Software-Defined Radio for Wireless Local-Area Networks

Roel Schiphorst

De promotiecommissie:

voorzitter en secretaris:	
Prof.dr. W.H.M. Zijm	Universiteit Twente
promotor:	
Prof.dr.ir. C.H. Slump	Universiteit Twente
leden:	
Prof.dr.ir. B. Nauta	Universiteit Twente
Prof.dr.ir. J.C. Haartsen	Universiteit Twente
Prof.dr.ir. J.W.M. Bergmans	TU Eindhoven
Prof.dr.ir. A.-J. van der Veen	TU Delft
Prof.dr.ir. Th. Krol	Universiteit Twente

University of Twente
The Netherlands

This research is conducted within the SDR project (TES.5177) supported by the PROGram for Research on Embedded Systems & Software (PROGRESS) of the Dutch Ministry of Economic Affairs and the technology foundation STW.

Signals & Systems group P.O. Box 217, 7500 AE Enschede, the Netherlands

Print: Wöhrmann Print Service
Typesetting: LaTeX2e

ISSN 1381-3617 (CTIT Ph.D.-thesis series No. 04-65)
ISBN 90-365-2071-1

SOFTWARE-DEFINED RADIO FOR WIRELESS LOCAL-AREA NETWORKS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 10 september 2004 om 15.00 uur

door

Roelof Schiphorst
geboren op 6 december 1976
te Havelte

Dit proefschrift is goedgekeurd door de promotor:

Prof.dr.ir. C.H. Slump

Samenvatting

Nieuwe draadloze standaarden vervangen geen oude, in plaats daarvan blijft het aantal standaarden stijgen en is er op dit moment een overvloed aan standaarden ontstaan. Bovendien is er geen reden om aan te nemen dat deze trend ooit zal stoppen. Daarom wordt het *software-radio concept* steeds meer gezien als een potentiële pragmatische oplossing: een software-matige implementatie van de gebruikershandset die geschikt is om zich dynamisch aan te passen aan de radio-omgeving waarin de handset is gelokaliseerd [1]. Voor fabrikanten kan dit resulteren in een kortere ontwikkeltijd en goedkopere productie door hogere volumes. Bovendien heeft software-radio voordelen voor consumenten omdat door middel van alleen software updates nieuwe functionaliteit kan worden verkregen zonder dat er nieuwe hardware nodig is.

Omdat radiosignalen analoog zijn van natuur, zal een software radio altijd een analoog deel hebben. In een ideale software radio zijn de analoog-naar-digitaal omzetter en digitaal-naar-analog omzetter direct gepositioneerd achter the antenne. Echter deze implementatie is niet maakbaar vanwege het vermogen dat zo'n apparaat zal verbruiken en andere natuurkundige beperkingen [2]. Het is daarom een uitdaging om een systeem te onwerpen wat de meeste eigenschappen van een ideale software radio heeft, maar tegelijkertijd, gemaakt kan worden met de technologie van vandaag. Zo'n systeem wordt een software-defined radio (SDR) genoemd.

Een flexibele radio die alle standaarden ondersteund, zal meer energie verbruiken dan een radio die toegespitst is op een enkele standaard. De eerste flexibele radio kan een toepassing zijn waarbij vermogensverbruik minder belangrijk is. Een voorbeeld is een flexibele radio in een notebook. Dit proefschrift beschrijft zo'n radio met de nadruk op draadloze netwerkstandaarden (wireless LAN). Deze standaarden gebruiken fasemodulatie of OFDM in de 2.4 GHz of 5 GHz frequentieband. Daarom is er binnen het project besloten om eerst een ontvanger van een fasemodulatie standaard (Bluetooth) te combineren met een OFDM ontvanger (HiperLAN/2). SDR is in onze visie een implementatie-technologie: de HiperLAN/2 ontvanger is dermate complex vergeleken met de Bluetooth onvanger dat de Bluetooth onvanger

gecombineerd kan worden met de HiperLAN/2 ontvanger tegen beperkte kosten.

In dit proefschrift wordt beschreven hoe op een functioneel niveau, de fysieke laag van het OSI model van een fasemodulatie ontvanger gecombineerd kan worden met die van een OFDM ontvanger. Het tweede doel van dit proefschrift is om te onderzoeken in hoeverre deze gecombineerde ontvanger kan worden geïmplementeerd op een generieke processor (GPP). Daarbij zijn de kosten op het gebied van vermogensverbruik en rekenkracht voor zo'n implementatie van belang. Om deze redenen is er een proefopstelling gebouwd waarbij alle basisbandfuncties van de fysieke laag succesvol zijn geïmplementeerd op een Pentium 4 processor. Dit is getest in samenhang met een breedbandig CMOS geïntegreerd analoog SDR front-end, welke een low-noise versterker, down-conversion mixers en filters bevat. De gebouwde proefopstelling kan uitgebreid worden naar andere standaarden omdat de enige beperkingen zijn de maximale bandbreedte van 20 MHz, het dynamisch bereik van het analoge front-end en de beschikbare rekenkracht van de gebruikte PC.

In 1999, heeft Bose [3] voor tweede generatie mobiele standaarden laten zien dat het mogelijk is om de meeste basisbandfuncties uit te voeren op een generieke processor. In zijn onderzoek was I/O een van de grootste bottlenecks. In dit proefschrift hebben we geïdentificeerd dat dit niet langer geldig is voor hedendaagse computers. Op dit moment is een belangrijke bottleneck de latentietijd (latency) van het besturingssysteem, omdat draadloze netwerkstandaarden snelle responsie-tijden vereisen. Belangrijke bijdragen van dit proefschrift aan het SDR onderzoek zijn het werk van Bose voortzetten voor draadloze netwerkstandaarden en het ontwikkelen van een functionele SDR ontvanger-architectuur dat geschikt is voor het ontvangen van standaarden die OFDM of fasemodulatie gebruiken.

Summary

New wireless communications standards do not replace old ones, instead the number of standards keeps on increasing and by now an abundance of standards exists. Moreover there is no reason to assume that this trend will ever stop. Therefore, the *software-radio concept* is emerging as a potential pragmatic solution: a software implementation of the user terminal able to dynamically adapt to the radio environment in which the terminal is located [1]. For manufacturers this could result in shorter development time, cheaper production due to higher volumes. Furthermore software radio has advantages for consumers because it enables only software updates for new functionality without new hardware.

Because of the analog nature of the air interface, a software radio will always have an analog front-end. In an ideal software radio, the analog-to-digital converter (ADC) and the digital-to-analog converter (DAC) are positioned directly after the antenna. Such an implementation is not feasible due to the power that such a device would consume and other physical limitations [2]. It is therefore a challenge to design a system that preserves most properties of the ideal software radio while being realizable with current-day technology. Such a system is called a software-defined radio (SDR).

A flexible, all-standard radio will consume more energy than a dedicated radio for a single standard. The first flexible radio could be an application where power consumption is less an issue; an example being a flexible radio in a notebook. This thesis presents such a radio with the focus on wireless LAN standards. These standards use phase modulation or OFDM in the 2.4 GHz or 5 GHz frequency band, so we decided in our project to combine a receiver of a phase modulation standard (Bluetooth) with an OFDM receiver (HiperLAN/2) first. In our view SDR is an implementation technology: the HiperLAN/2 hardware is that complex to the Bluetooth hardware that the Bluetooth receiver may be added to the HiperLAN/2 one at limited costs.

In this thesis we describe how the physical layer of the OSI model of a phase-modulation receiver can be combined with an OFDM receiver at a functional level. The second goal of this research is to investigate whether this combined receiver can be implemented in software running on a General

Purpose Processor (GPP) and estimate the costs of such an implementation with respect to power consumption and computational power requirements. For this reason, a testbed has been developed where all baseband physical layer functions have been successfully mapped on a Pentium 4 processor. This has been tested in combination with a CMOS-integrated wideband analog SDR front-end, containing a low noise amplifier, down-conversion mixers and filters. The testbed can be extended to other standards, because the only limitations in our testbed are the maximal channel bandwidth of 20 MHz, the dynamic range of the wideband SDR analog front-end and the processing capabilities of the used PC.

In 1999, Bose [3] showed for second-generation mobile standards that it is possible to perform most baseband functions of the physical layer on a GPP processor. In his research, I/O was one of the main bottlenecks. In thesis we have identified that this is no longer true for current-day computers. Nowadays, an important bottleneck is the latency of the operating system, because wireless standards require fast response times. Major contributions of this thesis to the field of SDR research are extending the work of Bose for wireless LAN standards and developing a functional SDR receiver architecture that is capable of receiving standards that use OFDM or phase modulation.

Contents

Chapter 1

Introduction

Since the introduction of second-generation mobile communication systems (e.g. GSM), wireless communication has become a major business. Besides the success of mobile communication systems, also other types of wireless communication such as wireless Local-Area Networks (LANs) and cordless telephone have become popular.

In the research of wireless radio communication one can identify two areas. The first is the theoretical view on radio communication that starts with a message which has to be transfered wirelessly from point A to B. Between these points there is a noisy channel which characteristics depend on the frequency band, environment, etc. In addition, there are several constraints of the system, like bandwidth, power and robustness. In this type of research, one analyzes the communication problem, simulates different system-level solutions for various environments and user scenarios. Moreover, one defines new radio algorithms and analyzes the pertinent properties.

On the other hand, the industry has to design, engineer and build wireless systems. Wireless communication systems have become popular and the industry has defined communication standards in order to manufacture and sell products. Due to the worldwide competition and other factors, this has led to a "jungle" of standards. In this field of engineering-oriented research, one tries to invent smart solutions which can cope efficiently with this abundance of standards.

This thesis contributes to the latter type of research. As each wireless communication standard needs its own radio, the Software-Defined Radio (SDR) concept is emerging as a potential pragmatic solution: a software implementation of the user terminal, able to dynamically adapt to the radio environment in which the terminal is located, given the user demands at hand [1]. Current practice, however, for a multi-standard receiver is to use separate radio chips in parallel. In the research that is described in this thesis, we have investigated whether we can design a flexible receiver for use in a notebook with the focus on the physical layer of current-day wireless LAN and Personal Area Network (PAN) standards. These standards use phase modulation or Orthogonal Frequency Division Multiplexing (OFDM) in the 2.4 GHz or 5 GHz frequency band. So, we decided in our project not to build an all-standard receiver, but to combine an instance of a phase modulation standard (Bluetooth) with an OFDM standard (HiperLAN/2) first. In addition, we want to generate an SDR design that is feasible for a product within a few years with respect to manufacturing costs and power consumption. Due to *Moore's* law [4], current General Purpose Processor (GPP) have relatively large processing capabilities. Therefore, a third goal in this research is to investigate in which extent we can use the notebook's GPP processor (e.g. Pentium 4) for digital signal processing purposes. Moreover, this solution saves hardware and *Moore's* law will lower in time the computational load as a percentage of the computational capacity.

Main research questions that we answer in this thesis are:

- How to integrate a phase-modulation receiver and an OFDM receiver?
- To what extent can we use the notebook's CPU for our purposes or do we need other digital signal processing platforms?
- Identify the best domain for performing an SDR function (analog/digital)?
- How to integrate the support of future standards?

1.1 Thesis outline

This chapter introduces definitions of software-defined radio. After these definitions, typical applications for SDR will be discussed i.e. the *why* question is answered. Moreover, this chapter describes our SDR project and compares it with other projects.

Chapter 2 discusses the Bluetooth standard and the functional architecture of both the transmitter and receiver. Chapter 3 describes the HiperLAN/2 standard in a similar manner.

Chapter 4 introduces at a functional level, an SDR receiver architecture which is suitable for OFDM and phase modulation standards. In addition, this chapter describes several implementation alternatives for this receiver.

The second goal of this research is to investigate whether the physical layer can be implemented in software running on a General Purpose Processor (GPP), e.g. Pentium 4, and estimate the costs of such an implementation with respect to power consumption and computational power requirements. In Chapter 5, both the hardware and software of the testbed are described. In order to estimate the computational power requirements for both standards, this chapter also introduces user scenarios.

Chapter 6 reports experiments conducted with the testbed and Chapter 7 evaluates the built SDR radio, draws conclusions and gives recommendations.

1.2 What is a software-defined radio?

There is not one unique definition for the software (defined) radio concept. The most common definitions are summed up below and quoted from [5]:

1. Flexible transceiver architecture, controlled and programmable by software.
2. Digital signal processing able to replace, as much as possible, radio functionalities.
3. Air-interface downloadability: radio equipment dynamically reconfigurable by downloadable software at every level of the protocol stack.
4. Software realization of terminal (multi mode/standard).
5. Transceiver where the following can be defined by software:
   - frequency band and radio channel bandwidth
   - modulation and coding scheme
   - radio resource and mobility management protocols
   - user applications

In our view on SDR, we agree with the latter definition (nr. 5). However, because we aim to build an SDR for a notebook we can use the existing digital processing capabilities of the notebook. So, we also agree in some extent with the fourth definition of SDR.

The SDR Forum (SDRF)[1] [6] defines Software-Defined Radio (SDR) as follows: *A radio that provides software control of a variety of modulation techniques, wide-band or narrow-band operation, communications security functions (such as hopping), and waveform requirements of current and evolving standards over a broad frequency range* [6]. Thus, an SDR is capable of receiving and transmitting multiple standards. This definition is very similar to the fifth definition on SDR. In addition, the SDRF has developed a reference model [7] which is depicted in Figure 1.1.

This model divides the radio into smaller units and defines only the interfaces between them. Each unit has both a software interface and a hardware

[1] The SDR Forum is an international association of 100+ organizations committed to promoting the development, deployment and use of software-defined radio.

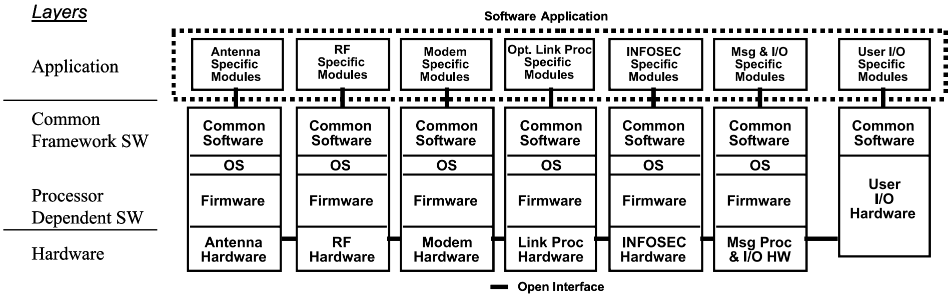


Figure 1.1: SDRF Multiprocessor Software Reference Model [7].

interface. In this way, an SDR can be made up from units of different manufacturers. In our project we did not use this reference model, because we have a different view on SDR. However, this thesis uses the definitions of the SDRF for describing the different forms of SDR.

The SDR Forum discriminates between different levels of flexibility in a radio:

- **Hardware Radio (HR)**: The functionality of this radio is fixed, so system attributes cannot be changed. However, this radio *can* use internally software as long as it cannot be changed externally.

- **Software-Controlled Radio (SCR)**: In this radio only the control functions are implemented in software, an example being the transmitted power level. Current radio design are often SCRs.

- **Software-Defined Radio (SDR)**: These radios provide software control over almost every radio function, including modulation technique, frequency band and multiple access method. In addition, a SDR is allowed to have a switch in the antenna system to cover multiple frequency bands.

- **Ideal Software-Defined Radio (ISDR)**: This radio has the same functionality as the Software-Defined Radio, but it does not have an analog front-end (amplification, mixers), thereby eliminating analog noise and distortions. So, the analog front-end only contains an antenna and the Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) are directly attached to it.

- **Ultimate Software Radio (USR)**: The USR is an ideal, utopical, flexible radio: a small, lightweight, low-power device which is fully programmable.

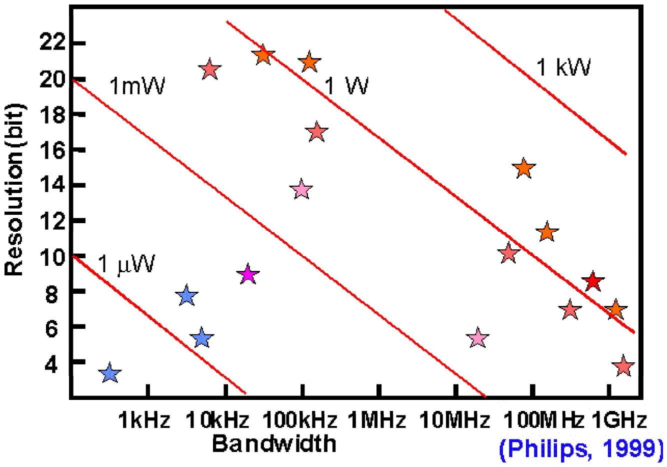


Figure 1.2: Commercially available ADCs in 1999.

The term Software Radio (SR) is used as a generic term, which covers all forms of flexible radio, i.e. SCR, SDR, ISDR and USR.

Because of the analog nature of the air interface, an SR will always have an analog front-end. In ISDR and USR radios, the ADC and the DAC are positioned directly after the antenna. Such an implementation is not feasible with current technology due to the power that such device would consume and other physical limitations [2] [8].

To illustrate this infeasibility, Figure 1.2 depicts the resolution versus bandwidth for commercially available ADCs in 1999. One can see that for a certain power consumption, there is a tradeoff between resolution and bandwidth. A ISDR and USR radio would require a sample rate of more than 8 GHz with a resolution of $\approx$ 12 bits [9]. Extrapolating the values from Figure 1.2, an ADC positioned directly after the antenna would consume several kWs. This is far too much for mobile applications[2].

