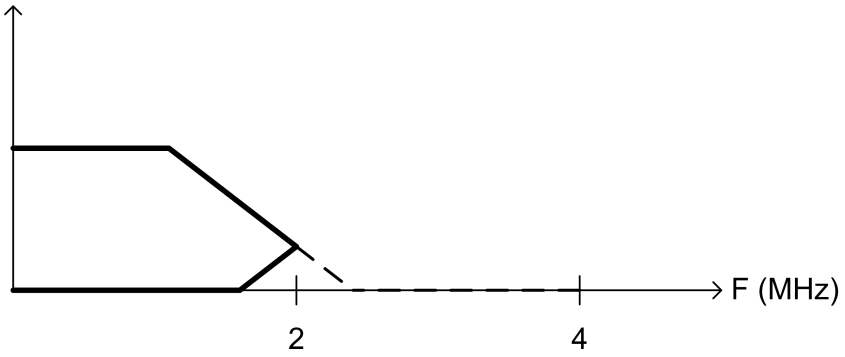
6.2.5 Down Sampling

When the signal is downsampled by 2, the repeated spectra will be around $F_s = 4MHz$ as seen from figure 6.10. This causes the spectrum to be folded around $F_s/2 = 2MHz$. The noise before and after down sampling is unchanged, and therefore

$$\sigma_d^2 = 0. \tag{6.2.10}$$



(a) The spectrum of the noise at the original sampling frequency



(b) The downsampled noise spectrum

Figure 6.10: The effect of downsampling on the noise spectrum.

6.2.6 CORDIC

The CORDIC algorithm is used for finding the argument in the frequency and phase offset estimator and to perform the complex rotation. In the CORDIC there will be a angle approximation error and a round-off error. The angle approximation is caused by the fact that all angles at the output of the CORDIC are given by left or right rotations, called micro rotation, by a finite number of elementary angles, $\alpha(i) = \tan^{-1}(2^{-i})$, where i is an integer. The CORDIC in this analysis is assumed to have a precision of 8 bits, since the width of the data path is chosen to be 8 bits wide [25]. It is therefore chosen to do 8 iterations of the CORDIC. The round-off error

is caused by finite precision arithmetic in the micro rotations and multiplication of the scaling factor. It is shown in [26] that the angle approximation mean square error in using the CORDIC for uniformly distributed errors is

$$E\left[|\mathbf{e}_a(N)|^2\right] = (2 - 2 \cdot \frac{\sin(\alpha(N-1))}{\alpha(N-1)}) \cdot E|\mathbf{v}(0)|^2, \tag{6.2.11}$$

where $E(\cdot)$ is the statistical expectation, $\mathbf{v}(0)$ is the input vector, $\alpha(i)$ is the i^{th} elementary angle. The input vector consists of the I and Q component of the signal.

The round-off error is given in [26] as

$$e_{or} = \frac{1}{K}(e_r(N) + \sum_{i=0}^{N-1} \prod_{j=i}^{N-1} P(j) e_r(i)) + e_s, \tag{6.2.12}$$

where K is the scaling factor, N is the number of iterations, $P(j)$ is the rotation matrix, $e_r(i)$ is the round-off error of the i^{th} iteration.

The total MSE is shown in [26] to be

$$E|e_o|^2 = E|e_a(N)|^2 + E|e_{or}|^2. \tag{6.2.13}$$

In the CORDIC implementation described in [25] the number of iterations $N = 8$, the average input power $E|\mathbf{v}(\mathbf{0})^2| = \sigma_s^2 = 0.5$, $K \approx 1.6468$. This gives

$$E|e_o|^2 = 3.3 \times 10^{-4}. \tag{6.2.14}$$

This gives a good estimate of the noise variance of the CORDIC, $\sigma_c^2 = 3.3 \times 10^{-4}$, since it is reasonable to assume $\mu_c = 0$, because the probability of a positive or negative deviation in the CORDIC is the same.

6.2.7 Frequency and Phase Offset Compensators

The frequency offset compensators are divided in two parts, the frequency and Phase offset estimator and the complex rotation (compensation). The frequency and phase offset estimator does not introduce additive noise, but introduces a time varying phase offset and will therefore be treated in a separate section regarding the frequency and phase offset's effect on the BER.

The complex rotation is performed by the CORDIC, and will thus add the noise variance, $\sigma_c^2 = 3.3 \times 10^{-4}$ for every sample.

6.2.8 Received Signal Strength Indicator

The estimate of the received signal strength indicator (RSSI) was done by taking only every 8 sample into account, and thus only taking some of the spectrum into account for estimating the power. This means that the variance increases as the number of samples decreases. It is shown in [20] that the average variance of an average over independent and identically distributed samples decreases proportionally with the number of samples, n. This means that the decimation by factor 8 increases the variance of the RSSI estimate 8 times. The variance of the RSSI does not affect the BER of the demodulator and is therefore omitted in figure 6.9.

6.2.9 Quantizer

The quantizer before the correlator lowers the resolution from 8 bits to 2 bits. This gives an additional quantization noise of

$$sigma_{q2}^2 = \frac{1}{3}(2^{-4} - 2^{-16}) = 0.021. \tag{6.2.15}$$

This formula assumes that the quantization error is a uniformly distributed random variable. This does not apply when the number of levels is low [14]. This is therefore a rough estimate of the quantization noise.

6.2.10 Matched Filter and Correlator

The total SNR before the processing gain (PG) is

$$\begin{aligned} SNR &= 10\log(\frac{\sigma_s^2}{\sigma_f^2+\sigma_d^2+\sigma_k^2+\sigma_c^2+\sigma_{q2}^2}) \\ &= 4.0\,\text{dB} \end{aligned} \tag{6.2.16}$$

The improvement in SNR is caused by the channel filter, because of the large channel spacing and the fact that the noise at the input of the demodulator dominates the SNR. Though, there is still a lot to gain in terms of noise enhancement by using a higher order filter, since the stop-band cut-off frequency is at 2.5 MHz.

The processing gain of DSSS is given by

$$PG = 10\log(\frac{R_c}{R_b}) = 9.0\,\text{dB}, \tag{6.2.17}$$

where R_c is the chip-rate and R_b bit-rate. There will also be a coding gain as a result of the increased Hamming distance. The coding gain was found to be

$$CG = 0.8\,dB, \tag{6.2.18}$$

as seen in figure 6.11, where the blue curve shows the uncoded and the red curve shows the coded case. It can be seen from the figure that the gain increases with increasing E_b/N_0, but the gain will be 0.8 dB for the worst allowable case, i.e 5.7×10^{-5} BER.

The total SNR of the demodulator is therefore

$$SNR_{\text{tot}} = SNR + PG + CG + 3dB = 16.8\,dB, \tag{6.2.19}$$

where the 3 dB gain comes from using coherent demodulation [14]. Thus, 6.2.19 shows that there will be a theoretic gain of 16.8 dB trough the demodulator assuming perfect sampling timing and no frequency and phase offset for 0 dB SNR at the input. The theoretic maximum gain of the demodulator is given by the case where the the channel noise totally dominates, i.e low SNR. The gain trough the demodulator in this case is

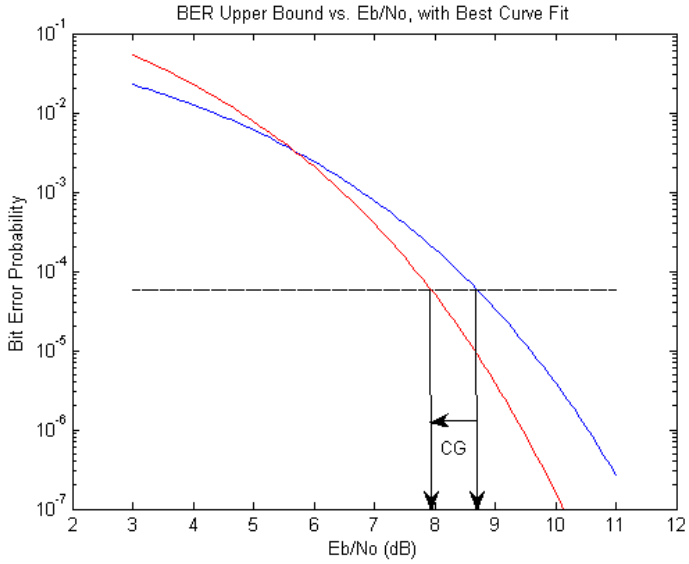


Figure 6.11: Coding gain in the demodulator.

$$SNR_{\max} = 10\log(\frac{1}{\int_0^B |H(F)|^2 dF}) + PG + CG + 3dB = 17.25\,dB, \tag{6.2.20}$$

where $|H(f)|^2$ is the squared magnitude spectrum of the channel filter described in section 6.1. The lowest gain trough the system is for high SNR, since the channel filter does not improve the SNR. The gain trough the demodulator in this case is

$$SNR_{\min} \approx PG + CG + 3B = 12.8\,dB. \tag{6.2.21}$$

This is an underestimation, since the coding gain increases with the SNR as seen from figure 6.11, but gives an indication of the lower bound of the gain in the demodulator. The estimate so far has not taken the frequency and phase offset into account. This is done in the following section.

6.2.11 Frequency and Phase offset

The constant rotation in the constellation caused by a frequency offset was compensated by the frequency offset compensator. The frequency offset estimators are unbiased but have an average deviation from the mean, i.e the standard deviation σ_k. Thus, the constellation still rotates, but at a much lower speed. This section will discuss the effect this rotation will have on the BER.

The preamble consists of 8 o-symbols, i.e 512 complex samples. Letting the first half symbol pass by and adding a margin on half a symbol at the end, gives 448 complex samples for use. The first 187 of these were used to obtain a coarse blind estimate of the frequency and phase offset.

Then, one symbol was used to find the right frame synchronization and the rest of the samples were used to find a fine estimate of the frequency offset. The frequency and phase estimate after each symbol was fed back to the phase accumulator to initialize phase and update the frequency estimate. This section will show the reasoning behind the choices made.