Moreover, the progress in AD converters with respect to power consumption is slow, about 1.5 bit in 8 years at the same power level (at the same sample frequency) [2]. It is for that reason that the SDR concept is the most promising alternative; a flexible radio which is realizable with current-day technology.

1.2.1 Trends

There are several trends in wireless digital communication that enable the use of SDR. The first trend is that due to Moore's law [4], [10] more functionality

[2] A typical power level for the receiver front-end of a mobile telephone is about 10 mW.

of the radio transceiver is implemented in the digital domain. The chip area necessary for the analog part remains about the same in every new Integrated Circuit (IC) process generation, whereas the digital part is scaled down. Thus, to reduce chip area it is efficient to implement more and more functions of the transceiver digital and it is easier to make a digital radio flexible than an analog one.

Moreover, the designers of new standards know the capabilities of new IC process generations and therefore they design standards that often fully utilize it. As a result, transceivers become more complex and it is more difficult to make these designs error-free. A flexible radio could solve this problem by making designs patchable. In this way, consumer-owned products can be patched afterwards. This reduces costs as these radios can still be used. Moreover, new functionality can be added afterwards as well [11], [12].

The last trend that enables SDR, is the abundance of digital communication standards. Table 1.1 gives an overview of important wireless standards together with the used frequency bands and modulation type. It seems that each standard can be seen as a family of standards, an example being Global System for Mobile communication (GSM). Thus, the number of existing standards that manufacturers have to support is even larger than one would initially expect. However, there are also similarities among them: the used frequency bands are between 0.8 and 6 GHz with dominant frequency bands around the 0.8 GHz, 2 GHz, 2.4 GHz and 5 GHz. In addition, three families of transceivers can be distinguished, single-carrier, multi-carrier and spread-spectrum transceivers.

It is for these reasons, abundance of standards, more complex designs and the digitalization of the transceiver that SDR is an interesting concept.

1.3 SDR benefits

In this section several benefits of SDR are introduced which include:

- **Cost reduction**: Every new IC process generation has higher initial costs, so the minimal production volume to be cost-effective becomes higher and higher. If a manufacturer can make fewer chip designs by using for example SDR technology, manufacturing costs could be reduced due to higher volumes. Moreover, SDR could reduce the number of ICs in a radio which also reduces costs.

- **Patchable devices**: Transceiver IC designs are complex and new designs are even more complex due to the fact that they use new IC process generations. Each new technology enables more complex designs as it allows the use of more transistors. It is therefore difficult to make

Standard	Frequency band	Modulation type
CT2	864/868 MHz	GFSK
CT2+	944/948 MHz	GFSK
DECT	1880-1900 MHz	GFSK
PHS	1893-1920 MHz	DQPSK
IEEE 802.15.4	2402 - 2480 MHZ (N. America)	GFSK
	2412 - 2472 MHz (Europe)	
	2483 MHz (Japan)	
Bluetooth	2402 - 2480 MHz (N. America & Europe)	GFSK
	2447 - 2473 MHz (Spain)	
	2448 - 2482 MHz (France)	
	2473 - 2495 MHz (Japan)	
HomeRF	2402 - 2480 MHz (N. America & Europe)	GFSK
	2447 - 2473 MHz (Spain)	
	2448 - 2482 MHz (France)	
	2473 - 2495 MHz (Japan)	
IEEE 802.11a	5150 - 5250 MHz (USA)	OFDM: 2/4/16/64 QAM
	5250 - 5350 MHz (USA)	
	5725 - 5825 MHz (USA)	
IEEE 802.11b	2410 - 2462 MHz (N. America)	GFSK/DBPSK/DQPSK/QPSK
	2412 - 2472 MHz (Europe)	
	2483 MHz (Japan)	
HiperLAN/2	5150 - 5250 MHz (USA)	OFDM: 2/4/16/64 QAM
	5250 - 5350 MHz (USA)	
	5725 - 5825 MHz (USA)	
	5150 - 5350 MHz (Europe)	
	5470 - 5725 MHz (Europe)	
	5725 - 5875 MHz (Europe)	
	5150 - 5250 MHz (Japan)	
IS-54/IS-136	824 - 849 MHz	CDMA: $\pi/4$ DQPSK
	869 - 894 MHz	
	1850 - 1910 MHz	
	1930 - 1990 MHz	
IS-95	824 - 849 MHz	CDMA: QPSK/OQPSK
	869 - 894 MHz	
	1850 - 1910 MHz	
	1930 - 1990 MHz	
	1920 - 1980 MHz (Asia only)	
	2110 - 2170 MHz (Asia only)	
IMT-2000/UMTS	1920 - 1980 MHz	CDMA: QPSK
	2110 - 2170 MHz	
	1900 - 1920 MHz	
	2010 - 2025 MHz	
GSM	824 - 849 MHz	GMSK
	869 - 894 MHz	
	880 - 915 MHz	
	925 - 960 MHz	
	1710 - 1785 MHz	
	1805 - 1880 MHz	
	1850 - 1910 MHz	
	1930 - 1990 MHz	
PDC	810 - 826 MHz	$\pi/4$ DQPSK
	940 - 956 MHz	
	1429 - 1441 MHz	
	1453 - 1465 MHz	
	1477 - 1489 MHz	
	1501 - 1513 MHz	

Table 1.1: Overview of wireless standards
[13]

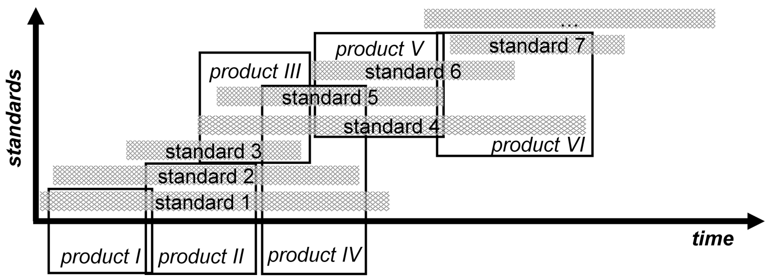


Figure 1.3: Standards support by products in time.

error-free designs and SDR could enable software-controlled designs which can be patched afterwards to correct possible design flaws.

- **Making devices adaptable**: Users want to send or receive information at minimal costs. So, a multi-standard radio could scan all frequency bands for available wireless infrastructure and use the one which meets the user demands at minimal costs. In addition, such a radio can adapt to the radio environment, thereby minimizing power.

- **Prolonging lifetime of a device**: Figure 1.3 illustrates the lifetime of products and wireless standards. One can see that current-day products support a fixed number of standards and in time new standards emerge and old ones disappear, making a product eventually obsolete. An SDR on the other hand will enable consumers to upgrade their radio with new functionality e.g. required by new standards, just by software updates, without the immediate need for new hardware[3].

However, the downside of SDR is an increased power consumption, because dedicated designs are more power efficient than flexible designs. Especially for mobile applications power consumption is very important as it determines battery life. Figure 1.4 illustrates this by showing that there is a huge gap between power consumption of dedicated designs and GPPs. This gap is increasing with every new IC process generation and nowadays dedicated designs can be a factor **1000** more power efficient. So there is a trade-off between power consumption and manufacturing costs: ASICs are very power efficient but fabrication of dedicated designs is time consuming and expensive due to small volumes whereas GPPs are very power inefficient but cheap to produce. An SDR design could have the advantages of both worlds by making an ASIC for multiple standards. Such a design will be far more power efficient that GPPs and will be cheaper to produce than an ASIC.

[3] In the long end, new hardware is still needed to support a new standard.

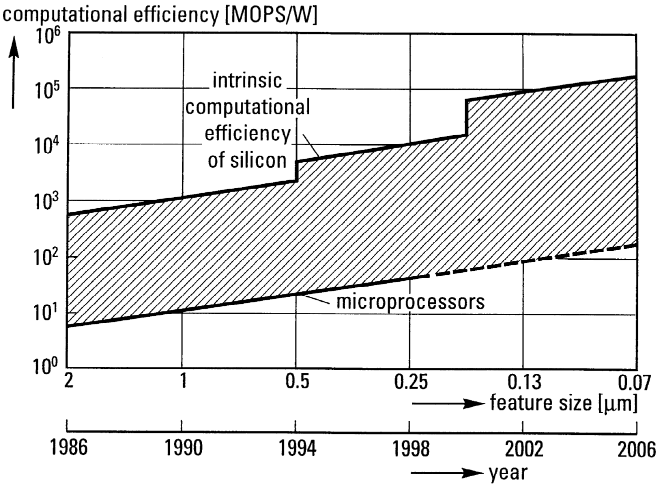


Figure 1.4: Computational efficiency, taken from [14].

1.4 SDR challenges

This section discusses technological issues which have to be solved in SDR terminals [15]:

- **Power consumption**: Compared with dedicated radio designs, SDR mobile terminals consume more power. For example, typical power consumption of a (dedicated) receiver front-end of a mobile phone is about 10 mW. SDR designs which are optimized for their task and smart power management can lower this gap.

- **Clock generation**: Every standard has its own clock rate. Using one reference oscillator per standard increases parts count, complexity and therefore costs. A single master clock saves costs but may lead to a very high clock rate which is power inefficient.

- **Receiver architecture**: As the receiver complexity is typically four or more times the transmitter complexity [15], it has a large impact on handset costs. The challenge is thus to develop a simple SDR receiver which has good performance.

- **Handset production**: The costs of handsets in volume productions is nearly a linear function of parts count. So, an SDR transceiver should use as few parts as possible. This involves also the need for efficient software because it lowers the memory needs and thus production costs of a handset. One way to cope with this challenge, which we also have used in our project, is to start with the transceiver of the most demanding standard and try to integrate transceivers of other standards.

1.5 TES.5177 project

The results presented in this thesis have been conducted within the TES.5177 project: *Development of a software-radio-based embedded mobile terminal* that is supported by the PROGram for Research on Embedded Systems & Software (PROGRESS) of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the technology foundation STW. The research is carried out by two chairs of the University of Twente: the IC-Design group which focuses on the Radio Frequency (RF) part and the Signals and Systems (SaS) group focussing on the digital baseband part. At the project's start we have defined the scope of project: the physical layer of the receiver for a mobile terminal. This section describes the project and introduces its goals.

Figure 1.5 depicts the Open System Interconnection protocol model (OSI). In current radio designs, the different radio functions are both mapped on software or hardware. The Physical layer (PHY) is generally implemented in hardware and higher layers are often software-based with the Logical Link Control (LLC) and Medium Access Control (MAC) layer as transition area. In the TES.5177 SDR project [16], we have designed an SDR for wireless LAN (and PAN) standards. Moreover, we have investigated to what extent the lowest layer, the physical layer of wireless standards can be implemented in software on a flexible platform and we have estimated the costs of such an implementation with respect to power consumption, computational power requirements and expected manufacturing costs. Already in 1999, Bose [3] showed for second-generation mobile standards that it is possible to perform most baseband functions of the physical layer on a GPP processor. This research tries to extend his work by investigating if this holds also for current-day wireless LAN standards. Although in the documentation of wireless standards the Forward Error Correction (FEC) coding and decoding is included in the physical layer, we have not implemented this functionality because in the definition of the OSI model, the physical layer does not include FEC coding/decoding.

Thus, we interpret SDR as an implementation technology, invisible for consumers. This line of thinking differs from the views of the SDRF: flexible, universal, radio systems with well-defined open interfaces at each layer of the OSI model where manufacturers, network operators and consumer can benefit (see Section 1.2). We doubt if network operators want to have *independent* consumers that can easily go to a competitor. The same applies for manufactures, because they also want to have in some way *dependent* oper-

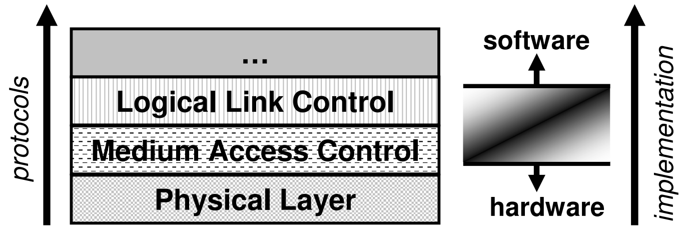


Figure 1.5: Mapping of protocols in current radio designs.

ators. On the other hand, using open interfaces in products can also be a selling point as operators do not want to rely on only one manufacturer.

1.5.1 Overview of wireless standards

In our SDR project we have decided not to focus on an all-standard radio but to start with an Software-Defined Radio (SDR) for wireless LAN (and PAN) standards. Table 1.1 gives an overview of important wireless standards together with the used frequency bands and modulation type. Wireless LAN standards use phase modulation or OFDM in the 2.4 GHz or 5 GHz frequency band, so we decided in the TES.5177 project to combine an instance of an phase-modulation standard (Bluetooth) with an OFDM standard (HiperLAN/2).

HIgh-PErformance Radio Local-Area Network/2 (HiperLAN/2) [17] is a high-speed Wireless LAN (WLAN) standard using OFDM. Its physical layer is very similar to the IEEE 802.11a standard (See also Chapter 3 for more information.). Bluetooth [18] on the other hand is a low cost, low speed PAN standard, designed for replacing fixed cables. Bluetooth uses Gaussian Frequency Shift Keying (GFSK) which is also used by other standards such as HomeRF and DECT (See Chapter 2 for more information.).

1.5.2 Demonstrator

A flexible, all standard radio will always consume more energy than a dedicated radio, thus the first application for a flexible radio will be an application where power consumption is less an issue; an example being a flexible radio in a notebook. Because one of the goals of our project is to build an SDR which is feasible within a few years (second goal on page 1, 2) we have decided to focus on a flexible radio in a notebook. This application for SDR has three advantages. First, we can use the processing capabilities of the general purpose processor for digital signal processing purposes (third goal). Second, in comparison to SDR for mobile phones, our application can consume

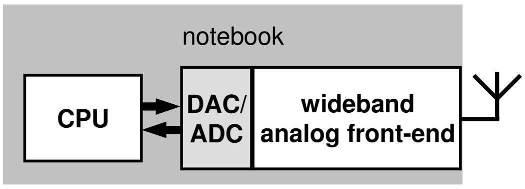


Figure 1.6: Project scope.

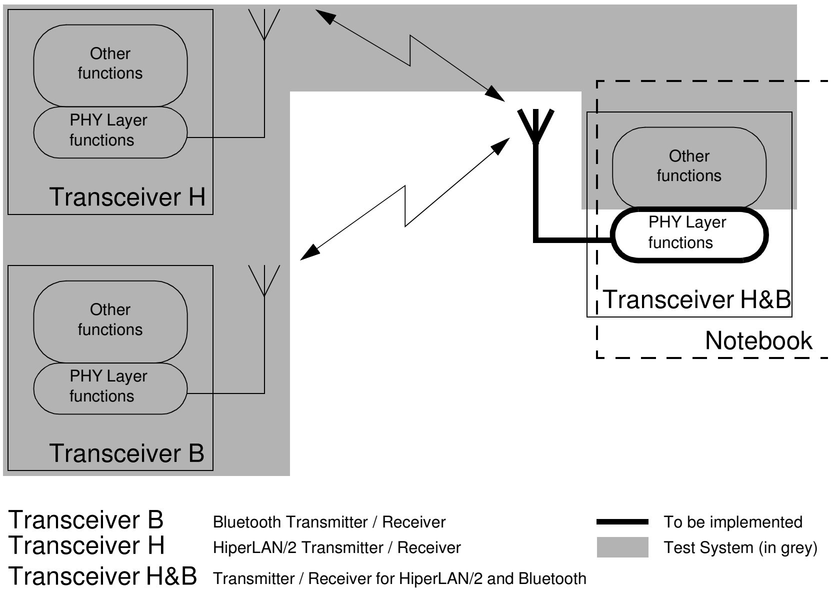


Figure 1.7: Rough testbed outline.

much more power (in the order of 1 W). Third, a notebook is very suited for demonstration purposes.

Figure 1.6 summarizes the design goal of our project, a notebook with a wideband RF front-end and a (partly) software implementation of the physical layer. Our intention is not to implement both a transmitter *and* receiver, but to focus on the receiver and to use Common Of The Shelf (COTS) components for building other components of our testbed. A rough sketch testbed is given in Figure 1.7. It is very difficult to integrate existing basestations into our testbed, because full control about the transmitted signals is often difficult and moreover these basestations require two-way communication. For these reasons, we have implemented the baseband parts of both transmitters as well.

1.5.3 Applications

SDR technology can be used both in basestations and mobile terminals. Our demonstrator is a good example of an application for the latter. The main disadvantages of a flexible SDR compared with dedicated designs are power consumption and higher manufacturing costs. These drawbacks are less important for basestations than for a mobile terminal because:

- Basestations have smaller productions volumes, so dedicated designs are more expensive compared with mobile terminals.
- Production costs are less important for basestations than for mobile terminals where every dollar counts.
- Basestations have to be backward compatible with all revisions of a standard
- and basestations always have to be upgraded to the newest revision of a standard.
- Power consumption is less important for basestations as they have line power.
- Multi-standard support is an important feature for basestations as it reduces costs significantly.
- Dedicated designs are designed for a worst-case scenario whereas SDR designs allow radio transmission and reception tailored to the radio channel.

Recently, a company in wireless infrastructure, Ericsson, announced that it shifts to a Digital Signal Processor (DSP)-only, software-defined approach to basestation design [19]. So, SDR has already been adopted by the industry for basestations and for that reason SDR in mobile terminals seems to be a more interesting research area.