The rotation of the constellation causes the I and Q-channel to loose orthogonality, i.e act as noise on the other channel. In addition to the added noise, the wanted signal will be attenuated. This has a large effect on the SNR and therefore degrade the precision of the fine frequency and phase offset estimator. This leads to a coarser frequency and phase estimate in the fine frequency offset estimator, which leads to degradation in the performance of the demodulator in mode 2.

This means that the ratio between the wanted and unwanted component has to be high. This ratio is for the I-channel at discrete time instance n

$$R = \frac{\cos(\theta(n))s(n)}{\sin(\theta)s(n) + r(n)}, \tag{6.2.22}$$

where $\theta(n) = \omega n + \theta_0$ is the phase offset, $s(n)$ is the I-channel signal and $r(n)$ is additive noise. Figure 6.12 shows the wanted and unwanted signal component for the I-branch for a rotated constellation.

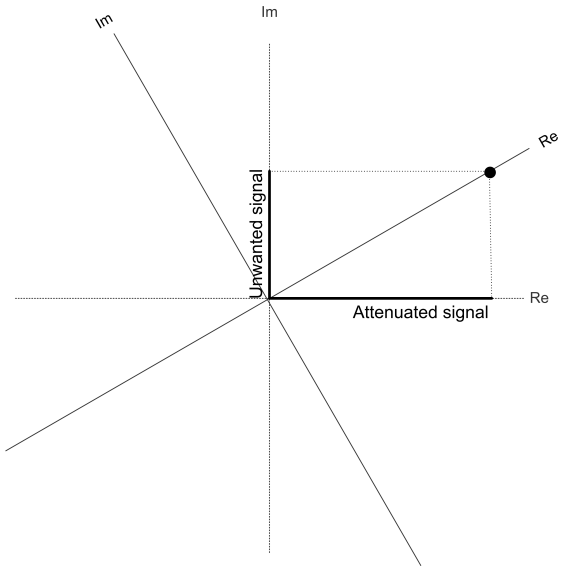


Figure 6.12: The effect of phase offset on the orthogonality between I and Q.

It was shown in the introduction to this chapter that the E_b/N_0 needed at the output of the system was 8.7 dB. The minimum SNR is therefore given by [6]

$$SNR_{\min} = (\frac{E_b}{N_0})_{min} + 10\log_{10}(b) = 11.7\,dB. \tag{6.2.23}$$

By applying the phase offset to equation 6.2.19, the SNR as a function of the time varying phase offset offset is given by

$$SNR(\theta(n)) = 10\log_{10}\left(\frac{\sigma_s^2\cos^2(\theta(n))}{\sigma_s^2 sin^2(\theta(n)) + \sigma_f^2 + \sigma_{q2}^2}\right) + CG + PG + 3dB, \tag{6.2.24}$$

where σ_k^2 and σ_c^2 from are negligible and therefore omitted. Figure 6.13 shows the SNR versus the phase offset in degrees in the blue line and the black dotted horizontal line shows the lowest allowable SNR. It can be seen from the figure that the demodulator falls below this boundary at an phase offset of approximately 39 degrees. If the phase offset is constant, it has to be lower than 39 degrees. If it is time varying, i.e there is a frequency offset, this boundary is not as strict, since the average BER can still be lower than the required 5.7×10^{-5} BER.

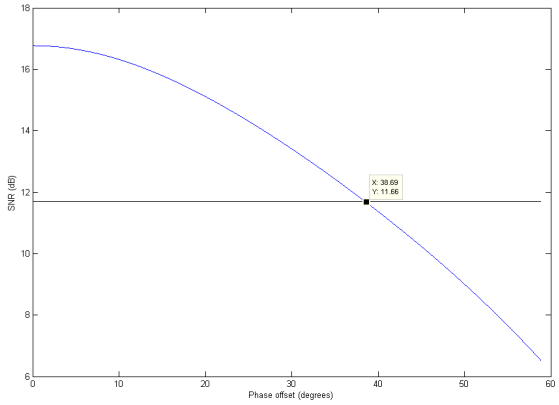


Figure 6.13: The SNR versus the phase offset.

Coarse Frequency and Phase Offset Estimator

It is important that the coarse estimate is good so the fine estimator estimator can work on high SNR, but it is also important that it uses as few samples as possible so the fine estimator can work on as many samples as possible.

The variance of the unweighted frequency estimator described by Kay is given by [18]

$$\sigma_K^2 = Var(\hat{\omega}_0) = \frac{1}{(N-1)^2 SNR}, \tag{6.2.25}$$

where SNR is the signal to noise ratio at the input of the estimator and N is the number of samples of the incoming signal and $SNR = 2.78$.

It was decided that the fine frequency estimator should work on SNR no lower than 16 dB, because the performance of the estimator degrades as a function of the SNR. This boundary can be put lower which leaves more samples for the fine estimator. This thesis will not consider the optimization of the number of samples in the two different estimators, although this may lead to improvement in the performance of the demodulator. It can be seen from 6.13 that this

means a maximum phase offset of 13.3 degrees at the fine estimator in the correlator. Since the fine estimator adjusts the phase every symbol, this offset is the maximum for one symbol, i.e 64 complex samples. This gives a maximum phase change per sample, i.e frequency offset, of 0.21 degrees/sample or equivalently 0.0036 rad/sample. By using the standard deviation as an estimate of the offset, the minimum number of samples needed for the coarse estimate is

$$N = \left\lceil \sqrt{\frac{1}{\sigma_k^2 SNR} + 1} \right\rceil = 167. \tag{6.2.26}$$

Because of the signal variance, some samples must also to be used to find an estimate of the phase offset. It is assumed that 20 samples is sufficient.

Frame Synchronization for Frequency Offset Estimator

In order to make use of the known information, the frame synchronization had to be found before starting the fine estimation. This was done by correlating with the 0-symbol and detecting the peak. The autocorrelation function for symbol 0 is shown in figure 2.2. The peak will give the instant of lag 0 and thus the right timing.

As shown, the coarse estimator needs 187 of the 448 complex samples, which means that 325 samples are left for the fine estimator. The worst case scenario is that 63 complex samples are needed to find the right synchronization. This means that 218 samples are left for the fine frequency estimation.

Fine Frequency Offset Estimator

The fine estimator has to make sure that the total phase offset does not exceed 39 degrees, i.e 0.68 radians, as shown in figure 6.13. The max payload package size for IEEE 802.15.4 is 254 symbols, i.e 16256 complex samples with sampling rate 4 samples/chip. This means that the maximum rotation per sample is 2.4×10^{-3} degrees/sample, i.e 4.2×10^{-5} rad/sample.

Figure 6.14 shows a figure from [21]. It can be seen that all the data-aided frequency offset estimators described in the article approaches the Cramér-Rao bound for estimation length of 128 samples for $SNR \geq 16$ dB. The Cramér-Rao bound expresses the upper bound on the precision of a statistical estimator. The Cramér-Rao bound for SNR=16 dB is approximately $\sigma^2_{\text{CR}} = 10^{-9}$. Adding a 100% margin to this gives $\sigma^2_{\text{freq}} = 2 \times 10^{-9}$, i.e. $\sigma_{\text{freq}} = 4.47 \times 10^{-5}$. The standard deviation can be used as an estimate of the minimum rotation per sample after compensation. This means that the phase must be adjusted at least 1 time during the reception of a maximum length packet.

The SNR for a received sequence was then obtained by using equation 6.2.24 and letting the phase at discrete time n be $\theta(n) = \omega n + \theta_0$, where $\omega = \sigma_{\text{freq}} = 4.47 \times 10^{-5}$ and θ_0 is the constant phase offset. Phase adjustment was made at time instant $N/2$. This was done by subtracting $\omega \cdot N/2$, where N is the number of complex sample. Figure 6.15a shows a plot of the SNR versus sample where, the phase adjustment is done after 127 of the 254 symbols are received. Figure 6.15b shows the same plot, but with a phase offset $\omega_0 = 5.6$ degrees. It can

be seen that the frequency and phase offset has a large impact on the SNR at the output of the demodulator.

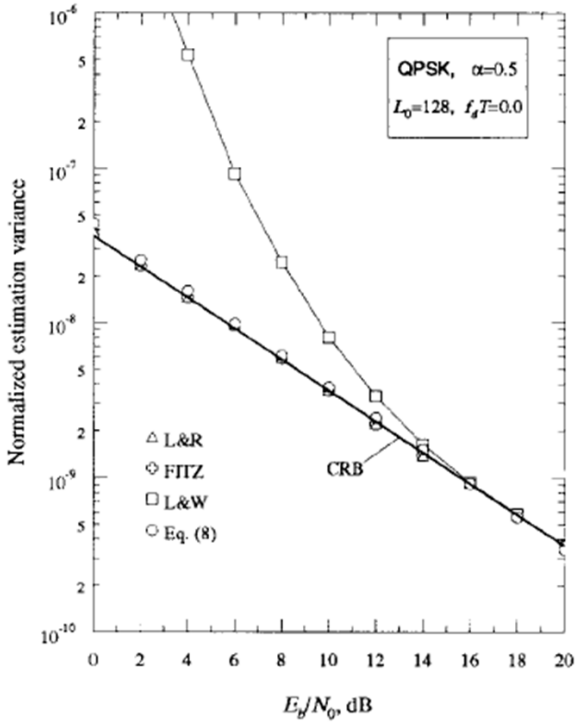


Figure 6.14: The Cramér-Rao Bound for data-aided frequency offset estimators [21].

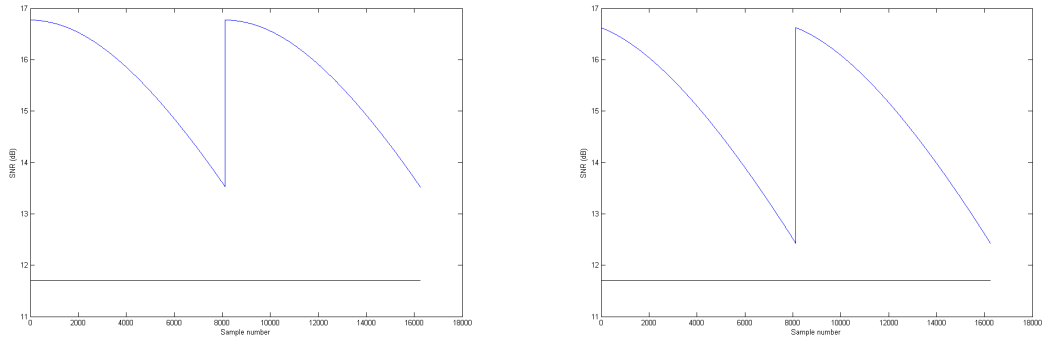
6.3 Architecture

This section will give a short summary of the estimated computational complexity for implementing the IEEE 802.15.4 digital demodulator [1]. Also current and area consumption of the proposed DSP architecture will the presented. All the results in this chapter are taken from [25]. The proposed architecture is shown in figure 6.16. For further details, the reader is referred to [25].

6.3.1 Estimation of Computational Complexity

The computational complexity for the different operations implemented on the proposed DSP during the coarse frequency offset estimation is shown in table 6.1, during the SFD is shown in table 6.2 and during the payload of 0 to 510 symbols, is shown in table 6.3.

[1] The estimates obtained by Hallvard Næss does not take the fine frequency offset estimator into consideration.



(a) With frequency offset after compensation (b) With frequency and phase offset after compensation

Figure 6.15: The SNR vs. sample number for the max length packet with one phase adjustment.

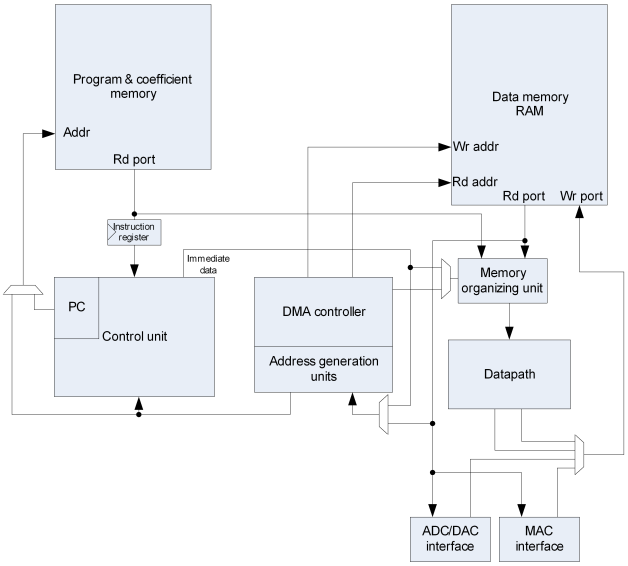


Figure 6.16: The architecture proposed by Hallvard Næss in [25].

6.3.2 Estimated Current Consumption

It is shown in [25] that the estimated current consumption for the DSP during packet reception using $0.18\mu m$ technology is approximately 5 mA.

6.3.3 Estimated Area

It is shown in [25] that the estimated area of the DSP is approximately 40000 gates.

Operation	**% of total complexity**
Channel filter	63
RSSI calculation	2
Frequency offset estimation	22
Additional control overhead	13
Minimum clock frequency	**50 MHz**

Table 6.1: Computational complexity of the frequency offset correction

Operation	**% of total complexity**
Channel filter	15
Frequency offset correction	3
Matched filter	7
SFD Correlation	73
Additional control overhead	2
Minimum clock frequency	**200 MHz**

Table 6.2: Computational Complexity during SFD detection.

Operation	**% of total complexity**
Channel filter	41
Frequency offset correction	7
Matched filter	19
Symbol Correlation	24
Additional control overhead	9
Minimum clock frequency	**75 MHz**

Table 6.3: Computational complexity during payload reception.

Chapter 7

Simulation

This chapter will present a Matlab simulation done on the demodulator proposed in chapter 6. Simulations was also done with a 4 bits ADC instead of 8 bits in addition to the quantization before the correlator and without quantization before the correlator. These 3 different realizations were also simulated with a frequency offset and a frequency and phase offset. The moodulator was realized as described in chapter 4. The modem was only simulated in mode 2, where assumptions on the synchronization and timing were made on basis of the previous chapter.

The simulation was done with 0.5 dB intervals for SNR and a maximum length of 300 packets, i.e 304800 bits. Thus, this simulation has a low precision, but can serve as an indication of the performance the modem and some of the trade-offs.

The simulation model is shown in figure 7.1.

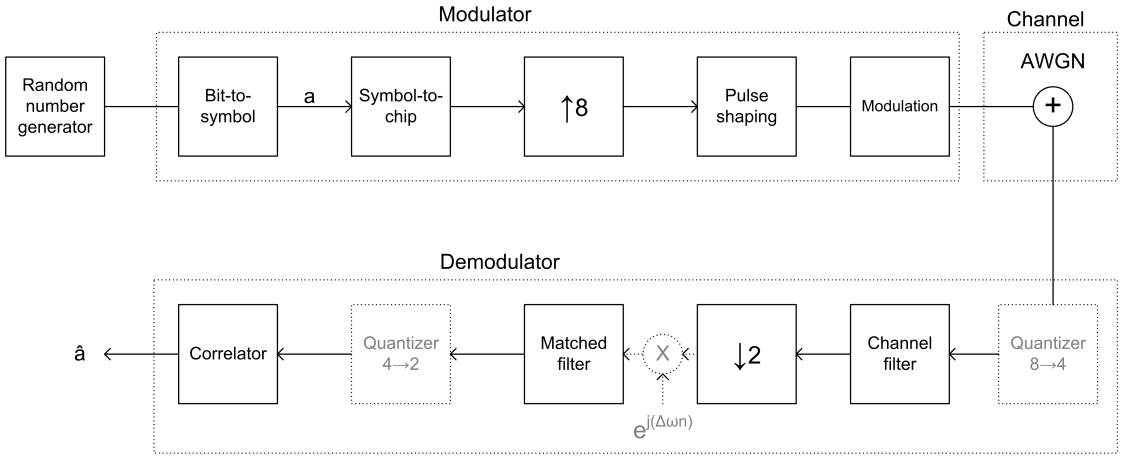


Figure 7.1: Equivalent model of the system.

7.1 Implementation

The modem was implemented as shown in figure 7.1. The random number generator produces n random binary numbers. The n was set on basis of the total number of bit error. For a low SNR, and thus high BER, n was set to be small, and vice versa. The size of n for each case will be specified in the presentation of the results. The bit-to-symbol mapping groups four and four bits to give a symbol a. This symbol is mapped to one of the 16 chip sequences described in chapter 2. Then, the sequence is upsampled 8 times because the model assumes 4 times oversampling by the ADC at the input of the digital demodulator. The half-sine pulse shaping filter has therefore 8 taps and the 8 times upsampling is necessary in order to prevent inter symbol interference (ISI). Then, every other chip is modulated onto the I and Q channel, using QPSK modulation. This simplification could be done because O-QPSK has exactly the same BER over a AWGN channel as the QPSK for coherent detection [14]. The channel adds complex white Gaussian noise (AWGN). The signal is then filtered by the channel filter, downsampled and put trough the 4 tap matched filter. At last, the information in the I and Q carrier is divided into separate branched and put into the correlator. The estimated symbol $\hat{a}$ is compared with the sent symbol a to find the BER.

The Matlab code for the simulation is shown in shown in Appendix B.

7.1.1 Case 1: Modem without Quantization

Case 1 was implemented as described in the introduction to this section. It was shown in the previous chapter that the quantization noise from a 8 bit quantizer is $\sigma_q^2 = 5.09 \times 10^{-6}$. The highest SNR trough the channel used in the simulation was 0 dB, which gives a negligible quantization noise for 8 bits. Thus, the performance of the modem in this case is equivalent to the one using 8 bits precision trough the whole modem.

7.1.2 Case 2: Modem with Quantization Before the Correlator

Case 2 was implemented as described in the introduction to the section, but the input of the correlator was quantized to 2 bits. The codebook of the quantizer contains the values -1, 0 and 1.

7.1.3 Case 3: The Modem with Quantization Before the Channel Filter and Correlator

Case 3 was implemented as described in the introduction to the section, but the input of the channel filter was quantized to 4 bits, which is equivalent to using a 4 bits ADC. As for case 1, the input of the correlator was quantized to 2 bits.

Since the number of bits per samples was halved compare to case 2, the complexity of the channel filter could be increased without increasing the total complexity of the modem. This means that the channel filter can be designed with sharper transition area which can compensate for the increased noise level. Thus, there is a trade-off between the precision per sample and the complexity of the channel filter. The channel filter used in case is shown in figure 7.2

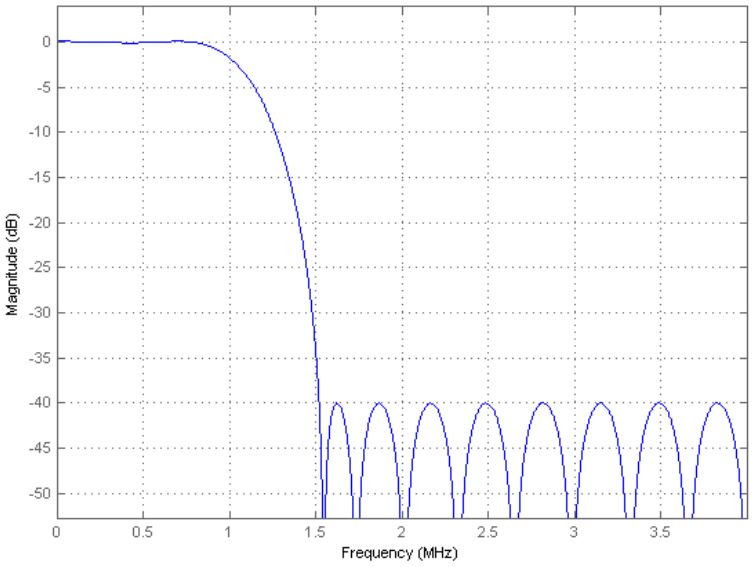


Figure 7.2: Magnitude response of the channel filter used in case 3

7.1.4 Phase and Frequency Offset

The previous chapter discussed the effects of phase and frequency offset. It was showed analytically how the SNR degradate as a consequence of the total phase offset. The simulation was done by introducing a frequency offset of as a function of the SNR as found in the previous chapter. The frequency offset was which was introduced into the simulation was based on the Cramér-Rao bound described in [21], and was given by

$$\Delta\omega = \sqrt{4\times10^{-8} - SNR\cdot 1.98\times10^{-9}}, \quad 0 \leq SNR \leq 20 \tag{7.1.1}$$

which is the square root of a linearized model of the variance boundary shown in figure 6.14. The frequency offset used in the simulation is shown in figure 7.3. The SNR used for the mapping from SNR to frequency offset in the simulation was given by

$$SNR_{\text{sim}} = SNR + 9.8, \quad -9.8 \leq SNR \leq 10.2, \tag{7.1.2}$$

Since $0 \leq SNR_{\text{sim}} \leq 20$. SNR is the channel SNR and 9.8 is the coding and processing gain.