1.5.4 Other SDR projects

Many other SDR projects are or have been conducted in the world (see for example [20]). It is for this reason cumbersome to generate a recent and complete list of all projects but at the start of the project we have performed a Google-based search (June 2001) on SDR projects [21] which resulted in 22 projects of which the research of 11 projects include the physical layer. Most projects focus on implementing a "flexible"ASIC for the digital part of 2^{nd} Generation (2G)/3^{rd} Generation (3G) mobile communication standards, an example being the SORT, SUNBEAM, SLATS, PASTORAL, GloMo projects and `Wireless3G4Free.com`; whereas the SODERA and PROMURA projects, research only the analog front-end. The remaining projects, the SpeakEasy and the FIRST project investigate both the analog and digital parts of the physical layer. The SpeakEasy project [20] is a military project of the USA and focuses for that reason on military communication standards. The FIRST project on the other hand has built a multi-mode terminal for 2G and 3G standards.

Given the project description above, the TES.5177 project is quite unique because we focus on integration of wireless LAN and PAN receivers of both the analog and digital front-end and investigate to what extent we can use a GPP processor for digital signal processing purposes.

1.6 SDR functional architecture

This section presents a functional architecture that brings the architectural descriptions of both standards to an equal level. This *SDR functional architecture* is used in this thesis for a number of narrow and wide scope objectives:

narrow scope objectives:

- *The definition of reference points*. The architecture should enable requirements definition and specification of systems parts.
- *The definition of interfaces*. As we do not plan to build all system parts ourselves, we need to be able to identify possibilities for usage of existing COTS components, possible re-use of existing software packages and potential alignment with SDRF architectures and implementations.
- *The delimitation of our demonstrator*. We need to be able to specify what is investigated and built and what the test environment is.

wide scope objectives:

- *The identification of inter-standard functional integration challenges*. An architectural model can be used as a tool for specifying a particular SDR *instance*, e.g. the Bluetooth instance. A functional model might also be useful in defining possible system parts for functional integration. Moreover, this may help to identify reconfigurability and reprogrammability aspects. Moreover, such a model might help in assessment of flexibility and re-use of system parts.
- *The identification of system aspects that need to be communicated to allow dynamic reconfiguration of a mobile*. If a mobile is to switch to another standard, it needs to be informed on what aspects to change. For this to happen, some sort of interaction-mechanism needs to be developed in which aspects that need to be communicated are identified.

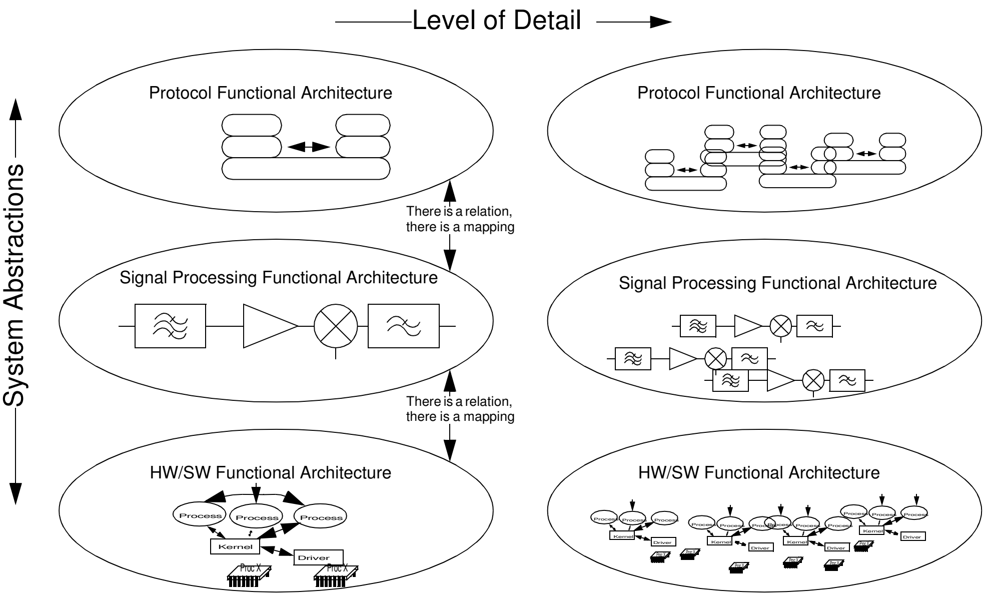

Level of Detail
System Abstractions
Protocol Functional Architecture
Protocol Functional Architecture
There is a relation, there is a mapping
Signal Processing Functional Architecture
Signal Processing Functional Architecture
There is a relation, there is a mapping
HW/SW Functional Architecture
HW/SW Functional Architecture
Process
Process
Process
Kernel
Driver
Proc X
Proc X

Figure 1.8: Architectures: Abstractions and Details.

Functional architectures can be described at different abstraction levels and different levels of detail, see for example Figure 1.8. In [22], we have investigated if the following candidate models satisfy the objectives above:

- the SDRF architectural model
- a layered protocol model
- a signal processing functional model
- a HardWare/SoftWare (HW/SW) functional model

The SDRF model Figure 1.1 is a model from an object-oriented software context in which functional blocks are shown together with the interfaces between them. The aim is to identify interfaces between software components that can be used in SDR systems. In our project we focus on the RF and Modulator/Demodulator (Modem) functions, however not as objects in object-oriented software, but as signal processing functions implemented in programmable hardware and software. The SDRF model is considered to be either too weak or too general for our specification purposes.

Moreover, the signal-processing functional model is, especially for the digital parts, too detailed for fulfilling the narrow scope objectives and also the HW/SW functional model is not the correct abstraction level to achieve the narrow scope objectives. Thus, we derived a layered protocol model at

Reference Point (RP)	Service Data Unit (SDU) stream
Antenna RP (ARP)	RF signal
Channel RP (CRP)	Channel signal
Demodulation RP (DRP)	Symbol stream
PHY RP (PRP)	PHY SDU stream
MAC RP (MRP)	MAC SDU stream

Table 1.2: Reference Points and Data Exchange

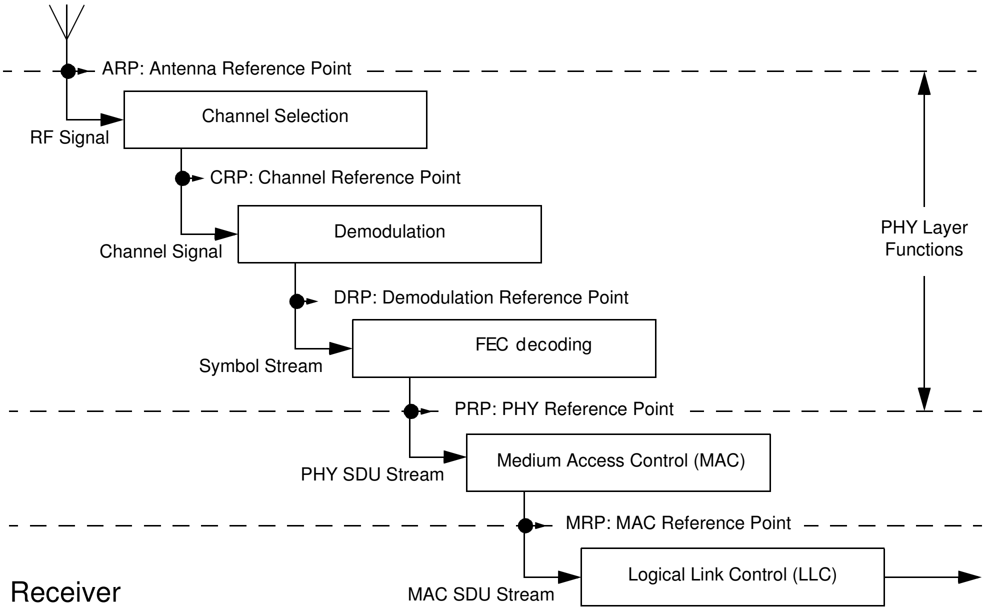


Figure 1.9: Layered protocol model of the receiver.

a suitable level of detail to achieve our goals [22]. This model is depicted in Figure 1.9[4]. This model uses Reference Points (RPs) which define interfaces between functional blocks and Service Data Units (SDUs) to describe the exchanged signals/data units between these blocks. All RFs and SDUs are listed in Table 1.2.

Using this model we can define the scope of the project: from Antenna RP to Demodulation RP.

[4] According to the OSI model, FEC decoding is not part of the physical layer. However, in the documentation of most standards, FEC decoding is included in this layer.

Chapter 2

Bluetooth

2.1 Introduction

Bluetooth is a low-cost, low-speed Personal Area Network (PAN) standard [18] , using GFSK modulation in the Industrial, Scientific and Medical (ISM) 2.4 GHz band. Moreover it uses Frequency Hopping (FH) for multiple access with 1600 hops/s. The channel spacing is 1 MHz and there are 79 available channels. Typical applications are replacements of cables, e.g. wireless headsets, keyboards, . . .).

First, this chapter specifies which part of the receiver has been built, using the layered protocol model of Section 1.6. Then, the functional architecture of both the transmitter and receiver are introduced (physical layer). Bluetooth is designed for low-cost and robust against interferers. For these reasons, it allows parameters in the radio system to have relatively large variations. At the receiver side, this allows the use of relatively simple demodulation algorithms as they are not affected by these parameter variations. Because there is a large gap between the Bit-Error Rate (BER) performance of these algorithms and the maximal achievable performance ($\approx$ 6 dB), we have also investigated if more advanced demodulation algorithms can be used. After the description of commonly used receiver architectures, a more advanced demodulation algorithm (Maximum A posteriori Probability (MAP) receiver) is presented. This receiver combines the demodulation and detection function.

2.1.1 Bluetooth layered protocol model

Figure 2.1 shows the Bluetooth protocol architecture. The Logical Link Control (LLC) layer of the OSI model is integrated in the *Logical Link Control and Adaptation Protocol (L2CAP)* whereas the MAC and PHY layer are provided by the *Baseband* function. In Section 1.6, a layered protocol model has been derived in order to define the scope of the project. This model has been ap-

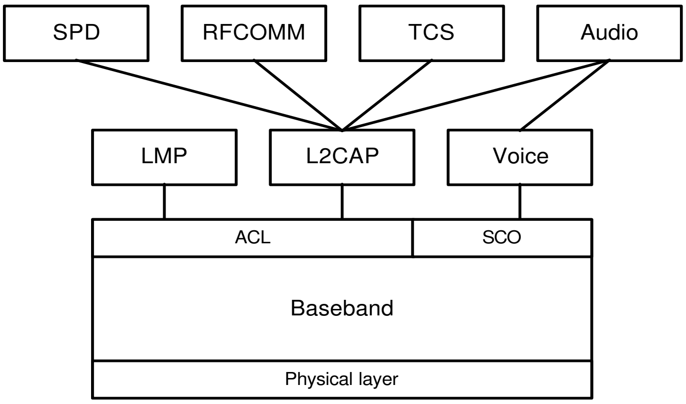


Figure 2.1: Bluetooth protocol architecture [18].

plied for the Bluetooth standard and is depicted in Figure 2.2. The scope of the project is from Antenna Reference Point (ARP) to Demodulation Reference Point (DRP). The Bluetooth receiver incorporates two main functions, channel selection and demodulation. Of course, the transmitter counterparts of these functions have to be implemented as well.

A HiperLAN/2 channel has a channel bandwidth of 20 MHz. Therefore, the output of the analog RF front-end (Section 4.3) should have a bandwidth of at least 20 MHz. In our project we have decided to use the same analog RF front-end, which implies that the output of the front-end contains 20 channels in Bluetooth mode. So, channel selection is performed both in the digital and analog domain, whereas demodulation is done completely digital. The FH algorithm transmits each packet at a different channel to cope with strong narrow-band interference and to enable multiple users. As we have chosen to perform channel selection both in the analog and digital domain, the reverse step, *de-hopping*, also has to be performed in both domains. The idea is that the analog front-end selects a 20 MHz wide band with a 10 MHz resolution [23].

The scope of the project lays in the PHY layer of wireless standards: from antenna output to raw bits. The next sections will describe in more detail the PHY layer functions of Bluetooth.

2.2 Transmitter

This section discusses the PHY layer of the Bluetooth transmitter and its functional architecture is shown in Figure 2.3. The architecture contains a *physical burst*, that creates packets from a bit stream. These packets contain besides the payload, a packet header and a device-specific access code. After packet generation, the packet will be modulated using GFSK modulation (*GFSK mod-*

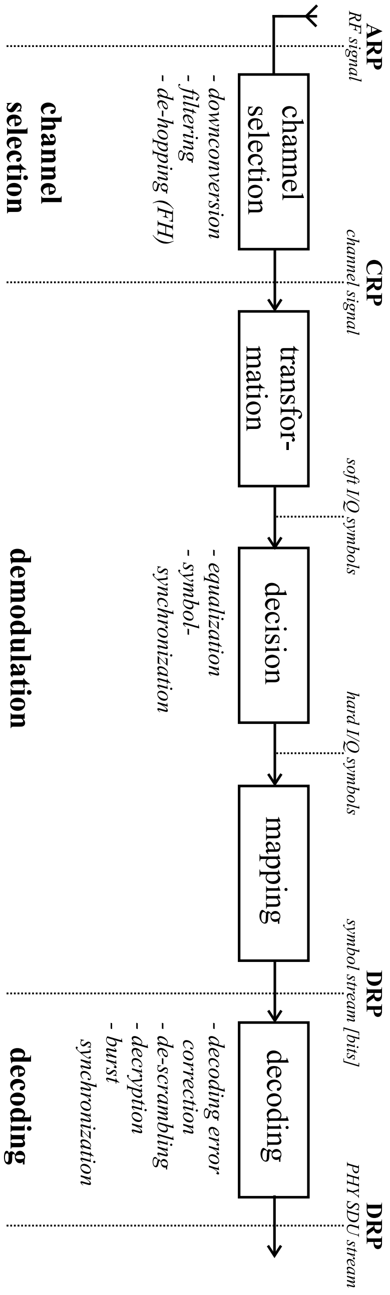


Figure 2.2: Bluetooth layered protocol model (receiver only).

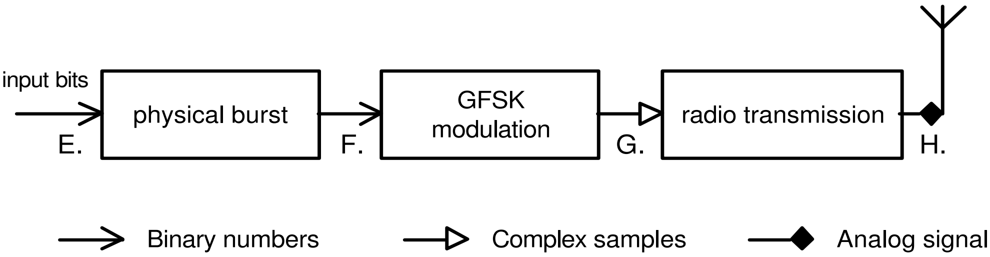


Figure 2.3: Bluetooth transmitter (physical layer).

ulation entity). The final step in the transmitter is to convert the baseband signal to RF frequencies (*radio transmission* entity). In most transmitters, the GFSK modulation and radio transmission function are not separate functions. However, in our implementation of the transmitter this is not the case, so the output of the GFSK modulation function is a complex baseband signal (with carrier frequency of 0 Hz). The next sections describe these parts in more detail. In order to test the SDR receiver functionality, we have implemented the transmitter from point E to H (the whole PHY layer).

2.2.1 Physical burst

The input of the physical burst is a (user) bit stream and this entity generates packets, also called MAC bursts, which contains besides the user bits special symbols such as the access code and header (Figure 2.4). The Bluetooth standard defines two types of communications: Synchronous Connection-Oriented (SCO) (e.g. audio applications) and Asynchronous Connection-Less (ACL) links (e.g. data applications). In addition, it defines 12 packets that can be used in both types of communication. Some of them use FEC (with a code rate R_c of 2/3), in the payload whereas other packets do not use FEC.

In our project we have not implemented the FEC coding/decoding functions, so we are interested in the raw BER only. Therefore, we have chosen to implement the *DH5* packet which uses no error correction in the payload[1]. Moreover, the DH5 packet is one of the longest packets in Bluetooth thereby having the most severe requirements. Figure 2.4 depicts the standard format of Bluetooth packets. The packet contains a device-specific access code, a header and a payload section. The latter is for the DH5 packet 339 bytes (2712 bits) long.

[1] The header section does use FEC coding, but we do not use this part.

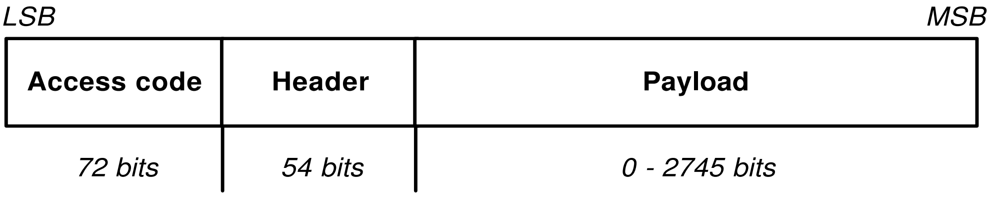


Figure 2.4: Bluetooth packet format.

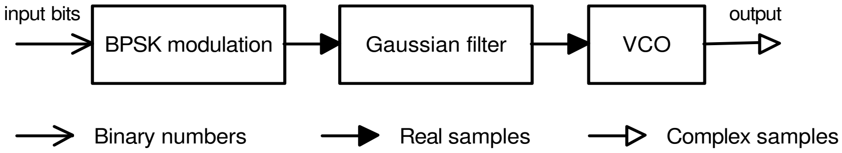


Figure 2.5: GFSK modulator.

2.2.2 GFSK modulation

The next step in the transmitter is to modulate packets using GFSK modulation. In normal continuous Frequency Shift Keying (FSK) a '0' is represented by an harmonic signal with frequency f_0 and a '1' by frequency f_1, both per interval of T seconds. Continuous FSK can be implemented by an Voltage-Controlled Oscillator (VCO) that is driven by the bit signal. In this implementation no phase shifts occur between bit transitions, which explains the name continuous FSK. However, due to the binary nature of the input signal, higher harmonics appear in the output signal and therefore results in a large bandwidth. It is for this reason that GFSK uses a Gaussian pre-modulation filter.

Figure 2.5 depicts the GFSK modulator. First the bits are Binary Phase-Shift Keying (BPSK) modulated: A '0' is being represented by a signal with value -1 and a '1' by a signal with value 1, each with a duration of T seconds. The symbol stream is low-pass filtered by a Gaussian filter. Its output is then connected to an VCO that translates the amplitude of the filtered bits into an frequency shift. In our implementation of the transmitter the output of the GFSK modulation function is a complex digital baseband signal (with carrier frequency of 0 Hz). In the next function, radio transmission, the signal is converted to the analog domain and up-converted to RF frequencies. In Figure 2.6, the effect of the Gaussian filter is shown: it reduces the bandwidth signal and therefore the output bandwidth of the VCO is also reduced. For this reason GFSK modulation is more spectrum efficient compared with FSK modulation at the cost of an increased BER [24].

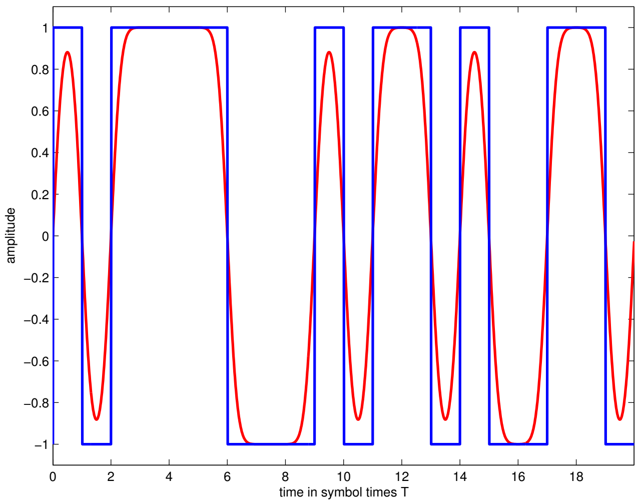


Figure 2.6: Time signal before and after the Gaussian filter.

2.2.3 Radio transmission

The last step in the transmitter is the radio transmission entity which converts the baseband signal to RF frequencies. It contains two functions: Digital-to-Analog (DA) conversion and up-conversion. The first function, DA conversion, converts the digital baseband signal to the analog domain. This analog signal is then up-converted to RF frequencies by multiplying the signal with a carrier frequency. This function also provides the FH functionality and has to comply to the power transmit requirements [18].

2.2.4 Power spectra

The standard allows the modulation index to vary between 0.28 and 0.35 [18]. The modulation index is defined as:

$$h = \frac{2f_d}{R} = 2f_dT \tag{2.1}$$

where f_d is the frequency deviation, R the bit-rate and T the symbol time [25]. The frequency deviation (f_d) is the maximum frequency shift with respect to the carrier frequency, if a ′0′ or ′1′ is being transmitted.

For Bluetooth signals f_d may vary between 140 and 175 kHz (Equation 2.1). Figure 2.7 shows the power spectrum of a Bluetooth signal at 2 MHz with minimal and maximal f_d. As expected, the power spectrum of the GFSK signal with maximal f_d is a little wider and more flat than the one with minimal f_d. Visual inspection of both figures shows that the signal strength has dropped about 40 dB for minimal f_d and about 25 dB for maximal f_d at the border of the channel (channel spacing is 1 MHz). Due to the relative small modulation index of Bluetooth GFSK, the signal energy is concentrated in a small band.

Figure 2.8 shows the power spectrum of two neighboring channel (one with center frequency 2 MHz and the other at 3 MHz) for $f_d = 175$ kHz. As expected, visual inspection of the curve shows that both channels are very well separated, although a lower f_d (see Figure 2.8(a)) results in lower co-channel interference.

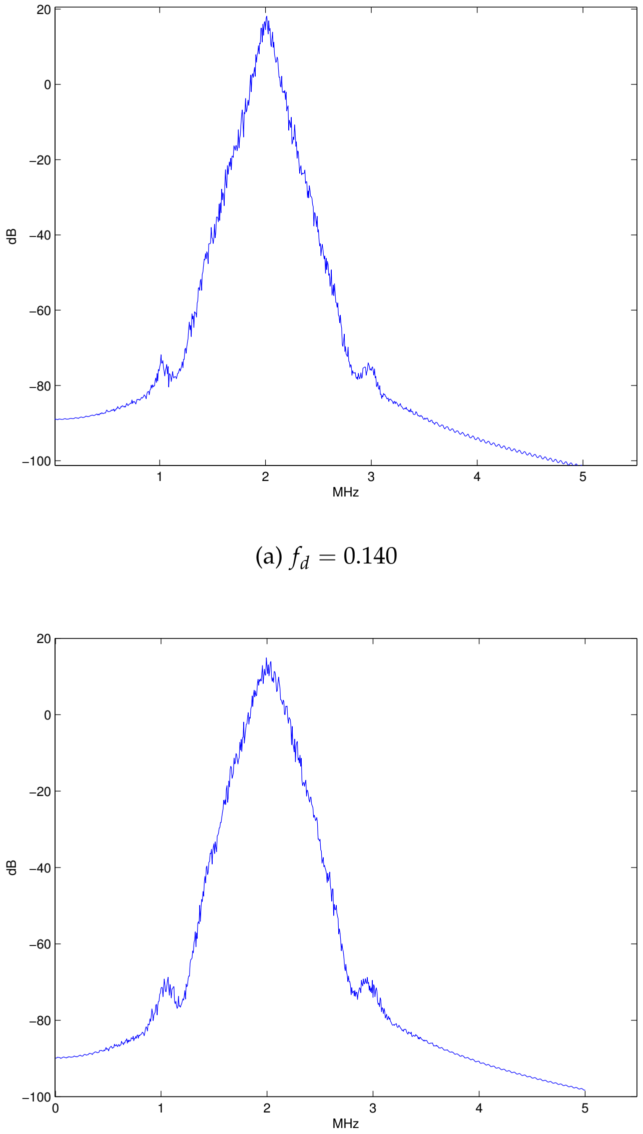
2.3 Receiver

Figure 2.9 shows the functional architecture of the Bluetooth receiver. The first step is to select the wanted Bluetooth channel and suppressing all others which is performed partially by the analog and partially by the digital front-end. This is achieved by mixing the wanted channel to baseband ($f_c = 0$) and applying a low-pass filter. The next step is to demodulate the Frequency Modulation (FM) signal into an Amplitude Modulation (AM) signal by taking the derivative of the phase. The synchronization/parameter estimation entity uses this signal to detect the start of a MAC burst (time/symbol synchronization) and estimates the frequency offset. A frequency offset introduces a Direct Current (DC) value in the AM signal and therefore it has to be corrected before bit decision. The next sections describe the functionality of each block in more detail.

2.3.1 Channel selection

Channel selection incorporates two functions: selecting the wanted channel and suppression of unwanted channels. In our testbed, we use a demodulator that requires a complex input signal at zero Intermediate Frequency (IF). So, the wanted channel is selected by mixing it to baseband. The Bluetooth receiver has knowledge about the hop pattern, so it knows at which frequency the next packet is transmitted. Therefore, it can mix the wanted channel to baseband by multiplying the incoming signal with the negative carrier frequency.

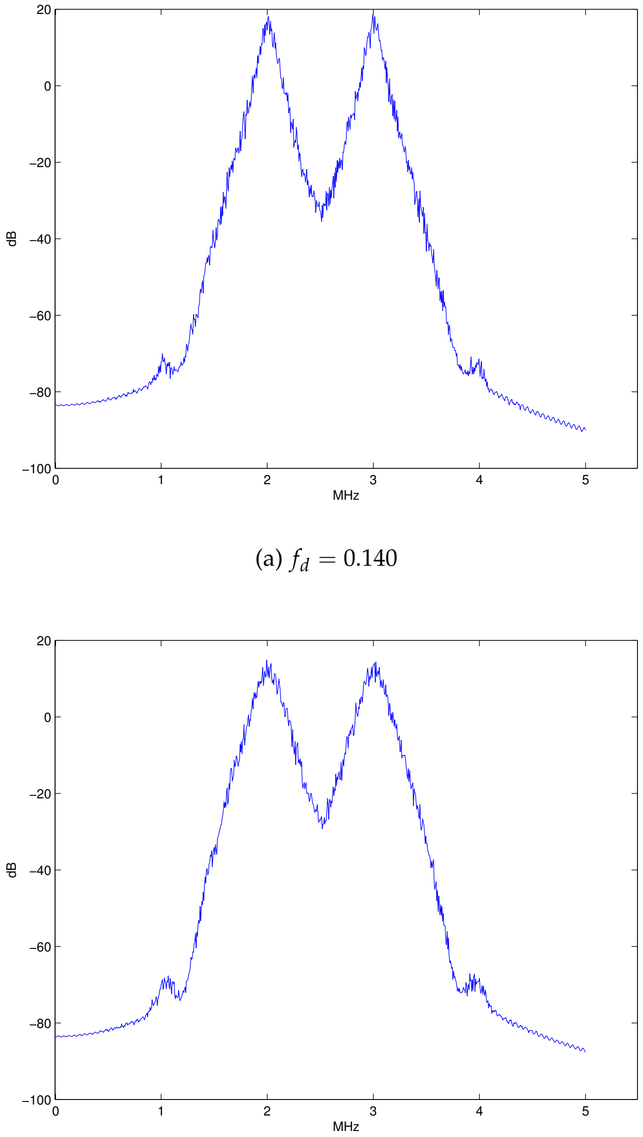
The other step, suppressing of unwanted channels, is mainly dictated by the Bluetooth standard: The BER has to be lower than 0.1% for a number of test conditions. In these experiments, the signal offered to the receiver consists of a wanted in-band signal (representing the information for a particular channel) and unwanted interfering signals (either in-band or out-band). So, for example, in the adjacent channel rejection experiment there is only one adjacent interferer and its power is increased until the BER exceeds 0.1%.



(a) $f_d = 0.140$

(b) $f_d = 0.175$

Figure 2.7: Power spectra of GFSK signals with minimal and maximal allowed frequency deviation (f_d).



(a) $f_d = 0.140$

(b) $f_d = 0.175$

Figure 2.8: Power spectrum of a two neighboring GFSK channels with minimal and maximal allowed frequency deviation (f_d).

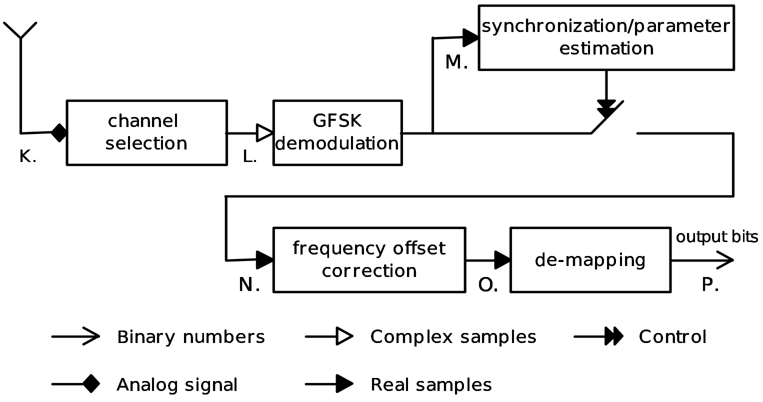


Figure 2.9: Bluetooth receiver (physical layer).

Requirement	Ratio
Co-Channel interference, $C/I_{co-channel}$	11 dB
Adjacent (1 MHz) interference, C/I_{1MHz}	0 dB
Adjacent (2 MHz) interference, C/I_{2MHz}	-30 dB
Adjacent ($\geq$ 3 MHz) interference, $C/I_{\geq 3MHz}$	-40 dB
Image frequency interference[2,3], C/I_{Image}	-9 dB
Adjacent (1MHz) interference to in-band image frequency, $C/I_{Image\pm 1MHz}$	-20 dB

Table 2.1: Interference performance [18].

The minimal requirements for each interferer is shown in Table 2.1 and Figure 2.10.

A rule of thumb is to design the channel selection filter in such a way that all interferers are suppressed 3 dB below the noise level required for a BER of 0.1%. As the demodulator does not specify an optimal pre-detection filter, simulations have been performed in order to determine other filter parameters, such as the passband width.

2.3.2 GFSK demodulation

After the wanted channel is selected, the GFSK signal has to be demodulated to bits. FM signals (e.g. GFSK signals) cannot be demodulated directly [26], but there exists several types of indirect methods [27]:

- FM-to-AM conversion
- Phase-shift discrimination
- Zero-crossing detection

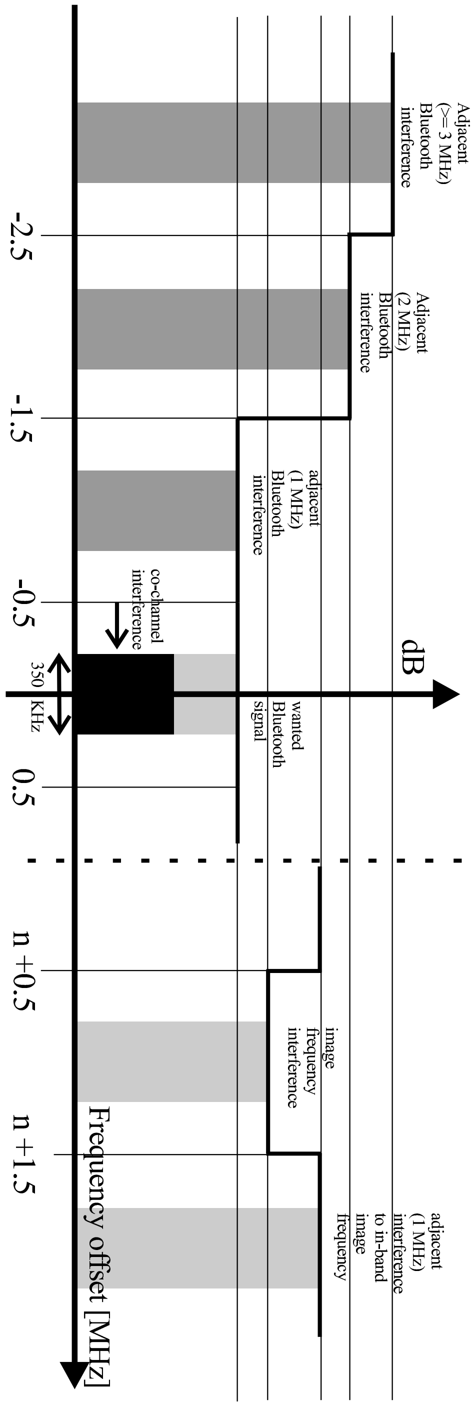


Figure 2.10: Bluetooth receiver interference definitions.

- Frequency feedback

For each method several implementation alternatives are possible. For example, the FM discriminator is an example of the FM-to-AM conversion method [26]. This method allows the implementation for low-cost radio units, which is essential for Bluetooth units [28]. However, we did not use this method in our project because it requires a real low-IF signal whereas the HiperLAN/2 receiver requires a complex zero-IF receiver. Efforts of using this demodulator in our testbed resulted in a computation-intensive channel selection part [29]. For these reasons, we used the Park's baseband FM demodulator [30] which is a phase-shift discrimination method. This demodulator requires a complex input signal at baseband and it is discussed below.

Park's baseband FM demodulator

In GFSK signals, the frequency property contains the data bits (Gaussian filtered). Thus, a receiver has to do the inverse of the transmitter: compute the phase of the incoming signal and take the derivative in order to retrieve frequency property i.e. the transmitted symbols:

$$\psi(t) = \frac{d\phi(t)}{dt} = \frac{d\tan^{-1}\left(\frac{Q(t)}{I(t)}\right)}{dt} = \frac{I(t)\frac{d}{dt}Q(t) - Q(t)\frac{d}{dt}I(t)}{I^2(t) + Q^2(t)} \qquad (2.2)$$

with $I(t)$ the in-phase input signal, $Q(t)$ the quadrature input signal and $\phi(t)$ the phase of the input signal.

Equation 2.2 only works if the input signal is at baseband ($f_c = 0$ MHz). The Park's baseband FM demodulator is a direct implementation of this equation (Figure 2.11).

2.3.3 Synchronization/parameter estimation

Data is transmitted in packets and the Bluetooth receiver has to detect the start of each packet (time synchronization). The receiver knows the first part of the packet, the access code (Figure 2.4) as the access code is some kind of destination address [18]. Therefore, the receiver has to demodulate only packets which start with the receivers access code[2].

Figure 2.4 shows the general packet format. Time and symbol synchronization is achieved by correlating the incoming signal with the known access code of 72 bits. If the correlation exceeds a threshold the receiver has detected the start of a packet. A single threshold value can be used, as the magnitude of the demodulator output signal (i.e. phase signal) is independent of the input signal power. The performance of this method is good enough, so more

[2] and some general broadcast access codes too.