A simulation was also done with a phase offset of 5.6 degees, i.e $\pi/32$.

The length of the packets was set to be 254 symbols, which is the maximum length of the payload of a 802.15.4 2.4 GHz physical layer packet. The average BER was found by averaging over 20 packets for the lowest SNR and 300 packets for the highest SNR.

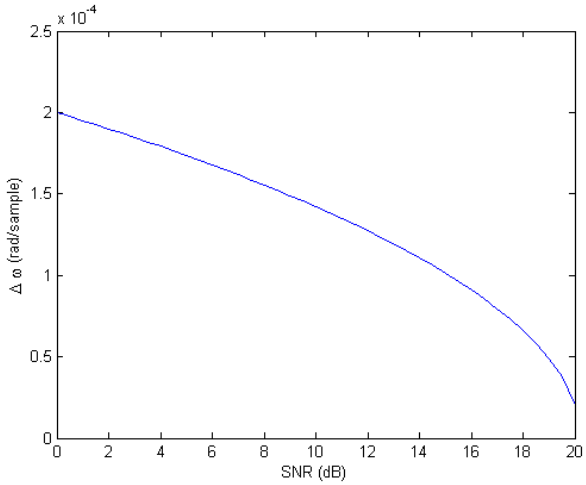


Figure 7.3: Frequency offset vs. SNR for the simulation model.

7.2 Results

This section will present the result from the simulation and the analytic computations.

7.2.1 Simulation

This section will first present the simulation results[1] for each of the 3 cases separately, then compare them and show the BER for the demodulator with frequency offset for different intervals between the phase compensation.

Figure 7.4 shows the the BER vs. SNR for the demodulator using no quantization. The requested BER of 5.7×10^{-5} is reached at -4.5 dB for the non-rotated case , 3.5 dB for the frequency offset case and 4 dB for the frequency and phase offset case. This means that there is a penalty of 8 dB as a consequence of the frequency and another 0.5 dB for the phase offset. This result is based on 1 phase compensation after half the packet is sent.

Figure 7.5 shows the the BER vs. SNR for the demodulator with 2 bits quantization before the correlator. The requested BER of 5.7×10^{-5} is reached at -3.5 dB for the non-rotated case , 6 dB for the frequency offset case and 7 dB for the frequency and phase offset case. This means that there is a penalty of 9.5 dB as a consequence of the frequency and another 1 dB for the phase offset. This result is based on 1 phase compensation after half the packet is sent.

Figure 7.6 shows the the BER vs. SNR for the demodulator with 2 bits quantization before the correlator and 4 bits quantization at the input of the channel filter. The requested BER of 5.7×10^{-5} is reached at -2.5 dB for the non-rotated case , 5.5 dB for the frequency offset case and 6.5 dB for the frequency and phase offset case. This means that there is a penalty of 8 dB as

[1]The simulations are done with up to 300 max length packets, i.e 304800 bits. The number of bits could not be increased because of lack of time. This caused low precision the results for the lowest BER.

a consequence of the frequency and another 1 dB for the phase offset. This result is based on 1 phase compensation after half the packet is sent.

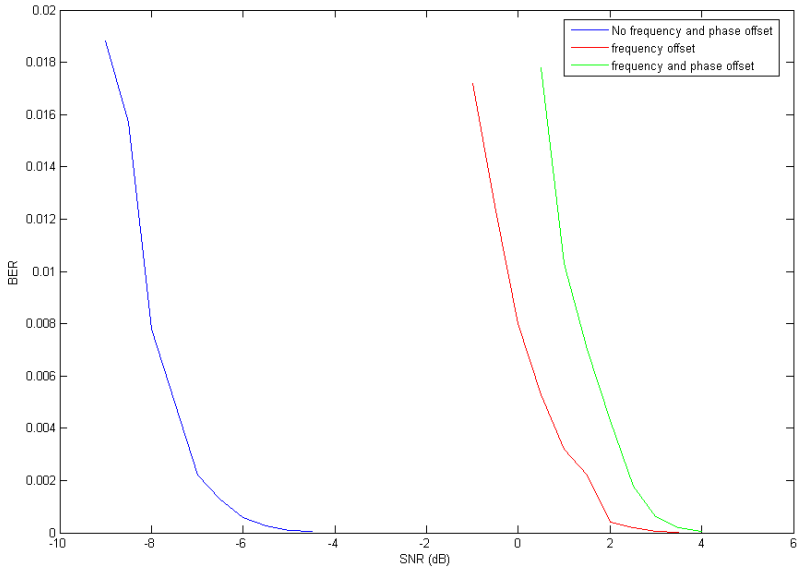


Figure 7.4: BER vs. SNR for the demodulator using no quantization.

Figure 7.7 shows the BER vs. SNR for the demodulator for the 3 cases without rotation, i.e frequency and phase offset. The requested BER of 5.7×10^{-5} is reached at -4.5 dB case 1 , -3.5 dB for case 2 and 2.5 dB case 3. Thus, there is a 1 dB penalty in using case 2 over case 1 and another 1 dB for choosing case 3.

Figure 7.8 shows the the BER vs. SNR for the demodulator for packet reception with frequency offset for 1, 2, 3 and 4 phase adjustments during reception. The blue line is the case without frequency offset. The requested BER of 5.7×10^{-5} is reached at -4.5 dB for the non-rotated case, -3.5 dB the case with 4 compensations, -2.5 dB the case with 3 compensations, -0.5 dB the case with 2 compensations and 3.5 dB the case with 1 compensations. Thus, there is a 1 dB penalty in using case 2 over case 1 and another 1 dB for choosing case 3.

7.2.2 Analytic

Figure 7.7 showed that the requested BER of 5.7×10^{-5} was reached at -4.5 dB case 1 , -3.5 dB for case 2 and 2.5 dB case 3. These SNR values were used to analytically compute the theoretical BER for the 3 cases. This was only done without introducing frequency and phase offset into the calculations. Details about the calculations is found in appendix A.

Case 1, with a -4.5 dB SNR in the channel was found to have a 7.8×10^{-6} BER. Case 2, with a -3.5 dB SNR in the channel was found to have a 1.1×10^{-6} BER. Case 3, with a -2.5 dB

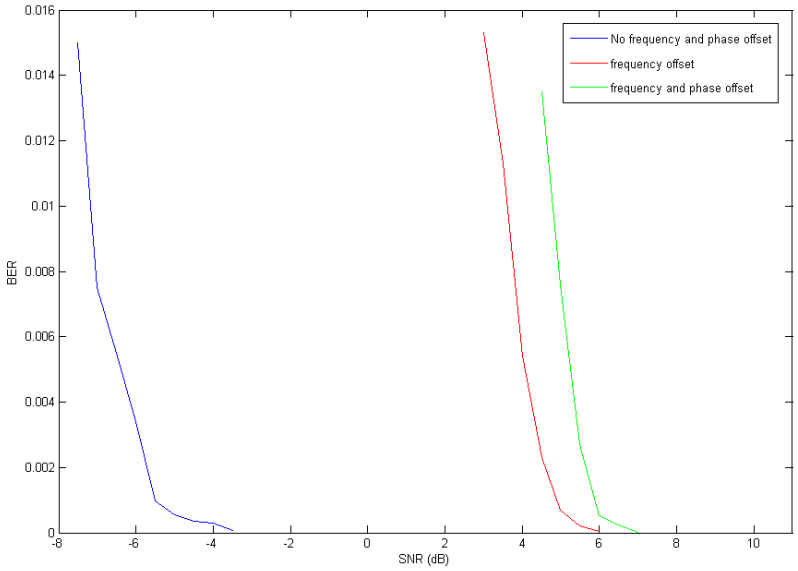


Figure 7.5: BER vs. SNR for the demodulator with 2 bits quantization before the correlator.

SNR in the channel was found to have a 6.8×10^{-13} BER.

The simulation results for case 1,2 and 3 were found to be 3.61×10^{-6}, 5.4×10^{-6} and 1.97×10^{-6}.

7.3 Discussion

It can be seen from figure 7.4, 7.5 and 7.6 that there is a large degradation of the system for frequency and phase offset. A lot may therefore be gained in optimizing the frequency and phase offset compensator. Another way of closing the performance gap between the non-rotated and rotated case can be to increase the number of phase adjustments, as shown in figure 7.8. Increasing the number of adjustments will lead to a small increase the complexity of the demodulator, but a big increase in the performance.

Figure 7.7 show the tree different cases, without rotation of the constellation. Case 3 used a 4 bit precision and a 24 tap FIR channel filter and a 2 bit precision in the correlator. Case 2 used a 8 bit input and a 11 tap FIR filter. Table 6.3 show the complexity for case 2 in mode 2. It can be seen that the channel filter has the highest complexity. Thus, case 2 and 3 has approximately the same complexity. This means that case 2 probably is the better of these two, since it has a 1 dB lower SNR for the requested BER. Case 1 has higher complexity than case 2, but also better performance. Table 6.3 show that the correlator takes up 24% of the complexity in mode 2. In case 1, the complexity of the correlator is 4 times as big as in case 2 when implemented on the DSP architecture proposed by [25]. This means that case 2 has much lower complexity than

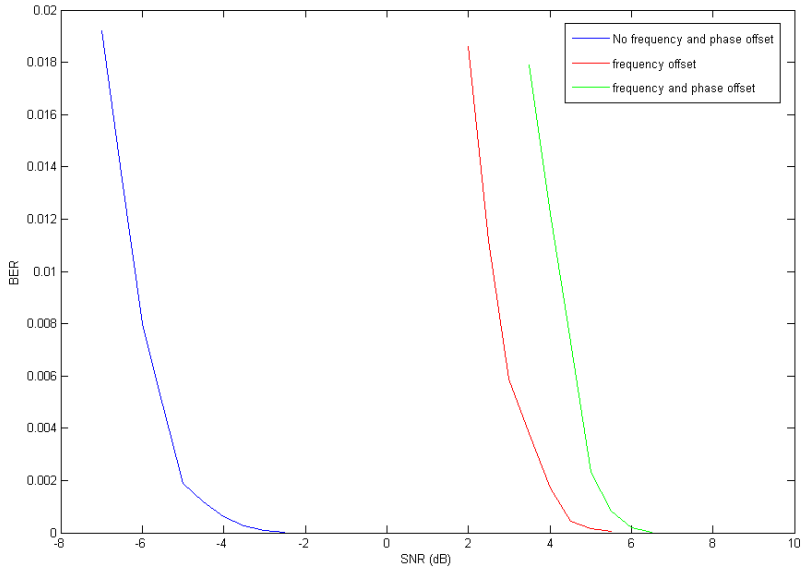


Figure 7.6: BER vs. SNR for the demodulator with 2 bits quantization before the correlator and 4 bits quantization at the input of the channel filter.

case 1, but only a 1 dB penalty in SNR for the required BER.

It was found that the deviation between simulated and computed results were acceptable for case 1 and 2, considering the low resolution in the simulation, whereas case 3 has i unacceptable deviation. The deviations probably occur from the computation of the quantization noise, because of the small number of levels. It can therefore not be assured that the quantization noise in a uniformly distributed random variable, which is assumed in the derivation of the formula for quantization noise [14].

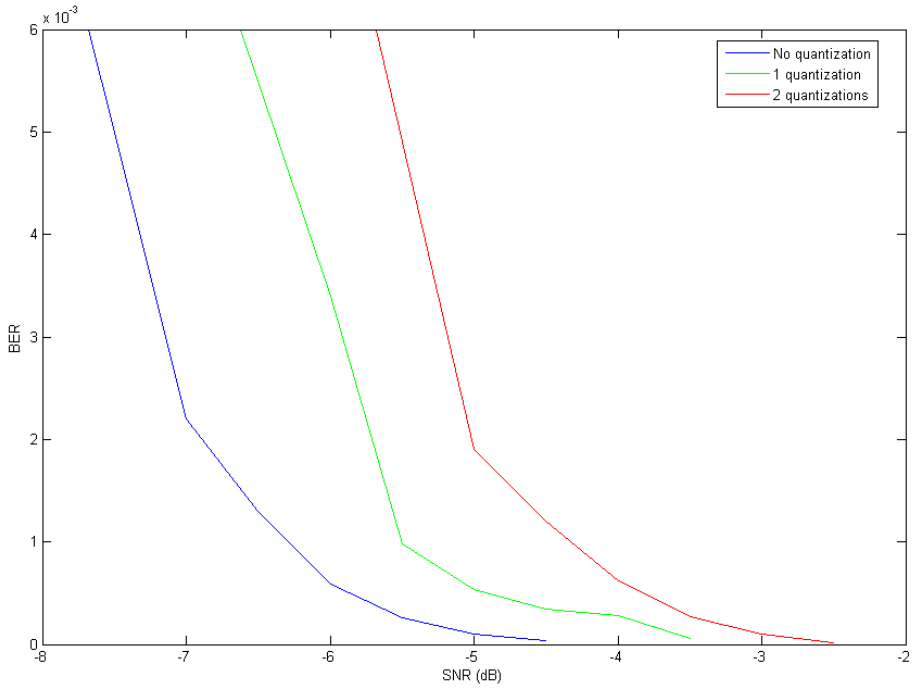


Figure 7.7: BER vs. SNR for the demodulator for the 3 cases without rotation.

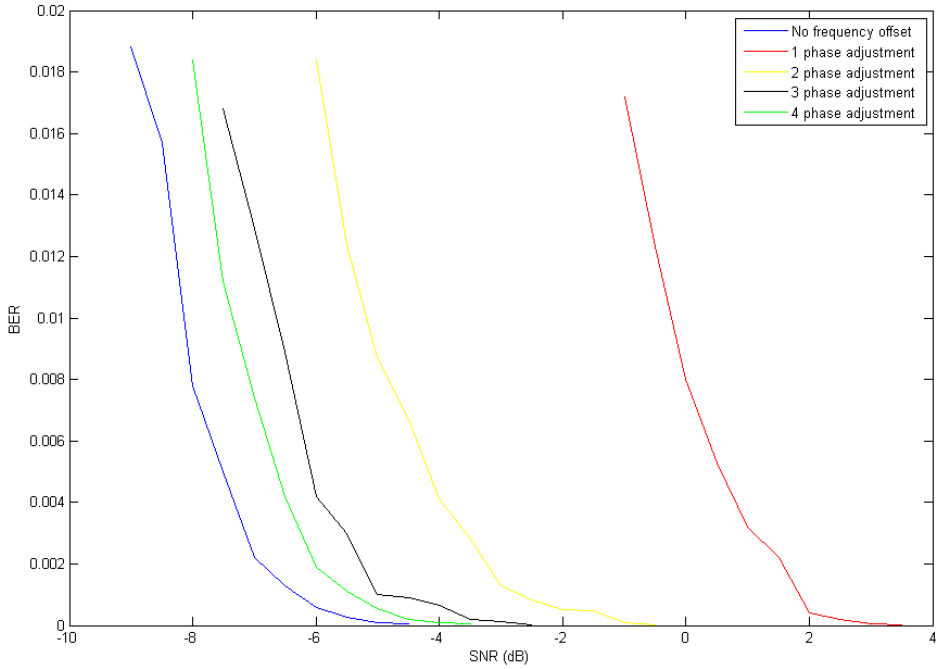


Figure 7.8: BER vs. SNR for the demodulator for packet reception with frequency offset for 1, 2, 3 and 4 phase adjustments during reception.

Chapter 8

Conclusion

Algorithms for a low complexity implementation of a IEEE 802.15.4 2.4 GHz physical layer modem have been proposed and evaluated. The proposed implementation formed a basis for a low-power, low complexity DSP architecture with an area of approximately 40000 gates and 5mA typical current consumption.

The simulations show that, if zero frequency or phase offset assumed, the 5.7×10^{-5} BER required by IEEE 802.15.4 physical layer standard can be reached for a -3.5 dB SNR at the input of the digital demodulator by using the chosen algorithms. Removing the quantizer in front of the correlator causes a big increase in the complexity, but only a 1 dB gain compared to the proposed solution. Simulations also show that the frequency and phase offset causes large degradations in the performance of the demodulator. Increasing the number of phase adjustments can give big improvements in the performance of the demodulator.

It is found that the analytic results underestimate the effect of the quantization and therefore give too optimistic results compared to the simulation.

8.1 Future Work

This thesis is only the first step on the way to realize the modem on a DSP. The modem should be simulated in mode 1 in addition to mode 2. The realization of the frequency and phase offset estimator should also be brought into the simulation. The resolution of the simulation should also be increased in order to get more reliable results for low BER. The modem may then be optimized to maximize the SNR for a given complexity restriction.

A good tool to optimize the use of resources may be to set up a time schedule which shows the time usage in the different functional blocks, as done in [32].

This thesis only considers the IEEE 802.15.4 2.4 GHz standard. It may be of interest to investigate the feasibility of implementing several standards on the same DSP. This increases the functionality at the expense of a complexity increase, although reuse of functionality may reduce the complexity so these solutions can be feasible.

Bibliography

[1] ZigBee Alliance. Zigbee specifications at http://www.zigbee.org/. 2005.

[2] M. Josie Ammer and Jan Rabaey. Frequency estimation from proper sets of correlations. *Transactions on Signal Processing*, Vol.50(No.4):791–802, 2002.

[3] M. Josie Ammer and Jan Rabaey. Frequency offset estimation with improved convergence time and energy consumption. *Symposium on Spread Spectrum Techniques and Applications*, (No.8):596–600, 2004.

[4] Ray Andranka. A survey of cordic algorithms for fpga based computer. 1998.

[5] John R. Barry, Edward A. Lee, and David G. Masserschmitt. *Digital Communication*. Kluwer Academic Publishers, 3 edition, 2004.

[6] S. Benedetto, M. Mondin, and G. Montorsi. Performance evaluation of trellis-coded modulation schemes. *IEEE Proceedings*, vol. 82(No.12):833–855, 1994.

[7] S. Bourdel, P. Pannier, H. Barthélemy, and N. Dehaese. Low-cost solutions for 802.15.4 rf. *Spread Spectrum Techniques and Applications, 2004 IEEE Eighth International Symposium on*, 2004.

[8] Chipcon. Data sheet for cc2420, 2.4 ghz ieee 802.15.4 / zigbee-ready rf transceiver. http://www.chipcon.com/files/CC2420_Data _Sheet _1 _3.pdf.

[9] F. Classen and H. Meyr. Two frequency estimation schemes operating independently of timing information. *Global Telecommunications Conference*, vol. 3:1996–2000, 1993.