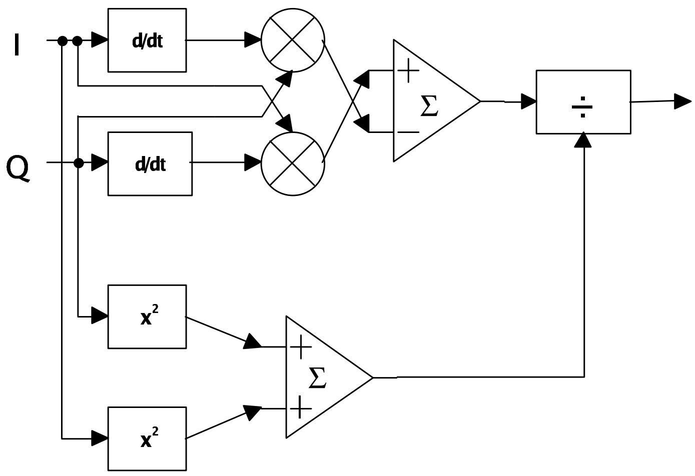


Figure 2.11: Park's demodulator.

advanced methods are not required for time synchronization. A receiver, which uses the FM-to-AM conversion method, is insensitive for a phase offset because it takes the derivative of the phase. Moreover, it also is insensitive for a change in modulation index. because a modulation index i.e. frequency deviation, changes the amplitude of the symbols. Therefore, a small modulation index will have a worse performance compared with a larger modulation index. However, a frequency offset will introduce a DC offset in the symbol signal. The standard allows for all packets an initial frequency accuracy of ±75 kHz of the carrier frequency, f_c and a frequency drift of ±40 kHz during the packet for DH5 packets.

For this reason, the maximal allowable frequency offset during the transmission of a packet is $75 + 40 = 115$ kHz and the minimal allowable frequency deviation f_d is 140 kHz[3]. These values are almost the same and therefore the receiver has to detect the frequency offset and correct the resulting DC offset. This offset can be calculated together with time synchronization, by determining the DC offset in the received access code.

2.3.4 De-mapping

The last step in the Bluetooth receiver is to convert the soft bits (BPSK modulated) to bits. For this purpose, a simple comparator is sufficient.

[3] The Bluetooth standard requires that the minimal frequency deviation should always be larger than 115 kHz.

2.4 MAP receiver

The Park's baseband FM demodulator is used in our project but it has two disadvantages: apparently a large difference with the theoretical performance[4] (Chapter 6.1.1) and it does not specify a pre-detection filter for optimal decision. The latter results in a channel selection filter based on trial and error. For these reasons, we investigated the use of a (simplified) Maximum A posteriori Probability (MAP) receiver. This receiver requires an orthogonal vector space which is given by the *Laurent decomposition* [31]. This Laurent decomposition describes the GFSK signal by a sum of linear, orthogonal, Pulse Amplitude-Modulated (PAM) waveforms. If the receiver has perfect time synchronization and knowledge of other parameters, such as phase offset, frequency offset and modulation index, the performance approximates the theoretical BER. The next sections introduce the Laurent decomposition and derive a MAP receiver for Bluetooth GFSK signals.

2.4.1 CPM signals

Bluetooth uses GFSK which belongs to the class of Continuous-Phase Modulation (CPM) signals. A complex envelope of a Continuous-Phase Modulation (CPM) signal can be written as follows [32]:

$$\tilde{s}(t, \boldsymbol{\alpha}) = e^{j\phi(t,\boldsymbol{\alpha})} \tag{2.3}$$

with

$$\phi(t, \boldsymbol{\alpha}) = h\pi \sum_n \alpha_n q(t - nT) \tag{2.4}$$

In the equation above, h is the modulation index; $\boldsymbol{\alpha}$, the symbol sequence belonging to the transmitted binary symbols ($\alpha_n = \{-1, 1\}$) and $q(t)$ is denoted as the phase response. For frequency modulation the relation between the phase and frequency response $g(t)$ is given by:

$$q(t) = \int_{-\infty}^{t} g(\tau) d\tau \tag{2.5}$$

The phase response $q(t)$ has the following properties:

$$q(t) = 0 \qquad t \leq 0 \tag{2.6}$$

$$q(t) = 1 \qquad t \geq LT \tag{2.7}$$

with T the symbol time or bit duration and L an integer value, which indicates the duration of the phase transition in symbol times.

[4] The Gaussian filter introduces a memory effect in the signal, which single-bit detectors do not utilize.

2.4.2 Laurent decomposition

In [31] it has been shown that Equation 2.3 can be written as a sum of PAM waveforms (this is also deduced in the appendix of [33]):

$$\tilde{s}(t, \boldsymbol{\alpha}) = \sum_{k=0}^{Q-1} \sum_{n} b_{k,n} c_k(t - nT) \tag{2.8}$$

with $Q = 2^{L-1}$

The *pseudo symbols* $b_{k,n}$ are given in Equation 2.9 and the PAM waveform $c_k(t)$ in Equation 2.11.

$$b_{k,n} = exp\{\jmath h\pi[(\sum_{m=-\infty}^{n} \alpha_m) - (\sum_{i=0}^{L-1} \alpha_{n-i}\beta_{k,i})]\} \tag{2.9}$$

with α_m the m^{th} data bit and $\beta_{k,i}$ is the i^{th} bit of the radix-2 representation of k , so $\beta_{k,i}$ has a value 0 or 1:

$$k = \sum_{i=1}^{L-1} 2^{i-1}\beta_{k,i} \qquad 1 \leq k \leq Q-1 \tag{2.10}$$

$c_k(t)$ is a product of *u(t)*:

$$c_k(t) = u(t) \prod_{i=1}^{L-1} u(t + iT + LT\beta_{k,i}) \qquad 1 \leq k \leq Q-1 \tag{2.11}$$

where the function *u(t)* is defined as follows:

$$\begin{array}{ll} u(t) = sin(h\pi q(t))/sin(h\pi) & 0 \leq t \leq LT \\ u(t) = u(2LT - t) & LT \leq t \leq 2LT \\ u(t) = 0 & elsewhere \end{array} \tag{2.12}$$

From Equation 2.12 it can be seen that the function *u(t)* is symmetric around $t = LT$ and has a duration of $2LT$. In many cases the signal power is concentrated in the first pulse, c_0. So, the CPM signal can be approximated by using only this pulse (which simplifies the construction of the MAP receiver):

$$\hat{\tilde{s}}(t, \boldsymbol{\alpha}) \approx \sum_{n} b_{0,n} c_0(t - nT) \tag{2.13}$$

2.4.3 Laurent decomposition of Bluetooth GFSK signals

This section derives the Laurent decomposition for Bluetooth GFSK signals. The frequency response function $g(t)$ is equal to the response of "one bit" to a Gaussian filter:

$$g(t) = \frac{1}{2}\left(erf\left(\chi(\frac{t}{T} + \frac{1}{2})\right) - erf\left(\chi(\frac{t}{T} - \frac{1}{2})\right)\right) \tag{2.14}$$

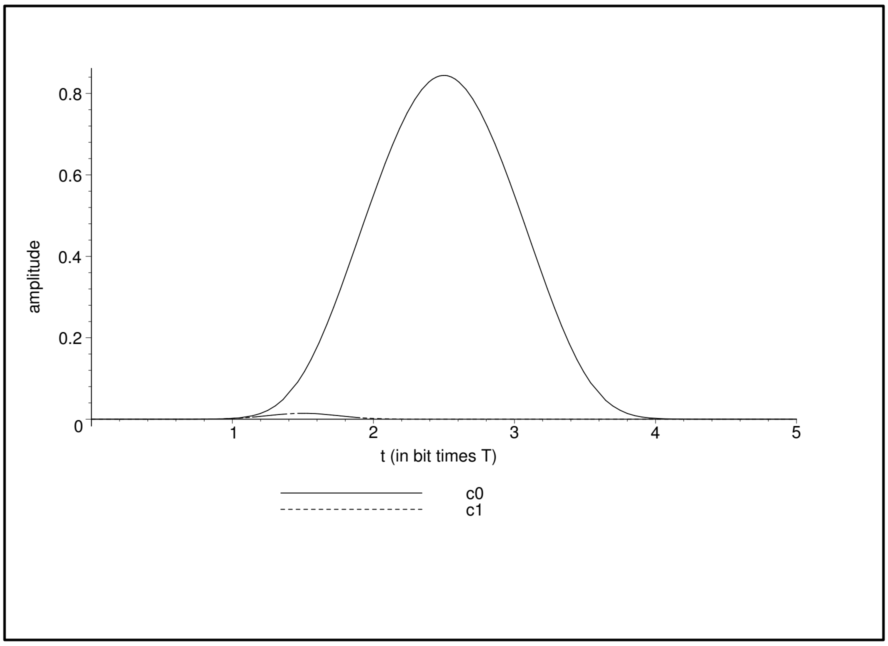


Figure 2.12: Waveform c_0 and c_1 of Bluetooth GFSK with $BT = 0.5$ and $L = 4$.

with:

$$\chi = \pi BT \frac{\sqrt{2}}{ln(2)} \tag{2.15}$$

and Bandwidth Time product (BT) is 0.5.

As the smallest modulation index gives the worst performance (at the same noise power), we have used this modulation index ($h = 0.28$) for the Laurent decomposition. The resulting PAM waveforms (with $L = 4$) are shown in Figure 2.12, Figure 2.13 and Figure 2.14 for a modulation index h of 0.28. From the figure, it can be seen that the first waveform c_0 is the most important pulse. PAM waveforms for larger modulation indices are very similar to ones depicted in Figure 2.12, Figure 2.13 and Figure 2.14.

Figure 2.15 shows the phase of an example GFSK signal and its Laurent approximation. Note that the approximation, using only the first Laurent term, and neglecting transients effects, already equals the original GFSK signal. Zooming in (Figure 2.16) reveals that there are some small differences.

2.5 A Maximum A posteriori Probability (MAP) receiver

In the previous section an orthogonal vector space has been derived that can be used in a MAP receiver. The first Laurent waveform contains more than 98 % of the signal energy, so other waveforms can be neglected, especially if

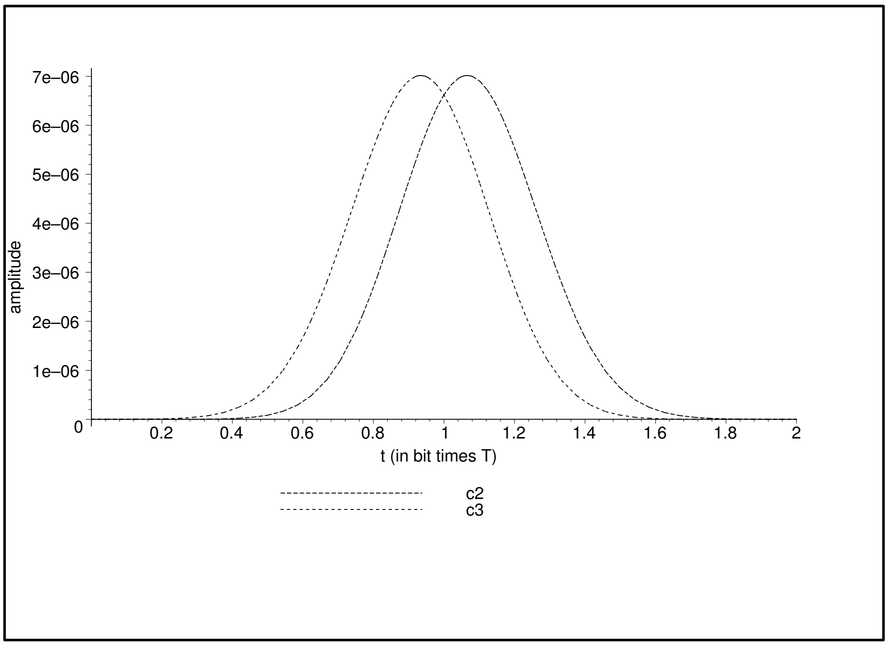


Figure 2.13: Waveform c_2 and c_3 of Bluetooth GFSK with $BT = 0.5$ and $L = 4$.

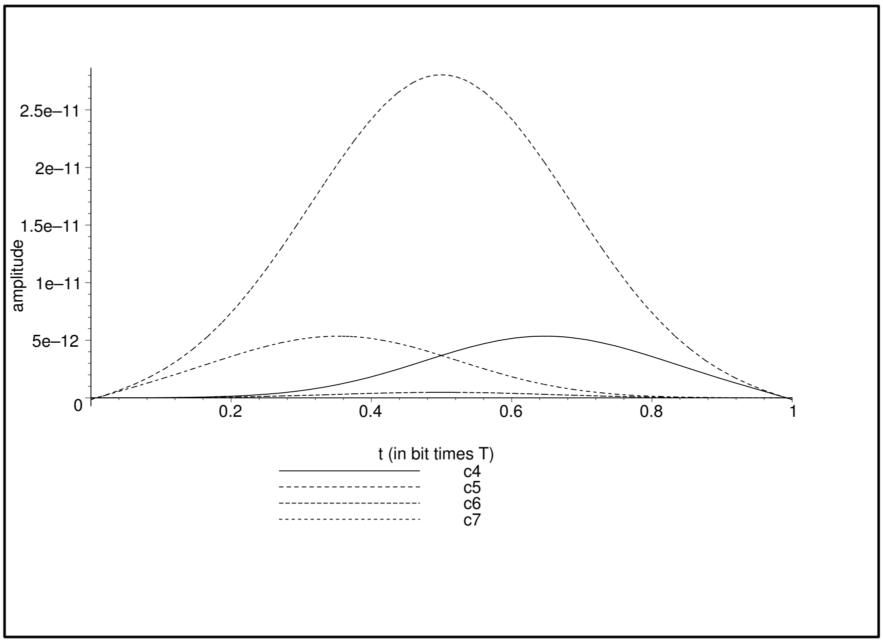


Figure 2.14: Waveform c_4, c_5, c_6 and c_7 of Bluetooth GFSK with $BT = 0.5$ and $L = 4$.

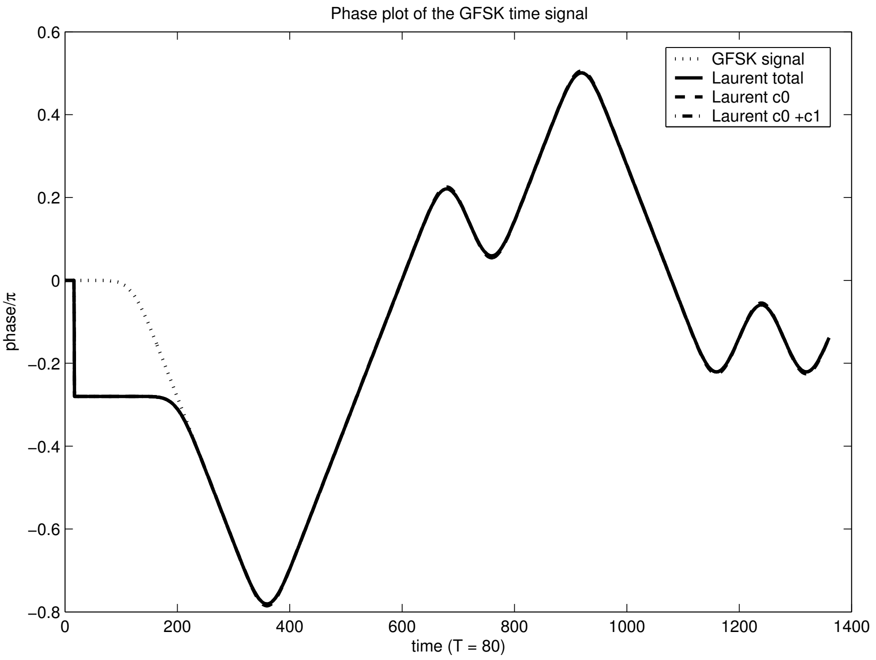


Figure 2.15: Phase plot of an example GFSK signal and its Laurent decomposition.

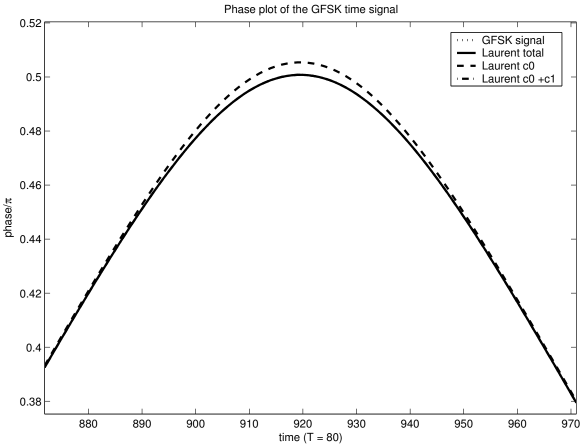


Figure 2.16: A zoomed-in version of figure 2.15.

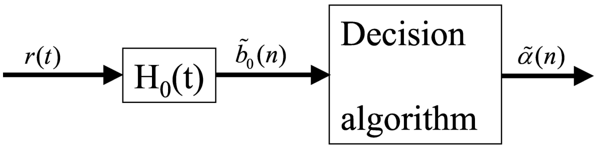


Figure 2.17: MAP receiver.

the noise power is high. Therefore we have to derive a matched filter for the first Laurent waveform:

$$H_0(t) = \frac{c_0(-t)}{||c_0(t)||} \tag{2.16}$$

The MAP receiver is shown in Figure 2.17. Recall that the input signal of the receiver, $r(t)$ can be approximated with the Laurent approximation (Equation 2.13):

$$r(t) \approx \sum_n b_{0,n} c_0(t - nT) \tag{2.17}$$

So, the filter $H_0(t)$ is a matched filter for the first Laurent waveform. Therefore the output of the filter is an (optimal) estimation of $b_{0,n}$. This estimation has an optimal $\frac{E_b}{N_0}$ but suffer also from Inter-Symbol Interference (ISI). Thus, an efficient search algorithm is needed which determines the *optimal* path through the trellis diagram. For our MAP receiver we used the *Viterbi* algorithm.