[10] Ercegovac and Lang. Cordic algorithm and implementations. 2003.

[11] Michael P. Fitz. Planar filtered techniques for burst mode carrier synchronisation. *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 365–369, 1991.

[12] IEEE Standard for Information Technology. Part 802.15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs). 2003.

[13] F. Gardner. A bpsk/qpsk timing-error detector for sampled receivers. *IEEE Transactions on Communications*, Vol.34(No.5):423–429, 1986.

[14] Simon Haykin. *Communication Systems*. John Wiley & Sons, Inc., 4 edition, 2001.

[15] Peter Händel, Anders Eriksson, and Torbjörn Wigren. Performance analysis of a correlation based sigle tone frequency estimator. *Elsevier Signal Processing*, (No.44):223–231, 1995.

[16] M. Kasal J. Sebesta. Effective dsp methods of psk feedback timing. *RADIOENGINEERING*, Vol.14(No.3):37–40, 2005.

[17] David A. Johns and Ken Martin. *Analog Integrated Circuit Design*. John Wiley & Sons, Inc., 1 edition, 1997.

[18] Steven Kay. A fast and accurate single frequency offset estimator. *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol.37(No.12), 1980.

[19] Louis Litwin. Matched filtering and timing recovery in digital receivers, a practical look at methods for signal detection and symbol syncronization. 2001. http://www.rfdesign.com.

[20] E.B Loewenstein. Reducing the effects of noise in a data acquisition system by averaging. *National Instruments, Application note 152*, April 2000.

[21] Umberto Mengali and M. Morelli. Data-aided frequency estimation for burst digital transmission. *IEEE Transactions on Communications*, VOL. 45(NO. 1):833–855, 1997.

[22] Sanjit K. Mitra. *Digital Signal Processing, a Computer-Based Approach*. McGraw-Hill Higher Education, 2 edition, 2002.

[23] K. Mueller and M. Muller. Timing recovery in digital synchronous data receivers. *IEEE Transactions on Communications*, Vol.24(No.5):516–531, 1976.

[24] Trung-Kien Nguyen, Nam-Jin Oh, Viet-Hoang Le, and Sang-Gug Lee. A low-power cmos direct conversion receiver with 3-db nf and 30-khz flicker-noise corner for 915-mhz band ieee 802.15.4 zigbee standard. *IEEE Transactions on Microwave Theory and Techniques*, Vol.54:735–741, 2006.

[25] Hallvard Næss. A programmable dsp for low-power, low-complexity baseband processing. master thesis. June 2006.

[26] Sang Yoon Park and Nam Ik Cho. Fixed point error analysis of cordic processor based on the variance propagation. *Proceedings on Acoustics, Speech, and Signal Processing*, Vol.2:565–568, 2003.

[27] David M. Pozar. *Microwave and RF Design of Wireless Systems*. John Wiley & Sons, Inc., 1 edition, 2001.

[28] Theodore S. Rappaport. *Wireless Communications, Principles and Practice*. Prentice Hall Communications Engineering and Emerging Technologies Series, 2 edition, 2002.

[29] Behzad Razavi. Design considerations for direct-conversion receivers. *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol.44:428–435, 1997.

[30] J Eric Salt. Effect of filtering on the performance of qpsk and msk modulation in d-s spread spectrum systems using rake receiver. *IEEE Journal on Selected Areas in Communication*, pages 707–715, 1994.

[31] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall International Editions, 1996.

[32] Eric Tell. Design of programmable baseband processors. phd thesis. linköping university. 2005.

[33] Walter Tuttlebee. *Software Defined Radio, Enabling Technologies*. John Wiley & Sons, Ltd., 1 edition, 2002.

[34] Björn Völker and Peter Händel. On design of correlation based frequency estimators. *IEEE*, pages 425–428, 2001.

[35] Björn Völker and Peter Händel. On design of correlation based frequency estimators. *Proceedings of the Signal Processing Workshop on Statistical Signal Processing*, (No.11), 2001.

[36] Jianliang Zheng and Myung J. Lee. A comprehensive performance study of ieee 802.15.4.

Appendix A

Analytic results

A.1 Demodulator with 8 bits ADC and -4.5 dB Channel SNR

$$\sigma_s^2 = \frac{1}{2}. \tag{A.1.1}$$

$$SNR_{\text{ch}} = \frac{\sigma_s^2}{\sigma_n^2}. \tag{A.1.2}$$

Which gives the noise power

$$\sigma_n^2 = \frac{\sigma_s^2}{SNR_{\text{ch}}} = 1.409. \tag{A.1.3}$$

$$\sigma_q^2 = \frac{1}{3}1^2 2^{-16} = 5.09 \times 10^{-6}. \tag{A.1.4}$$

The ratio between channel noise and quantization noise is

$$10\log(\frac{\sigma_n^2}{\sigma_q^2}) \approx -54\,\text{dB}. \tag{A.1.5}$$

The quantization noise is approximately 54 dB smaller than the channel noise and can therefore be ignored. This gives the power spectral density at the input of the channel filter

$$S_y(F) = S_n(F). \tag{A.1.6}$$

This means that the noise variance at the output of the filter is

$$\sigma_f^2 = \int_0^{F_s/2} |H(F)|^2 S_y(F) dF = \int_0^B |H(F)|^2 N_0 dF = 0.5055, \tag{A.1.7}$$

where $B = F_s/2 = 4$ MHz and $|H(f)|^2$ is the squared magnitude spectrum of the channel filter described in section 6.1.

$$\sigma_c^2 = 3.3 \times 10^{-4}. \tag{A.1.8}$$

This formula assumes that the quantization error is a uniformly distributed random variable, which is not the case here. This is therefore a rough estimate of the quantization noise.

The total SNR before the processing gain (PG) is

$$SNR = 10\log_{10}(\frac{\sigma_s^2}{\sigma_f^2+\sigma_c^2}) = -0.05. \tag{A.1.9}$$

The total SNR of the demodulator is therefore

$$SNR_{\text{tot}} = SNR + PG + CG + 3dB = 12.75dB, \tag{A.1.10}$$

and thus

$$\frac{E_b}{N_0} = SNR_{\text{tot}} - 10\log b = 9.7dB. \tag{A.1.11}$$

This gives a 7.8×10^{-6} BER.

A.2 Demodulator with 8 bits ADC, 2 Bits Quantizer in front of the Correlator and -3.5 dB Channel SNR

$$\sigma_s^2 = \frac{1}{2}. \tag{A.2.1}$$

$$SNR_{\text{ch}} = \frac{\sigma_s^2}{\sigma_n^2}. \tag{A.2.2}$$

Which gives the noise power

$$\sigma_n^2 = \frac{\sigma_s^2}{SNR_{\text{ch}}} = 1.12. \tag{A.2.3}$$

$$\sigma_q^2 = \frac{1}{3}1^2 2^{-16} = 5.09\times10^{-6}. \tag{A.2.4}$$

The ratio between channel noise and quantization noise is

$$10\log(\frac{\sigma_n^2}{\sigma_q^2}) \approx -54\,\text{dB}. \tag{A.2.5}$$

The quantization noise is approximately 54 dB smaller than the channel noise and can therefore be ignored. This gives the power spectral density at the input of the channel filter

$$.S_y(F) = S_n(F) \tag{A.2.6}$$

This means that the noise variance at the output of the filter is

$$\sigma_f^2 = \int_0^{F_s/2} |H(F)|^2 S_y(F)dF = \int_0^B |H(F)|^2 N_0 dF = 0.402, \tag{A.2.7}$$

where $B = F_s/2 = 4$ MHz and $|H(F)|^2$ is the squared magnitude spectrum of the channel filter described in section 6.1.

$$\sigma_c^2 = 3.3 \times 10^{-4} \tag{A.2.8}$$

$$\sigma_{q2}^2 = \frac{1}{3}(2^{-4} - 2^{-16}) = 0.021. \tag{A.2.9}$$

This formula assumes that the quantization error is a uniformly distributed random variable, which is not the case here. This is therefore a rough estimate of the quantization noise.

The total SNR before the processing gain (PG) is

$$SNR = 10\log_{10}(\frac{\sigma_s^2}{\sigma_f^2 + \sigma_c^2 + \sigma_{q2}^2}) = 0.728dB. \tag{A.2.10}$$

The total SNR of the demodulator is therefore

$$SNR_{\text{tot}} = SNR + PG + CG + 3dB = 13.5dB, \tag{A.2.11}$$

and thus

$$\frac{E_b}{N_0} = SNR_{\text{tot}} - 10\log b = 10.5dB. \tag{A.2.12}$$

This gives a 1.1×10^{-6} BER.

A.3 Demodulator with 4 Bits ADC, 2 bits Quantizer in front of the Correlator and -2.5 dB Channel SNR

$$\sigma_s^2 = \frac{1}{2}, \tag{A.3.1}$$

$$SNR_{\text{ch}} = \frac{\sigma_s^2}{\sigma_n^2}. \tag{A.3.2}$$

Which gives the noise power

$$\sigma_n^2 = \frac{\sigma_s^2}{SNR_{\text{ch}}} = 0.889. \tag{A.3.3}$$

$$\sigma_q^2 = \frac{1}{3}1^2 2^{-8} = 0.0013. \tag{A.3.4}$$

This formula assumes that the quantization error is a uniformly distributed random variable, which is not the case here. This is therefore a rough estimate of the quantization noise. The ratio between channel noise and quantization noise is

$$10\log(\frac{\sigma_n^2}{\sigma_q^2}) \approx -27\,\text{dB}. \tag{A.3.5}$$

The quantization noise is approximately 27 dB smaller than the channel noise and is therefore chosen not to be ignored. This gives the power spectral density at the input of the channel filter

$$S_y(F) = S_n(F) + S_q(f). \tag{A.3.6}$$

This means that the noise variance at the output of the filter is

$$\sigma_f^2 = \int_0^{F_s/2} |H(F)|^2 S_y(F) dF = \int_0^B |H(F)|^2 N_0 dF = 0.1709, \tag{A.3.7}$$

where $B = F_s/2 = 4$ MHz and $|H(F)|^2$ is the squared magnitude spectrum of the channel filter described in section 7.1. It was found that $\int_0^{F_s/2} |H(F)|^2 = 0.1920$ in this case, whereas it was 0.1762 for the channel filter in the two previous sections. The reason is the quantization of the filter coefficients.

$$\sigma_c^2 = 3.3 \times 10^{-4}. \tag{A.3.8}$$

$$\sigma_{q2}^2 = \frac{1}{3}(2^{-4} - 2^{-8}) = 0.0193. \tag{A.3.9}$$

This formula assumes that the quantization error is a uniformly distributed random variable, which is not the case here. This is therefore a rough estimate of the quantization noise.

The total SNR before the processing gain (PG) is

$$SNR = 10 \log_{10}\left(\frac{\sigma_s^2}{\sigma_f^2 + \sigma_c^2 + \sigma_{q2}^2}\right) = 4.18 dB. \tag{A.3.10}$$

The total SNR of the demodulator is therefore

$$SNR_{\text{tot}} = SNR + PG + CG + 3dB = 17.0 dB, \tag{A.3.11}$$

and thus

$$\frac{E_b}{N_0} = SNR_{\text{tot}} - 10 \log b = 14.0 dB. \tag{A.3.12}$$

This gives a 6.8×10^{-13}BER.

Appendix B

Matlab Code for the Simulation

```
%**********************************The MoDem*****************************

function [number_of_errors,bit_error_rate]=Simulasjon_hele(snr,Phase)
%%---------------------------------Set parameters-----------------------
M = 4; % Size of signal constellation
k = log2(M); % Number of bits per channel symbol
n = 1016; % Number of bits to process (size of the maximum packet)

%%---------------------------------Signal Source------------------------
% Random number generator
x = randint(n,1);

%%---------------------------------Modulator----------------------------
%%Bit-to-symbol mapping
xsmb=reshape(x,4,length(x)/4).';

%%symbol-to-chip mapping in bits
[l1,l2]=size(xsmb);
for l=1:l1
    [yI(l,:),yQ(l,:)]=mapping(xsmb(l,:));
end
yI=yI';
yI=yI(:);
yQ=yQ';
yQ=yQ(:);

%QPSK modulation
yI=yI*2 - 1; % I branch mapped from 0/1 to -1/1
yQ=yQ*2 - 1; % Q branch mapped from 0/1 to -1/1
y=yI+j*yQ;
```

```
%Gard interval
y=[0 0 y' 0 0]';

% Upsample and apply the 8 tap half-sine pulse-shaping filter.
y=upsample(y,8);
ytx=shape2(y);

%%---------------------------------Channel-------------------------------
% Send signal over an AWGN channel.
ynoisy = awgn(ytx,snr,'measured');

%%---------------------------------Demodulator---------------------------
%quantize the input of the demodulator down to 4 bits before the channel
%filter for case 3. Equivalent to using 4 bits ADC.
ynoisy=kvantiser4(real(ynoisy))+j*kvantiser4(imag(ynoisy));

%Channel filter used in case 1 and 2
% yrx = kanalfilter(ynoisy);

%Channel filter is used in case 3. It has approximately twice times as
%many taps as the filter used in case 1 and 2.
yrx = kanalfilter2(ynoisy);

% Downsample the output of the channel filter
yrx=downsample(yrx,2,1);
k=length(yrx);

offset=freqoffset(snr); %Get the frequency offset for the given SNR
for r=1:k
    if r<k/5
        n=(r *offset)+Phase; %Phase offset at time instance r
        yrx(r)=yrx(r)*exp(j*n);
    end
    if r>=k/5 & r<=2*k/5
        n=((r-k/5) *offset)+Phase; %Phase offset at time instance r,
                    %with compensated phase at 1/5 the packet length
        yrx(r)=yrx(r)*exp(j*n);
    end
    if r>=2*k/5 & r<=3*k/5
        n=((r-2*k/5) *offset)+Phase; %Phase offset at time instance r,
                      %with compensated phase at 2/5 the packet length
        yrx(r)=yrx(r)*exp(j*n);
```

```
    end
    if r>=3*k/5 & r<=4*k/5
        n=((r-3*k/5) *offset)+Phase; %Phase offset at time instance r,
                        %with compensated phase at 3/5 the packet length
        yrx(r)=yrx(r)*exp(j*n);
    end
    if r>=4*k/5
        n=((r-4*k/5) *offset)+Phase; %Phase offset at time instance r,
                       % with compensated phase at 4/5 the packet length
        yrx(r)=yrx(r)*exp(j*n);
    end
end


% Matched filter. Apply the 4 tap half-sine pulse-shaping filter.
yrx = shape(yrx);

%Frame synchronization
%For Case 3
yrx = yrx(17:end);
%For Case 1 and 2
% yrx = yrx(14:end-3);

% Divide into two paths
yrxI= real(yrx);
yrxQ =imag(yrx);

%Quantize to 2 bits for case 2 and 3
yrxI=kvantiser(yrxI);
yrxQ=kvantiser(yrxQ);

%Group the sequence so the correlator can take in one and one symbol
l=length(yrx)/64;
yrxI=reshape(yrxI,64,l).';
yrxQ=reshape(yrxQ,64,l).';

%Demodulate the signal in the correlator
for r=1:l
z(r,:)=korrelator2(yrxI(r,:),yrxQ(r,:));
end

%reshape the matrix to a bitstream.
z;
```

```
z=z';
z=z(:);

%%--------------------------------BER Computation-----------------------
% Compare x and z to obtain the number of errors and the bit error rate.
[number_of_errors,bit_error_rate] = biterr(x,z);


%*************************The Symbol-to-Chip Mapping*********************
function [yI,yQ]=mapping(x)

%The 0-symbol. All other symbols are given by shifts or inversions
seq=[1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0];

%Divide into the I and Q information
seqI=seq(1:2:end);
seqQ=seq(2:2:end);

if x==[0 0 0 0]  %0-symbol
    yI=seqI;
    yQ=seqQ;
elseif x==[1 0 0 0] %1-symbol
        yI=circshift(seqI,[0 2]);
        yQ=circshift(seqQ,[0 2]);
elseif x==[0 1 0 0] %2-symbol
        yI=circshift(seqI,[0 4]);
        yQ=circshift(seqQ,[0 4]);
elseif x==[1 1 0 0] %3-symbol
        yI=circshift(seqI,[0 6]);
        yQ=circshift(seqQ,[0 6]);
elseif x==[0 0 1 0] %4-symbol
        yI=circshift(seqI,[0 8]);
        yQ=circshift(seqQ,[0 8]);
elseif x==[1 0 1 0] %5-symbol
        yI=circshift(seqI,[0 10]);
        yQ=circshift(seqQ,[0 10]);
elseif x==[0 1 1 0] %6-symbol
        yI=circshift(seqI,[0 12]);
        yQ=circshift(seqQ,[0 12]);
elseif x==[1 1 1 0] %7-symbol
        yI=circshift(seqI,[0 14]);
        yQ=circshift(seqQ,[0 14]);
elseif x==[0 0 0 1] %8-symbol
```

```
    yI=seqI;
    yQ=~seqQ;
elseif  x==[1 0 0 1] %9-symbol
        yI=circshift(seqI,[0 2]);
        yQ=~circshift(seqQ,[0 2]);
elseif  x==[0 1 0 1] %10-symbol
        yI=circshift(seqI,[0 4]);
        yQ=~circshift(seqQ,[0 4]);
elseif  x==[1 1 0 1] %11-symbol
        yI=circshift(seqI,[0 6]);
        yQ=~circshift(seqQ,[0 6]);
elseif  x==[0 0 1 1] %12-symbol
        yI=circshift(seqI,[0 8]);
        yQ=~circshift(seqQ,[0 8]);
elseif  x==[1 0 1 1] %13-symbol
        yI=circshift(seqI,[0 10]);
        yQ=~circshift(seqQ,[0 10]);
elseif  x==[0 1 1 1] %14-symbol
        yI=circshift(seqI,[0 12]);
        yQ=~circshift(seqQ,[0 12]);
elseif  x==[1 1 1 1] %15-symbol
        yI=circshift(seqI,[0 14]);
        yQ=~circshift(seqQ,[0 14]);
end

%************************%8 tap half-sine pulse-shaping filter***********
function y = shape2(x);
%Filter coefficients from a sampled half-sine
filt = [0 0.3827  .7071  0.9239 1  0.9239 .7071 0.3827];
y = filter(filt,1,x) ;

%************************%4 bits Quantizer*****************************
function [w] = kvantiser4(y)
%Returns the 4 bits quantized value

%partition
q = [-0.9334 -0.8000 -0.6667 -0.5333 -0.4000 -0.2667 -0.1334 0 ...
    0.1334 0.2667 0.4000 0.5333 0.6667 0.8000 0.9334];

%codebook
c = [-1.0000 -0.8667 -0.7333 -0.6000 -0.4667 -0.3333 -0.2000 -0.0667 ...
    0.0667 0.2000 0.3333 0.4667 0.6000 0.7333 0.8667 1.0000];
```

```
[index,w] = quantiz(y,q,c);

%************************%4 bits Channel Filter for Case 3***************
function y = kanalfilter2(x)
%Returns the low-pass filtered signal.

% All frequency values are in KHz.
Fs = 8000;  % Sampling Frequency

N     = 23;    % Order
Fpass = 820;   % Passband Frequency
Fstop = 1520;  % Stopband Frequency
Wpass = 1;     % Passband Weight
Wstop = 1;     % Stopband Weight
dens  = 20;    % Density Factor

% Calculate the coefficients using the FIRPM function.
bb  = firpm(N, [0 Fpass Fstop Fs/2]/(Fs/2), [1 1 0 0], [Wpass Wstop], ...
           {dens});

b=kvantiser4(bb); %4 bit resolution on the coefficients