As the Gaussian filter has an filter length of about 3 bit times (see Figure 2.6), a Viterbi algorithm with maximal 2 state variables is sufficient: (α_{n-2} and α_{n-1}). The states and their branches to the next states are shown in Figure 2.18.

Steps in the Viterbi algorithm

Every sample the *Viterbi* algorithm must:

- calculate all 8 branch metrics (see Equation 2.18).
- two paths enter each state and only the path with the highest *state* i.e. branch metric, has to saved, the other can be discarded.
- determine the state with the highest value and then decide the value of α_{n-2}, and update the state variables α_{n-2}, α_{n-1} and b_{n-3}.

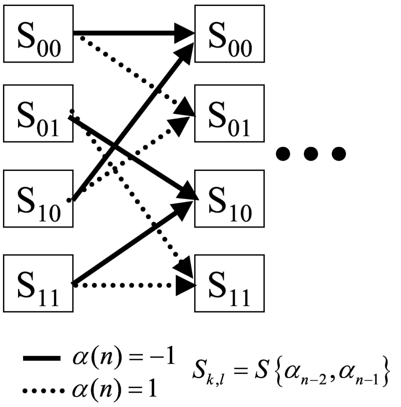


Figure 2.18: Trellis diagram in the MAP receiver.

The Viterbi algorithm can be initialized by setting the first sample to b_{n-3} and setting all states to zero. Then, the algorithm starts at the 4^{th} sample for decoding the first bit.

The Branch Metric (BM) is defined as follows:

$$BM = b_{0,n-3} \cdot e^{jh\pi(\alpha_{n-2}+\alpha_{n-1}+\alpha_n)} \cdot \tilde{b}^*_{0,n} \tag{2.18}$$

where $\tilde{b}_{0,n}$ is the output of filter $H_0(t)$ for the n^{th} bit. In fact, the first part of this equation calculates the output vector for each combination of α_{n-2}, α_{n-1} and α_n. This output vector is then compared with the measured output value $\tilde{b}_{0,n}$. The most likely combination determines the value of α_{n-2}.

2.5.1 Synchronization/parameter estimation

The MAP receiver has a good performance but it requires on the other hand a very precise knowledge of signal properties such as phase offset, frequency offset and modulation index. This precise knowledge is required because these effects determine the trellis diagram of the received signal. Small differences between the trellis diagram of the received signal and the diagram used in the receiver will result in bit errors.

Time synchronization and frequency offset estimation for the MAP can be achieved by using an extra Park's receiver and use the output of this receiver for time synchronization and estimation of the frequency offset. In addition the phase offset can be neglected if the first input sample is used as start (origin) in the Viterbi algorithm. The last parameter, the modulation index, has to be estimated too, because it determines the states in the trellis diagram. More information about modulation index estimation can be found in [34].

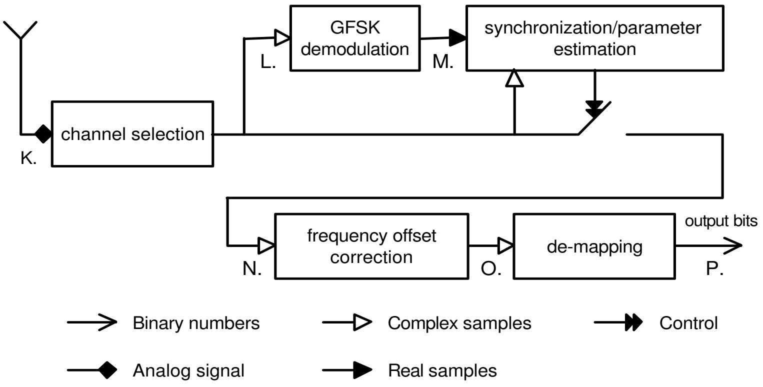


Figure 2.19: Bluetooth MAP receiver (physical layer).

2.5.2 MAP Receiver

The structure of the physical layer using the MAP receiver is depicted in Figure 2.19. The first step is channel selection which mixes the wanted channel to baseband and filters the signal with the matched filter for the first Laurent waveform, $H_0(t)$. The output of the filter is an (optimal) estimation of $b_{0,n}$. For time synchronization and frequency-offset estimation the signal is demodulated using the Park's algorithm. Althoug, the Park's algorithm is not sufficient for demodulation of data symbols, it has sufficient performance for estimation of the timing and frequency offset. Because the Park's algorithm is only needed at the start of the packet, it does not increase the computational requirements of the MAP receiver much. For estimation of the modulation index the synchronization/parameter estimation entity needs the complex output of the matched filter [34]. Moreover this entity also determines the optimal sample moment and decimates the signal to one sample per symbol. The frequency offset is corrected before the de-mapping entity, where a Viterbi algorithm is used for bit decision.

Chapter 3

HiperLAN/2

3.1 Introduction

HiperLAN/2 [35] [17] is a high-speed Wireless LAN (WLAN) standard using Orthogonal Frequency Division Multiplexing (OFDM) modulation in the 5 GHz frequency band. It has been developed by the European Telecommunications Standard Institute (ETSI) and the physical layer is very similar to the American Institute of Electrical and Electronics Engineers (IEEE) 802.11a standard. Both standards use more or less the same physical layer, but differ in other layers [36]. In Europe, the frequency spectrum allocated for HiperLAN/2 (Table 1.1), is divided into 19 radio channel of 20 MHz bandwidth each and multiple access is achieved by using Time Division Multiple Access (TDMA). The HiperLAN/2 MAC frame has a fixed duration of 2 ms and is divided in six *logical* channels: a Broadcast CHannel (BCH), a Frame control CHannel (FCH), an Access Feedback Channel (AFC), a downlink phase, a uplink phase and a Random access CHannel (RCH) [36]. See Figure 3.1.

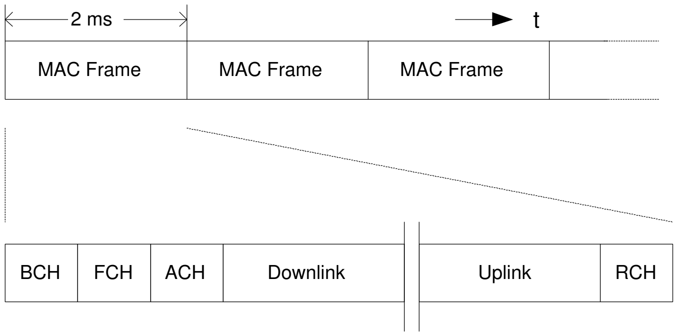


Figure 3.1: HiperLAN/2 MAC frame structure.

First, this chapter specifies which part of the receiver has been built, using the layered protocol model of Section 1.6. Then, the functional architecture of both the transmitter and receiver are introduced. The standard is designed for digital baseband processing and for this reason, the functional architecture is more or less fixed and differs only in the used algorithms.

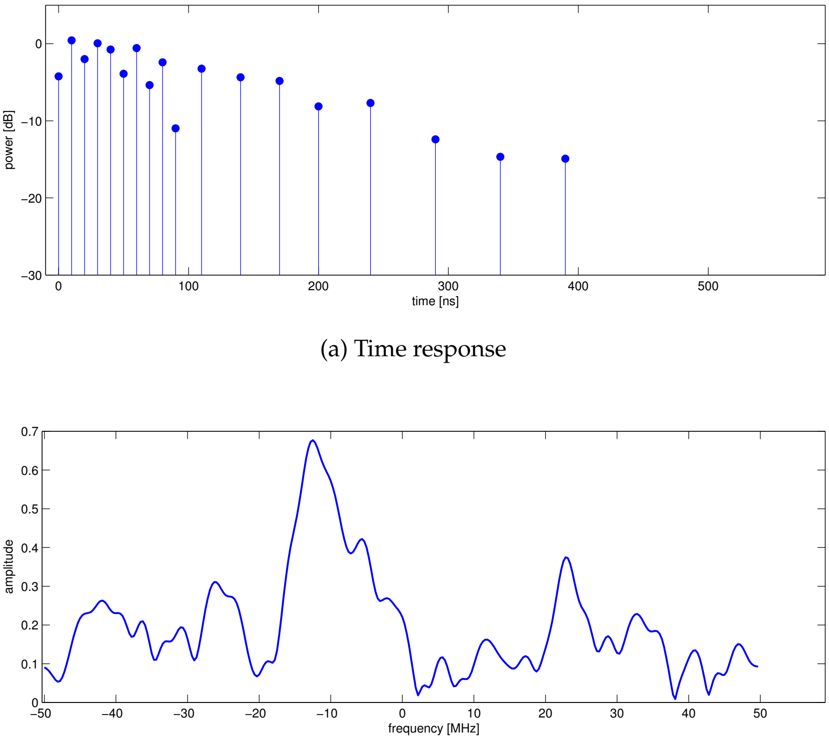
3.1.1 OFDM

Wireless radio channels are determined by three physical mechanisms; reflection, diffraction and scattering [37], [38]. As a result of these influences, a transmitted signal is received multiple times. The number of received signals and the time between the first and last signal i.e. delay spread is influenced by the environment. Typical values for the maximum delay spread varies from 30 ns for a single office room to 130 ns for a office building [38]. An example of a power delay profile for an wireless radio channel is depicted in Figure 3.2. This figure shows for each received signal, the average power and the time offset compared with the first signal.

If the symbol time is much larger than the delay spread, the Inter-Symbol Interference (ISI) is small and can be corrected in the receiver. However, current high-speed standards require symbol times which are smaller than the delay spread and ISI limits the transmission speed. A solution to this problem, is to use not one transmission carrier, but use multiple carriers. In this case, the symbol time of each subcarrier is much larger than the delay spread, resulting in small ISI, and the total bit-rate can be increased by using more carriers. An efficient multi-carrier system is OFDM. OFDM uses a minimal, orthogonal, distance between subcarriers which is accomplished by using an Inverse Fast Fourier Transform (IFFT) in the transmitter. At the receiver side, an Fast Fourier Transform (FFT) can be used for demodulation. By using the IFFT and FFT an OFDM radio is an efficient modulation technique for communication with respect to used spectral efficiency and computational power requirements.

3.1.2 HiperLAN/2 layered protocol model

Figure 3.3 shows the HiperLAN/2 protocol architecture. The standard only defines the lower layers of the OSI and can be used in combination with other communications standards. The second layer of the OSI model is the Data Link Control (DLC) layer which is also shown in Figure 3.3. It consists of two sublayers: the Logical Link Control (LLC) and Medium Access Control (MAC) layer. A layered protocol model (Section 1.6) for the PHY layer is depicted in Figure 3.4. The receiver has the same main functions as the Bluetooth receiver, channel selection and demodulation. However, channel selection is much easier as it does not require de-hopping functionality. In



(a) Time response

(b) Amplitude spectrum

Figure 3.2: Instance of a wireless channel model.

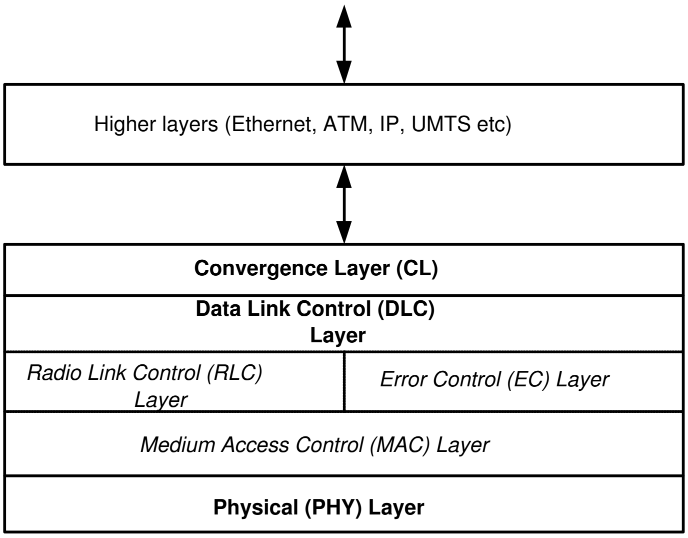

Higher layers (Ethernet, ATM, IP, UMTS etc)
Convergence Layer (CL)
Data Link Control (DLC) Layer
Radio Link Control (RLC) Layer
Error Control (EC) Layer
Medium Access Control (MAC) Layer
Physical (PHY) Layer

Figure 3.3: HiperLAN/2 protocol model (see [36]).

order to test the SDR receiver, the physical layer functions of the transmitter have to be implemented as well.

The scope of the project lays in the PHY layer of of HiperlAN/2 and the next sections describe this layer in more detail.

3.2 Transmitter

The PHY layer provides transportation mechanisms of bits between the DLC layer in transmitter and receiver. The standard defines seven functions in the transmitter:

- Scrambling of the binary input stream
- Forward Error Correction (FEC) coding
- Interleaving
- Mapping
- Modulation using OFDM
- Physical burst generation
- Transmitting of the burst

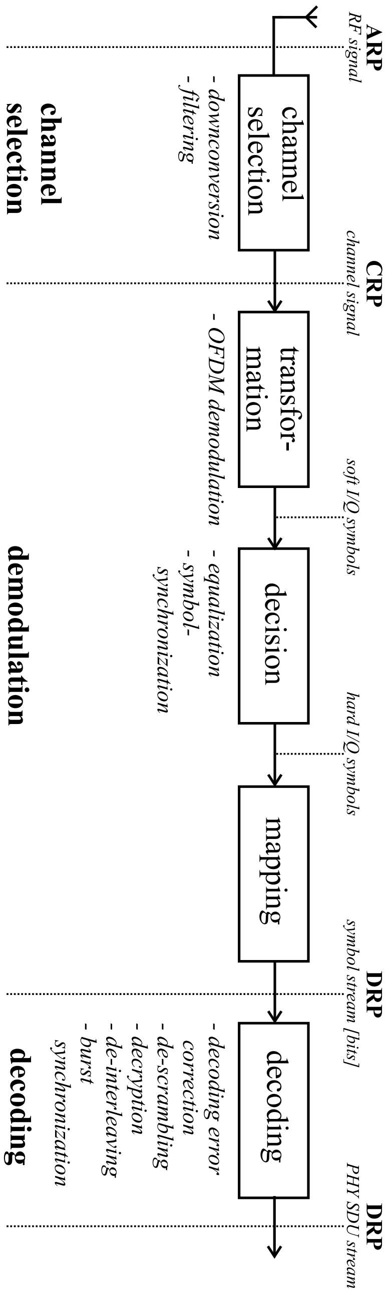


Figure 3.4: HiperLAN/2 layered protocol model (receiver only).

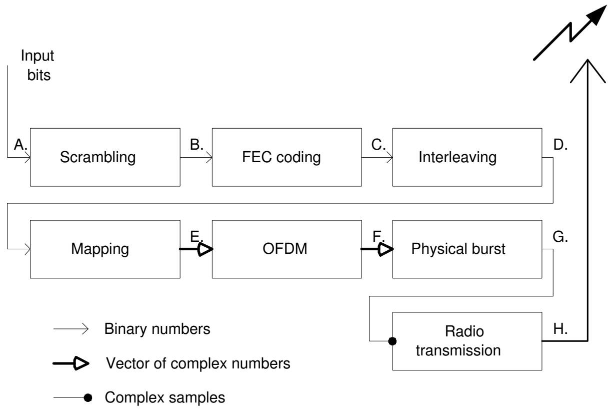


Figure 3.5: Data flow between the seven functions of the HiperLAN/2 PHY layer in the transmitter ([17]).

In Figure 3.5, the data flow between these functions is outlined. The PHY layer starts with point A, where the output bits of the DLC layer enter. First, the bits are scrambled, FEC encoded and interleaved. This results in a bit group at D called *raw* bits. The raw bits are converted to complex symbols by the mapping function and the OFDM function maps these symbols on subcarriers. Finally, a physical burst (i.e. MAC burst) is generated, by adding special symbols (*preambles*). All steps are performed in the digital domain, except for the last function, radio transmission, where the complex digital signal is converted to the analog domain and up-converted to radio frequencies (RF).

In order to test the receiver functionality, the transmitter has been implemented from point D to H. For verification with literature, the *FEC coding* and *Interleaving* block is included in the baseband simulation model by using the IT++ library [39].

3.2.1 Scrambling

The first function in the PHY layer is *Scrambling*, which function is to randomize the bit stream. In this way the number of consecutive ones or zeros is avoided which result in a possible lower BER. This function is not implemented in the baseband simulation model because the model uses already random user bits. More information about this function can be found in [17].

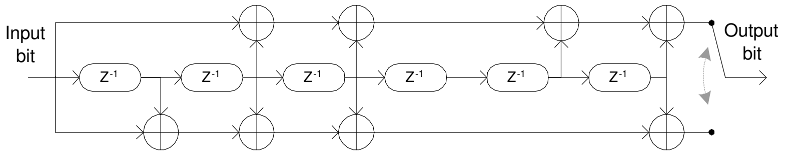


Figure 3.6: Bit-rate independent convolutional encoder [37] (modulo-2 arithmetic).

$R_c = \frac{1}{2}$

Input bit	d_0
Output bits	$\mathbf{b_0}$ $\mathbf{b_1}$

$R_c = \frac{9}{16}$

Input bit	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
Output bits	$\mathbf{b_0}$ $\mathbf{b_1}$	$\mathbf{b_2}$ $\mathbf{b_3}$	$\mathbf{b_4}$ $\mathbf{b_5}$	$\mathbf{b_6}$ $\mathbf{b_7}$	$\mathbf{b_8}$ b_9	$\mathbf{b_{10}}$ $\mathbf{b_{11}}$	$\mathbf{b_{12}}$ $\mathbf{b_{13}}$	$\mathbf{b_{14}}$ $\mathbf{b_{15}}$	b_{16} $\mathbf{b_{17}}$

$R_c = \frac{3}{4}$

Input bits	d_0	d_1	d_2
Output bits	$\mathbf{b_0}$ $\mathbf{b_1}$	$\mathbf{b_2}$ b_3	b_4 $\mathbf{b_5}$

Table 3.1: Bit-rate mode dependent puncturing. The **bold** values are placed in the output bit stream. The input bit stream of the FEC coding is denoted as $d_0, ...$ and the output of the convolutional encoder as $b_0, ...$ [37].

3.2.2 FEC coding

The next step is Forward Error Correction (FEC) which adds redundant bits in order to provide reliable transmission. The FEC of HiperLAN/2 contains two parts: a bit-rate *independent* and bit-rate *dependent* coding. The bit-rate independent coding part is a non-systematic convolutional encoder with constraint length seven [17], [40] and is shown in Figure 3.6. This encoder generates two bits per input bit, so the code rate $R_c = \frac{1}{2}$.

HiperLAN/2 supports three code rates, $R_c = \frac{1}{2}, \frac{3}{4}, \frac{9}{16}$, see also Table 3.2. The bit-rate dependent coding part converts the coded bit stream from the convolutional encoder to the three code rates by omitting certain bits i.e. puncturing of the code (Table 3.1). Puncturing increases the bit-rate of the user bits (less coding), but will decrease the BER performance of the system.

3.2.3 Interleaver

In HiperLAN/2, the used interleaver is a block interleaver with a block size corresponding to the number of bits in a single OFDM symbol, N_{CBPS}. The interleaver is defined by a two step permutation. The interleaver ensures that adjacent coded bits are mapped onto non-adjacent subcarriers and alternatively onto less and more significant bits of the used constellation.

The two *permutations* are defined as follows:

- Let k be the index of bits at the input, let i be the bit index after this permutation. The first permutation is given by (see [17]):

$$i = \frac{N_{CBPS}}{16}(k \mod 16) + \lfloor \tfrac{k}{16} \rfloor \tag{3.1}$$

  with $k = 0, 1, \ldots, N_{CBPS} - 1$ and $\lfloor \cdot \rfloor$ the floor operation.

- Let i be the bit index after the first permutation and let j be the index after this permutation. The second permutation is given by:

$$j = s \, \lfloor \tfrac{i}{s} \rfloor + \left(i + N_{CBPS} - \lfloor \tfrac{16i}{N_{CBPS}} \rfloor \mod s \right) \tag{3.2}$$

  with $i = 0, 1, \ldots, N_{CBPS} - 1$, $s = \max(\frac{N_{BPSC}}{2}, 1)$ and N_{BPSC} the number of bits mapped per OFDM subcarrier.

3.2.4 Mapping

This function maps groups of bits on complex symbols. The HiperLAN/2 standards supports seven bit-rate modes. This is accomplished by using different code rate and different subcarrier modulation types, see Table 3.2. So, this function maps bits on BPSK, QPSK, 16-QAM or 64-QAM Gray-coded constellations, depending on the used bit-rate mode. In addition, an normalization factor, K_{mod} is applied to achieve the same average power for all modulation types (Table 3.3).

3.2.5 OFDM

This block constructs an OFDM symbol by:

- Generating pilot carriers
- Constructing of an OFDM symbol (frequency domain)
- Conversion of the symbol to the time domain
- Adding the cyclic prefix

Mode	Data bit-rate [Mbit/s]	Subcarrier modulation	Code rate R_c	Coded bits per sub-carrier N_{BPSC}	Coded bits per OFDM symbol symbol N_{CBPS}	Data bits per OFDM symbol N_{DPBS}
A	6	BPSK	1/2	1	48	24
B	9	BPSK	3/4	1	48	36
C	12	QPSK	1/2	2	96	48
D	18	QPSK	3/4	2	96	72
E	27	16QAM	9/16	4	192	108
F	36	16QAM	3/4	4	192	144
G	54	64QAM	3/4	6	288	216

Table 3.2: Bit-rate modes of supported by the HiperLAN/2 standard.

Generating pilot carriers

Four subcarriers of the OFDM symbol are used to transmit a known sequence. These carriers are called *pilot carriers* and can be used in the receiver for synchronization. The known sequence can be generated with the same polynomial used in the scrambling function (with an initial state of all ones i.e. 1111111):

$$X_7 \oplus X_4 \oplus 1 \tag{3.3}$$

Where $X_{1..7}$ represents the state of the scrambler. Note that "$\oplus$" is the exclusiveve OR operator.

The value p_n $(\in -1, 1)$ is the n^{th} output of the scrambling function.

Constructing of an OFDM symbol (frequency domain)

Each OFDM symbol consists of 48 data carriers and 4 pilot carriers and therefore the stream of complex symbols is divided into groups of $N_{SD} = 48$ complex numbers first. Each group of data carriers is indicated as $D_{m,n}$ (with m denoting the subcarrier number and n denoting the OFDM number). The carriers of the OFDM symbol, $C_{l,n}$ are constructed as follows:

$$C_{l,n} = \begin{cases} D_{l+26,n} & -26 \leq l \leq -22 \\ +p_n & l = -21 \\ D_{l+25,n} & -20 \leq l \leq -8 \\ +p_n & l = -7 \\ D_{l+24,n} & -6 \leq l \leq -1 \\ 0 & l = -7 \\ D_{l+23,n} & 1 \leq l \leq 6 \\ +p_n & l = 7 \\ D_{l+22,n} & 8 \leq l \leq 20 \\ -p_n & l = 21 \\ D_{l+21,n} & 22 \leq l \leq 26 \end{cases} \tag{3.4}$$

with l denoting the subcarrier number and n denoting the OFDM number. The system does not output DC component as $C_{0,n} = 0$.

$C_{l,n}$ represents the OFDM symbol in the frequency domain and can be transformed to the time domain by using the Fourier transform:

$$\widetilde{s_n}(t) = \begin{cases} \sum_{l=-\frac{N_{ST}}{2}}^{\frac{N_{ST}}{2}} C_{l,n} e^{j2\pi l \Delta_f (t - T_{CP} - nT_S)} & , nT_S \le t < (n+1)T_S \\ 0 & , \text{else} \end{cases} \tag{3.5}$$

The meaning of the symbols and their values are explained in Table 3.4.

Conversion of the symbol to the time domain

The ETSI proposes a sample frequency $f_{sample} = 20$ MHz. In that case one OFDM symbol has a duration T_S of 80 samples with a useful data part T_U of 64 samples.

Let us assume that we calculate OFDM symbol $n = 0$ and $0 \le t < T_U$, then the sampled version of equation 3.5 results in:

$$\widetilde{s_n}[m] \triangleq \widetilde{s_n}(t)\Big|_{t=\frac{m}{f_{sample}}} = \sum_{l=-\frac{N_{ST}}{2}}^{\frac{N_{ST}}{2}} C_{l,n} e^{j2\pi l \Delta_f \frac{m}{f_{sample}}} \tag{3.6}$$

with $m = 0, 1, 2, .., T_U \cdot f_{sample} - 1$.

The Inverse Discrete Fourier Transformation (IDFT) is defined as:

$$F[y] = \frac{1}{N} \sum_{x=0}^{N-1} f[x] e^{j2\pi \frac{xy}{N}} \tag{3.7}$$

with x and y=0,1,2,...,N-1. Equation 3.6 can be written in the form of Equation 3.8:

$$\widetilde{s_n}[m] = \sum_{x=0}^{N-1} f_n[x] e^{j2\pi \frac{xm}{N}} = N \cdot IDFT(f_n[x]) \tag{3.8}$$

and with $N = f_{sample}/\Delta_f$ and

$$f_n = [0\, C_{1,n}\, C_{2,n} \ldots C_{26,n}\, 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\, C_{-26,n} \ldots C_{-2,n}\, C_{-1,n}] \tag{3.9}$$

HiperLAN/2 uses $N = 64$ (power of two) which enables the use of the efficient IFFT algorithm to calculate the time signal.

Adding the cyclic prefix

The cyclic prefix can be generated by copying the last 16 complex time samples of the useful data part of an OFDM symbol and transmit them before

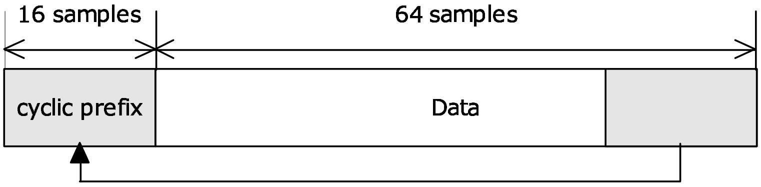


Figure 3.7: Timing definitions.

transmitting the regular 64 samples. This is depicted in Figure 3.7. Thus, an OFDM symbol consists of 80 samples. The purpose of the prefix is to avoid ISI and moreover it relaxes time synchronization as the receiver only requires 64 samples of the OFDM symbol. The samples at the border of the symbol are discarded.

3.2.6 Physical burst

The HiperLAN/2 MAC frame is divided in six logical channels (Section 3.1). Logical channel are embedded in MAC burst which is generated by preceding one or more logical channels with special OFDM symbols, i.e. *preambles*, that are known to the receiver. There exists three MAC bursts: the BCH, downlink and uplink MAC burst. For simplicity, we only researched the transmission and reception of one of these bursts, the BCH burst[1]. All preambles are generated by using the same building blocks, *preamble sections*, A, B and C. The BCH channel uses all preamble sections (Figure 3.8).

- section A: preamble section A consists of five equal parts of 16 samples: **A IA A IA IA**[2]. It is a special OFDM symbol with 12 loaded subcarriers.

- section B: preamble section B is a special OFDM symbol also with 12 loaded subcarriers and consists of five equal parts of 16 samples: **B B B B IB**.

- section C: preamble section C has a duration of 160 samples and contains a OFDM symbol that is transmitted twice. This OFDM symbol is a symbol with all carriers loaded and can be used for channel estimation. In addition, this section has an Cyclic Prefix (CP) of 32 samples.

Before transmission, both preamble A and B are multiplied with $\sqrt{\frac{13}{6}}$ [17].

[1] Every MAC frame starts with a BCH burst and only the uplink and downlink burst contain user bits. For simplicity we consider in this thesis a BCH burst with user bits.

[2] $IA = -A$ (sign inversion)

Modulation type	K_{mod}
BPSK	1
QPSK	$\frac{1}{\sqrt{2}} \approx 0.70711$
16-QAM	$\frac{1}{\sqrt{10}} \approx 0.31623$
64-QAM	$\frac{1}{\sqrt{42}} \approx 0.15430$

Table 3.3: Modulation-type dependent normalization factor.

Parameter		**Value**	
Sampling rate	f_{sample}	20 MHz	($\triangleq 1/T$)
Symbol interval	T_S	$4.0\mu s$	$(= 80 \cdot T)$
Useful symbol part duration	T_U	$3.2\mu s$	$(= 64 \cdot T)$
Cyclic prefix duration	T_{CP}	$0.8\mu s$	$(= 16 \cdot T)$
Number of data carriers	N_{SD}	48	
Number of pilot carriers	N_{SP}	4	
Total number of carriers	N_{ST}	52	
Subcarrier spacing	Δ_f	0.3125 MHz	$(= 1/T_U)$

Table 3.4: HiperLAN/2 OFDM parameters.

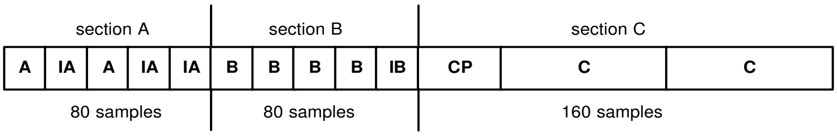


Figure 3.8: BCH Preamble.

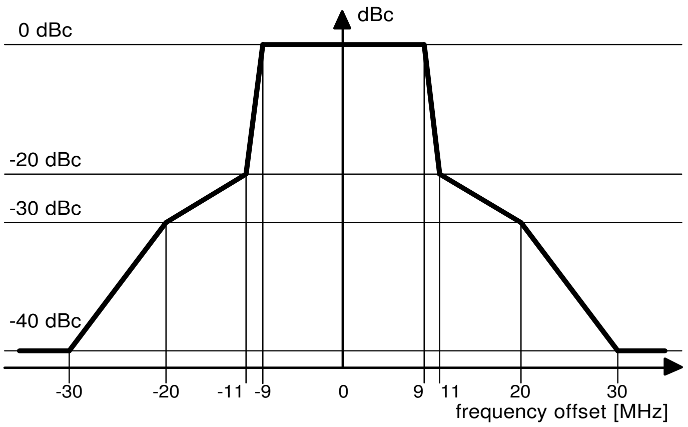


Figure 3.9: Transmit power mask [17].

3.2.7 Radio transmission

The final step in the transmitter provides: DA conversion and mixing to RF frequencies. In addition, the transmitted signal has to comply to the power transmit mask which are shown in Figure 3.9. The analog reconstruction filters used in the DA conversion can be used for this purpose.

3.3 Receiver

The ETSI only defines the transmitter. Therefore we are free to choose a receiver architecture as long as it meets the requirements, set by ETSI. A commonly used OFDM receiver architecture is depicted in Figure 3.10.

This architecture consists of ten functions:

- Channel selection
- Synchronization/parameter estimation
- Frequency-offset correction
- Inverse OFDM
- Channel equalization
- Common phase offset detection and correction
- De-mapping
- De-interleaving
- FEC decoding

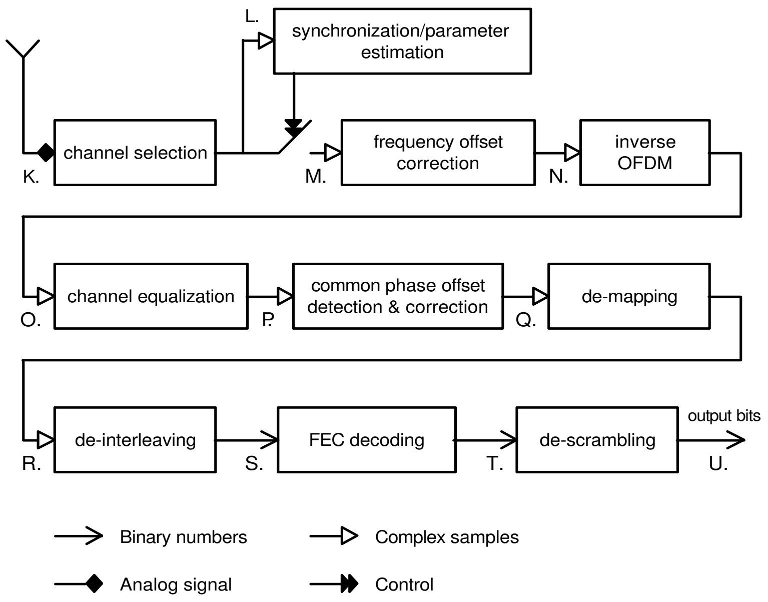


Figure 3.10: Data flow between the functions of the HiperLAN/2 PHY layer in the receiver.

- De-scrambling

In Figure 3.10 the data flow between these functions is outlined. The receiver start with point K, the antenna input. The first step is *channel selection*. This selects the used frequency channel by suppressing other channels (filtering) and in addition, mixes the channel to baseband. This step also includes Analog-to-Digital (AD) conversion. Then, in the next step, *Synchronization/parameter estimation*, the receiver searches for the start of a MAC burst i.e. preambles. If found, it also estimates the frequency offset and the channel transfer function in the frequency domain by using the preambles.

An frequency offset results in *inter-subcarrier* interference, so the first step in demodulation of the data is to correct the frequency offset. This is achieved by multiplying the signal with an negative frequency. Then, the signal is converted to the frequency domain by using an FFT (the reverse of Section 3.2.5). In the frequency domain, the channel and the common phase offset are corrected. The resulting signal constellations at point Q are then de-mapped into raw bits. The last steps of the PHY layer convert the raw bits to user bits by de-interleaving, FEC decoding and de-scrambling of the raw bits.

In the demonstrator we have implemented point K to point R. The de-interleaving and FEC decoding entities are implemented as well in the baseband simulation model in order to compare the BER performance with values from literature.

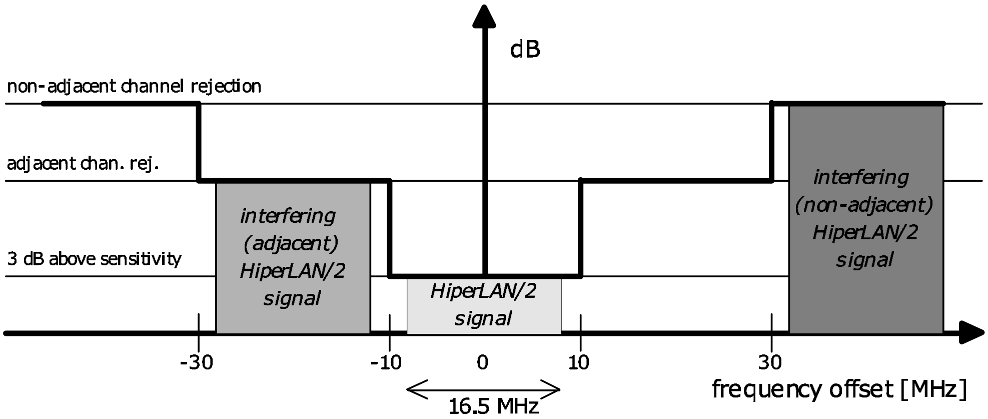

dB
non-adjacent channel rejection
adjacent chan. rej.
3 dB above sensitivity
interfering (adjacent) HiperLAN/2 signal
HiperLAN/2 signal
interfering (non-adjacent) HiperLAN/2 signal
-30
-10
0
10
30
16.5 MHz
frequency offset [MHz]

Figure 3.11: HiperLAN/2 interference definitions.