y=filter(b,1,x);

%************************%Unquantized Channel Filter for Case 1 and 2****
function y= kanalfilter(x);
%Returns the low-pass filtered signal.

% All frequency values are in kHz.
Fs = 8000;  % Sampling Frequency

N     = 10;    % Order
Fpass = 820;   % Passband Frequency
Fstop = 2500;  % Stopband Frequency
Wpass = 1;     % Passband Weight
Wstop = 1;     % Stopband Weight
dens  = 20;    % Density Factor

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, [0 Fpass Fstop Fs/2]/(Fs/2), [1 1 0 0], [Wpass Wstop], ...
           {dens});

y=filter(b,1,x);
```

```
%************************%Frequency Offset*****************************
function offset=freqoffset(snr)
%Returns a frequency offset as a function of SNR
offset=sqrt(4*10^-8 -(1.98e-009)*(snr+9.8));

%************************%4 tapped Matched Filter ***********************
function y = shape(x);
%Create the 4 tap half-sine pulse-shaping filter
filt = [0 .7071 1 .7071 ];
y = filter(filt,1,x) ;


%************************%2 bits quantizer ************************

function [w] = kvantiser(y)
%Returns the 2 bits quantized value

%partition
q = [-0.5 0.5];

%codebook
c = [-1 0 1];
[index,w] = quantiz(y,q,c);

%************************%Correlator ************************
function det= korrelator2(xI,xQ)

%The 0-symbol
R = [1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0];
R = 2*R - 1;
%-----------------The I Correlator Information----------------------------
rI = R(1:2:end);

%The I branch information of the
rI1 = rI; % 0/8-symbol
rI2 = circshift(rI,[0 2]); % 1/9-symbol
rI3 = circshift(rI,[0 4]); % 2/10-symbol
rI4 = circshift(rI,[0 6]); % 3/11-symbol
rI5 = circshift(rI,[0 8]); % 4/12-symbol
rI6 = circshift(rI,[0 10]);% 5/13-symbol
rI7 = circshift(rI,[0 12]);% 6/14-symbol
rI8 = circshift(rI,[0 14]);% 7/15-symbol
```

```
%-----------------The Q Correlator Information--------------------------
rQ=R(2:2:end);

% rQ=rQ*2 -1; %0/1 to -1/1 mapping
rQ1=rQ; %0-symbol
rQ2=circshift(rQ,[0 2]); %1-symbol
rQ3=circshift(rQ,[0 4]); %2-symbol
rQ4=circshift(rQ,[0 6]); %3-symbol
rQ5=circshift(rQ,[0 8]); %4-symbol
rQ6=circshift(rQ,[0 10]);%5-symbol
rQ7=circshift(rQ,[0 12]);%6-symbol
rQ8=circshift(rQ,[0 14]);%7-symbol
rQ9=-rQ1;                %8-symbol
rQ10=-rQ2;                   %9-symbol
rQ11=-rQ3;                   %10-symbol
rQ12=-rQ4;               %11-symbol
rQ13=-rQ5;               %12-symbol
rQ14=-rQ6;               %13-symbol
rQ15=-rQ7;               %14-symbol
rQ16=-rQ8;               %15-symbol

%-----------------The I Correlator ---------------------------------------
%distribute the incoming sequence on the 4 branches for
%finding best timing
xI1=xI(1:4:end);
xI2=xI(2:4:end);
xI3=xI(3:4:end);
xI4=xI(4:4:end);

detI1(1)=sum (xI1 .* rI1);
detI1(2)=sum (xI2 .* rI1);
detI1(3)=sum (xI3 .* rI1);
detI1(4)=sum (xI4 .* rI1);
det(1)=max(detI1);            % 0/8-symbol Correlator value

detI2(1)=sum (xI1 .* rI2);
detI2(2)=sum (xI2 .* rI2);
detI2(3)=sum (xI3 .* rI2);
detI2(4)=sum (xI4 .* rI2);
det(2)=max(detI2);            % 1/9-symbol Correlator value

detI3(1)=sum (xI1 .* rI3) ;
```

```
detI3(2)=sum (xI2 .* rI3) ;
detI3(3)=sum (xI3 .* rI3) ;
detI3(4)=sum (xI4 .* rI3) ;
det(3)=max(detI3);            % 2/10-symbol Correlator value

detI4(1)=sum (xI1 .* rI4) ;
detI4(2)=sum (xI2 .* rI4) ;
detI4(3)=sum (xI3 .* rI4) ;
detI4(4)=sum (xI4 .* rI4) ;
det(4)=max(detI4);            % 3/11-symbol Correlator value

detI5(1)=sum (xI1 .* rI5) ;
detI5(2)=sum (xI2 .* rI5) ;
detI5(3)=sum (xI3 .* rI5) ;
detI5(4)=sum (xI4 .* rI5) ;
det(5)=max(detI5);            % 4/12-symbol Correlator value

detI6(1)=sum (xI1 .* rI6) ;
detI6(2)=sum (xI2 .* rI6) ;
detI6(3)=sum (xI3 .* rI6) ;
detI6(4)=sum (xI4 .* rI6) ;
det(6)=max(detI6);            % 5/13-symbol Correlator value

detI7(1)=sum (xI1 .* rI7) ;
detI7(2)=sum (xI2 .* rI7) ;
detI7(3)=sum (xI3 .* rI7) ;
detI7(4)=sum (xI4 .* rI7) ;
det(7)=max(detI7);            % 6/14-symbol Correlator value

detI8(1)=sum (xI1 .* rI8) ;
detI8(2)=sum (xI2 .* rI8) ;
detI8(3)=sum (xI3 .* rI8) ;
detI8(4)=sum (xI4 .* rI8) ;
det(8)=max(detI8);            % 7/15-symbol Correlator value


%Find biggest correlation value
maks = find(det==max(det));
s=length(maks);

%If more than 1 value is the biggest, choose randomly.
if s>=2
    maks=maks(ceil(rand()*s));
```

```
end

%-----------------The Q Correlator -------------------------------------
% xQ=xQ*2 -1; %0/1 to -1/1 mapping

%distribute the incoming sequence on the 4 branches for finding best
%timing
xQ1=xQ(1:4:end);
xQ2=xQ(2:4:end);
xQ3=xQ(3:4:end);
xQ4=xQ(4:4:end);

%Check the sign of correlation value of Q-symbol number (maks-1)
if maks ==1
    detQ_sum(1)=sum(xQ1 .* rQ1);
    detQ_sum(2)=sum(xQ2 .* rQ1);
    detQ_sum(3)=sum(xQ3 .* rQ1);
    detQ_sum(4)=sum(xQ4 .* rQ1);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum))))) ;
elseif maks ==2
    detQ_sum(1)=sum(xQ1 .* rQ2);
    detQ_sum(2)=sum(xQ2 .* rQ2);
    detQ_sum(3)=sum(xQ3 .* rQ2);
    detQ_sum(4)=sum(xQ4 .* rQ2);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum)))));
elseif maks ==3
    detQ_sum(1)=sum(xQ1 .* rQ3);
    detQ_sum(2)=sum(xQ2 .* rQ3);
    detQ_sum(3)=sum(xQ3 .* rQ3);
    detQ_sum(4)=sum(xQ4 .* rQ3);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum)))));
elseif maks ==4
    detQ_sum(1)=sum(xQ1 .* rQ4);
    detQ_sum(2)=sum(xQ2 .* rQ4);
    detQ_sum(3)=sum(xQ3 .* rQ4);
    detQ_sum(4)=sum(xQ4 .* rQ4);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum)))));
elseif maks ==5
    detQ_sum(1)=sum(xQ1 .* rQ5);
    detQ_sum(2)=sum(xQ2 .* rQ5);
    detQ_sum(3)=sum(xQ3 .* rQ5);
    detQ_sum(4)=sum(xQ4 .* rQ5);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum)))));
```

```
elseif maks ==6
    detQ_sum(1)=sum(xQ1 .* rQ6);
    detQ_sum(2)=sum(xQ2 .* rQ6);
    detQ_sum(3)=sum(xQ3 .* rQ6);
    detQ_sum(4)=sum(xQ4 .* rQ6);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum)))));
elseif maks ==7
    detQ_sum(1)=sum(xQ1 .* rQ7);
    detQ_sum(2)=sum(xQ2 .* rQ7);
    detQ_sum(3)=sum(xQ3 .* rQ7);
    detQ_sum(4)=sum(xQ4 .* rQ7);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum)))));
elseif maks ==8
    detQ_sum(1)=sum(xQ1 .* rQ8);
    detQ_sum(2)=sum(xQ2 .* rQ8);
    detQ_sum(3)=sum(xQ3 .* rQ8);
    detQ_sum(4)=sum(xQ4 .* rQ8);
    detQ=sign(detQ_sum(find(max(abs(detQ_sum)))));
end


if length(detQ)==0
    detQ=0 ;
end

%The signum function is not defined for 0
if detQ==0
    detQ=2*randint()-1 ;
end

%Detect symbol
if detQ==1
    det1=maks-1 ;
elseif detQ==-1
    det1=maks+7 ;
end

%Mapp from symbol number to bit values and return the detected
if det1==0
    det=[0 0 0 0];
elseif det1==1
    det=[0 0 0 1];
elseif det1==2
```

```
    det=[0 0 1 0];
elseif det1==3
    det=[0 0 1 1];
elseif det1==4
    det=[0 1 0 0];
elseif det1==5
    det=[0 1 0 1];
elseif det1==6
    det=[0 1 1 0];
elseif det1==7
    det=[0 1 1 1];
elseif det1==8
    det=[1 0 0 0];
elseif det1==9
    det=[1 0 0 1];
elseif det1==10
    det=[1 0 1 0];
elseif det1==11
    det=[1 0 1 1];
elseif det1==12
    det=[1 1 0 0];
elseif det1==13
    det=[1 1 0 1];
elseif det1==14
    det=[1 1 1 0];
elseif det1==15
    det=[1 1 1 1];
end

det=fliplr(det);
```