Nominal bit rate [Mbit/s]	minimum sensitivity	adjacent chan. rejection	non-adjacent chan. rejection
6	-85 dBm	21 dB	40 dB
9	-83 dBm	19 dB	38 dB
12	-81 dBm	17 dB	36 dB
18	-79 dBm	15 dB	34 dB
27	-75 dBm	11 dB	30 dB
36	-73 dBm	9 dB	28 dB
54	-68 dBm	4 dB	23 dB

Table 3.5: Adjacent channel and non-adjacent channel rejection requirements [17]

3.3.1 Channel selection

Channel selection incorporates two functions: selecting the wanted channel and suppression of unwanted channels. The wanted channel is selected by mixing the RF signal to baseband.

The other step, suppressing of unwanted channels, is mainly dictated by the HiperLAN/2 standard: The Packet-Error Rate (PER) has to be lower than 10% for a number of test conditions. In these experiments, the signal offered to the receiver consists of a wanted in-band signal (representing the information for a particular channel) and unwanted interfering signals. So, for example, in the adjacent channel rejection test conditions there is only one adjacent interferer and its power is increased until the PER exceeds 10%. The minimal requirements for each interferer is shown in Table 3.5 and Figure 3.11.

A rule of thumb is to design the channel selection filter in such a way that all interferers are suppressed 3 dB below the noise level required for a PER of 10%. This function also includes AD conversion.

3.3.2 Synchronization/parameter estimation

We assume that the system parameters are constant during a MAC burst, because of the HiperLAN/2 standard requirements, the coherence time of the channel (≈ 10 ms) and the introduced phase noise by the oscillators [41]. Therefore, parameters, such as the frequency offset, the channel transfer function in the frequency domain and timing, only have to be estimated per burst.

Timing synchronization and frequency offset estimation

Before the receiver can demodulate data symbols, it has to find the start of a MAC burst. All OFDM synchronization algorithms use some kind of correlation-based time synchronization because the preamble structure contains sections with equal parts:

$$M[j] \quad = \quad \sum_{m=0}^{C-1} \widetilde{r}^{*}[j+m] \cdot \widetilde{r}[j+m+L] \tag{3.10}$$

where,

1. $\widetilde{r}[m]$: The received complex signal samples.

2. C: The length of the two 'segments' that are correlated.

3. L: The *distance* between the segments that are correlated.

4. j: The time index of the start of the first segment that is correlated. The peak value of the correlation represents the *estimated start time* of the MAC burst.

Most algorithms perform joint timing synchronization and frequency offset estimation, thus after the optimal timing is found, the frequency offset is estimated as well. Algorithms that use joint estimation include:

1. Hanzo & Keller algorithm [42]

2. Schmidl & Cox algorithm [43]

3. van der Beek algorithm [44]

In our project we have used the Schmidl & Cox algorithm because it is a very simple algorithm and has a good performance. More information about these synchronization algorithms and their performance can be found in [45].

Two effects can cause an frequency offset in the received signal. The first is an Doppler shift which is introduced by moving terminals. The standard allows a maximum Doppler shift of 42 Hz [37] which is negligible compared with the second effect. This effect is caused by a difference in the oscillator frequency of the transmitter and receiver. The ETSI allows an minimal oscillator accuracy of 20 parts per million which is $\approx$ 120 kHz[3]. So the used LOs should have a better accuracy. If the receiver has the same oscillator accuracy, the maximum frequency offset is 240 kHz. It is for the same reason that timing synchronization does not need to be tracked because the maximum symbol drift during a MAC frame is $\approx$ 1.6 sample. The MAC frame consists of multiple parts, such as downlink or uplink phase, which have a separate preamble section. The receiver has to synchronize on each part and therefore the maximum symbol drift for each burst is smaller than 1 sample.

The Schmidl & Cox algorithm uses the following Timing Metric (TM) [46]:

$$TM[j] = \frac{\mid P[j] \mid^2}{(R[j])^2} \tag{3.11}$$

with

$$P[j] = \sum_{m=0}^{L-1} \widetilde{r}[j+m] \cdot \widetilde{r}^*[j+m+L] \tag{3.12}$$

and

$$R[j] = \sum_{m=0}^{L-1} \mid \widetilde{r}[j+m+L] \mid^2 \tag{3.13}$$

This metric can be seen as a normal correlation divided by the energy of the second segment.

If the optimal timing, j_{opt} is found, the frequency offset estimation is:

$$\widehat{\delta f}[j_{opt}] = \frac{N}{L} \Delta f \frac{tan^{-1}(P[j_{opt}])}{2\pi} \tag{3.14}$$

with N the FFT length i.e. 64.

Channel estimation

Channel estimation is performed in the frequency domain, using preamble section C as this section has all carriers loaded. Conversion to the frequency domain is performed by the inverse OFDM function (Section 3.3.4). In the current implementation we use the *Zero-forcing* equalizer for simplicity at the

[3] at 6 GHz Local Oscillator (LO) frequency

cost of performance degradation due to noise enhancement. A better equalizer is the Minimum Mean Square Error (MMSE) equalizer, but this equalizer requires estimation of the noise power which increases complexity. Furthermore, the de-mapping function uses hard decision which means that the output of this function are raw bits. So, if there is a carrier which has a channel transfer function of almost zero, both the Zero-forcing and MMSE equalizer algorithm cannot reconstruct the signal in a noise environment because it is lost. Therefore we expect that the performance degradation by using a zero-forcing equalizer is acceptable.

The zero-forcing equalizers estimates the channel for each carriers with:

$$\widehat{H}_{f,l} = \frac{\widehat{C}_{l,preambleC}}{C_{l,preambleC}} \tag{3.15}$$

with $\widehat{C}_{l,preambleC}$ the complex value of the received l^{th} carrier and $C_{l,preambleC}$ the l^{th} transmitted carrier value of the preamble C section. Preamble C is transmitted twice, so the channel can estimated more accurate by using both symbols.

3.3.3 Frequency offset correction

A frequency offset causes *inter subcarrier* interference [37]. This offset can be corrected both in the time and the frequency domain. Correction in the time domain is preferable because this requires multiplication with a complex value (negative frequency) and not a convolution as in the frequency domain.

Let the received signal be defined as:

$$\widetilde{r}[k] = \widetilde{s}[k]e^{\jmath(2\pi k\delta f/T+\theta_0)} \tag{3.16}$$

with $\widetilde{s}[k]$ the transmitted complex baseband signal, δf the frequency offset, T the symbol time and θ_0 the phase offset.

The received signal can be corrected by multiplying it with the negative frequency:

$$\widetilde{r_c}[k] = \widetilde{r}[k]e^{-\jmath(2\pi\frac{k}{T}\widehat{\delta f}+\theta_1)} = \widetilde{s}[k]e^{\jmath(2\pi\frac{k}{T}\delta f+\theta_0)}e^{-\jmath(2\pi\frac{k}{T}\widehat{\delta f}+\theta_1)} \approx \widetilde{s}[k]e^{(\jmath\theta_0+\theta_1)} \tag{3.17}$$

A more efficient and better way of correcting the frequency offset is to calculate the frequency offset correction values only for the first OFDM symbol, because a time difference of n samples results in a phase offset:

$$e^{\jmath(2\pi\frac{k+n}{T}\delta f)} = e^{\jmath(2\pi\frac{k}{T}\delta f+2\pi\frac{n}{T}\delta f)} = e^{\jmath(2\pi\frac{k}{T}\delta f+\theta_n)} \tag{3.18}$$

The resulting phase offset is corrected in the *common phase offset detection and correction* block (Section 3.3.6) which estimates the common phase offset for each OFDM symbol. Moreover, larger errors are allowed in the frequency offset estimation because the estimation error only influences the current OFDM symbol and not the whole MAC burst.

3.3.4 Inverse OFDM

This block converts the time signal into the frequency domain by using an FFT (opposite of Section 3.2.5):

$$\widehat{C}_{m,n} = \sum_{x=0}^{N-1} \widetilde{r_c}[x]e^{-\jmath 2\pi \frac{xm}{N}} = N \cdot DFT(\widetilde{r_c}[x]) \tag{3.19}$$

with $\widehat{C}_{m,n}$ the m^{th} subcarrier value of the n^{th} OFDM symbol and $N = f_{sample}/\Delta_f$ which is in the HiperLAN/2 case equal to 64. As N is a power of two, the efficient FFT algorithm can be used to calculate the frequency signal[4].

3.3.5 Channel equalization

This block corrects the channel which is estimated in the *synchronization and parameter estimation* part (Section 3.3.2).

All carriers are multiplied with the inverse of the channel:

$$\widehat{C}^a_{m,n} = \widehat{C}_{m,n} \cdot \widehat{H}^{-1}_{f,m} \tag{3.20}$$

with $\widehat{C}_{m,n}$ the value of subcarrier m of the n^{th} received OFDM symbol.

3.3.6 Common phase offset detection & correction

Each OFDM symbol contains 4 pilot carriers (Section 3.2.5) which can be used for estimation of the common phase offset detection:

$$\widehat{\phi}_{common} = (tan^{-1}(\widehat{C}^a_{-21,n}) + tan^{-1}(\widehat{C}^a_{-7,n}) \tag{3.21}$$

$$+tan^{-1}(\widehat{C}^a_{7,n}) - tan^{-1}(\widehat{C}^a_{21,n}))/4 \tag{3.22}$$

All data carriers are multiplied with the inverse phase offset to get the constellation correct:

$$\widehat{C}^b_{m,n} = \widehat{C}^a_{m,n} \cdot e^{-\jmath \widehat{\phi}_{common}} \tag{3.23}$$

[4] Another option is to use a 128 or 256-point FFT. In this case, adjacent channels will appear in unused bins of the FFT. So, this solution can reduce the requirements of the channel selection function. However, reducing the requirements of the filters will also affect the timing and parameter estimation function in a negative manner.

3.3.7 De-mapping

De-mapping can be implemented in different ways, but in our receiver we have implemented de-mapping by using a lookup table.

For example, both the real and imaginary part of 64-QAM have values $\{-7, -5, -3, -1, 1, 3, 5, 7\}$. These values can be converted to an index by adding 7 and divide it by 2. (The range is now 0...7.) As the received values contain noise, the receiver has to check for invalid indices. The generated index is used in a lookup table which contains Gray-coded values.

3.3.8 De-interleaving

This block de-interleaves the incoming bit stream to the orginal bit sequence. In the receiver we used the IT++ library for this function [39].

3.3.9 FEC decoding

The block decodes the FEC-coded bits into user bits by using the Viterbi algorithm. For this function we also used the IT++ library.

3.3.10 De-scrambling

This function de-scrambles the bits. The function is not implemented in the baseband simulation model, as our simulations already use random bits.

Chapter 4

SDR transceiver

4.1 Introduction

Chapter 1 has introduced the definitions of Software Radio (SR) and has described the goals of our project: to demonstrate a design that is expected to be feasible within a few years with respect to manufacturing costs and power consumption. Our vehicle is a Bluetooth-enabled HiperLAN/2 receiver to demonstrate our ideas about SDR for wireless LAN standards. In fact, by building this receiver it is expected that we can integrate other wireless LAN standards very easily because they use similar modulation techniques and frequency bands. The physical layer of both standards is described in Chapter 2 and Chapter 3. This chapter discusses the project in more detail. The physical layers of both standards are completely different and therefore result in different receiver architectures. First, the chosen RF receiver architecture is motivated and the interface between the analog and digital domain is defined. In the digital domain, the functional architecture of both receivers is (also) different and implementation of our SDR receiver requires the integration of both receivers at a functional level. The second part of this chapter discusses several implementation alternatives for this functional architecture. In addition, the platform that we use for the demonstrator is motivated.

4.2 Analog-digital partioning

One of the objectives of the project is to implement the channel selection function and demodulation function of a combined HiperLAN/2 and Bluetooth receiver. Chapter 1 has introduced a layered protocol model to define to scope of the project. This model is shown in Figure 4.1 for reasons of convenience. Part of the channel selection function is provided by *analog signal conditioning* and part by digital processing. The *demodulation* function is fully digital, see Figure 4.2.

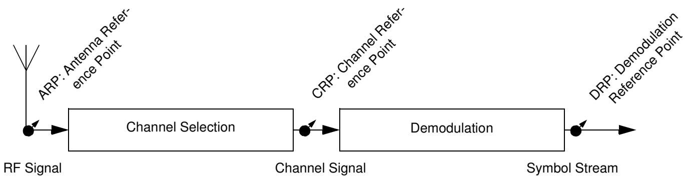

ARP: Antenna Reference Point
CRP: Channel Reference Point
DRP: Demodulation Reference Point
Channel Selection
Demodulation
RF Signal
Channel Signal
Symbol Stream

Figure 4.1: Signal Path Functions Model (SPFM).

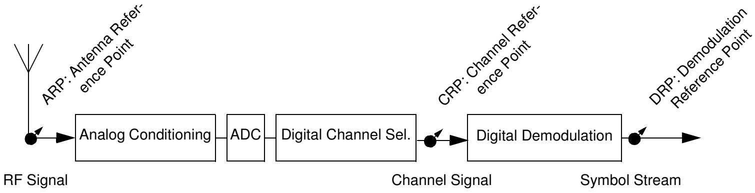


Figure 4.2: Analog-digital partioning.

Focus of this thesis is on the digital part of the receiver. For the demodulation function, which is performed completely in the digital domain, the chosen analog front-end has not much influence. In other words, integration of both demodulators does not depend on the analog front-end. However, some of the signal distortions are caused by the chosen analog architecture which have to be compensated in or before the demodulator. For example, a zero-IF receiver introduces a DC offset in the output signal, whereas other architectures will not have this distortion. Thus, the demodulator function may need extra functions to correct specific front-end distortions. The other function, channel selection, is performed in combination with the analog front-end and its architecture depends heavily on the analog part. For these reasons, a short overview of analog front-end architectures is given and the chosen front-end architecture in our demonstrator is motivated. Due to time constraints and because at this moment there not yet sufficient data available about the wideband SDR analog front-end, a detailed discussion has been left out of this thesis. It is although a very interesting and important area of research for SDR receivers. More information about the analog front-end and it's distortions can be found in the Ph.D. thesis of V.J. Arkesteijn [47].

4.3 RF receiver architectures

Important criteria in selecting a receiver design are complexity, cost, power dissipation and the number of external components [48]. Most popular receiver architectures are the *super-heterodyne*, the *zero-IF* and the *digital-IF* receiver architecture that are shown in Figure 4.3.

In the super-heterodyne receiver, the first step is to select the wanted frequency band by using a Band-Pass Filter (BPF) filter. In case of a HiperLAN/2 receiver, this includes the whole frequency band covered by the standard. The signal is then amplified by the Low-Noise Amplifier (LNA) and mixed to an Intermediate Frequency (IF). In most cases this step is fixed, so the Local Oscillator (LO) is also fixed at certain frequency. After mixing, a second BPF filter removes unwanted mixer products and the signal is amplified by an Automatic Gain Control (AGC) if passive mixers are used. The second receiver stage performs channel selection by mixing the wanted channel to baseband. A baseband signal is a complex signal and for this reason, this step requires two mixers for both the In-phase component (I) and Quadrature-phase component (Q). Unwanted mixer products are suppressed by a Low-Pass Filter (LPF) and the complex signal is digitized by two ADCs. The *digital baseband processing* block performs additional filtering and demodulates the signal to raw bits.

Advantages of this architecture include [49] good adjacent channel selection (with the use of external Surface Acoustic Wave (SAW) filters) and no DC offset problems (which is important for OFDM demodulation). However, the main drawbacks of this architecture is the use of external components (SAW filters) which increases manufacturing costs. Moreover, this architecture is tailored/designed for one standard. To support multiple standards (2.4 and 5 GHz band), this architecture would require that all stages in the analog front-end are more flexible. Especially for external components, this is a difficult task.

The second architecture is called the zero-IF architecture. The RF signal is mixed in a single stage to baseband. This receiver has less components than the super-heterodyne receiver which is important for single-chip solutions. For multi-standard support, this architecture (and other architectures) requires that the first BPF filter is suited for multiple frequency bands. A BPF filter with a very wide passband (e.g. $2 - 6$ GHz) is not possible because in this case, the LNA requirements are too severe. A solution to this problem is to use separate filters for each frequency band. Other components in the architecture have to be designed for the standard with the largest signal bandwidth. Once the signal is digitized additional channel selection can be performed in the digital domain.

A zero-IF receiver has a number of problems which do not exist in a super-heterodyne receiver. The first problem is DC-offset. Due to capacitive, substrate and bond-wire coupling there is no perfect isolation between LO and the input of the mixer/LNA. This effect is called *LO-leakage* which results in self-mixing of the LO frequency. As a result, a DC offset appears in the output signal. Large DC offsets can saturate analog components (e.g. ADCs). In addition, in the digital domain, it can corrupt synchronization/parameter estimation algorithms. Moreover, a DC offset in combination with a frequency offset, leaks into neighbouring subcarriers for OFDM systems [50].

The second issue in a zero-IF receiver architecture, is a mismatch between the I and Q path. This mismatch is caused by two effects. The first effect is that the two paths do not have a phase difference of exact 90° and the second effect is a gain difference between both paths. Especially, for wideband receivers it is difficult to have 90° phase difference for all frequencies. A mismatch between both paths can be compensated in the digital domain, but will degrade the performance as it introduces crosstalk between positive and negative frequencies (See Section 7.1.1).

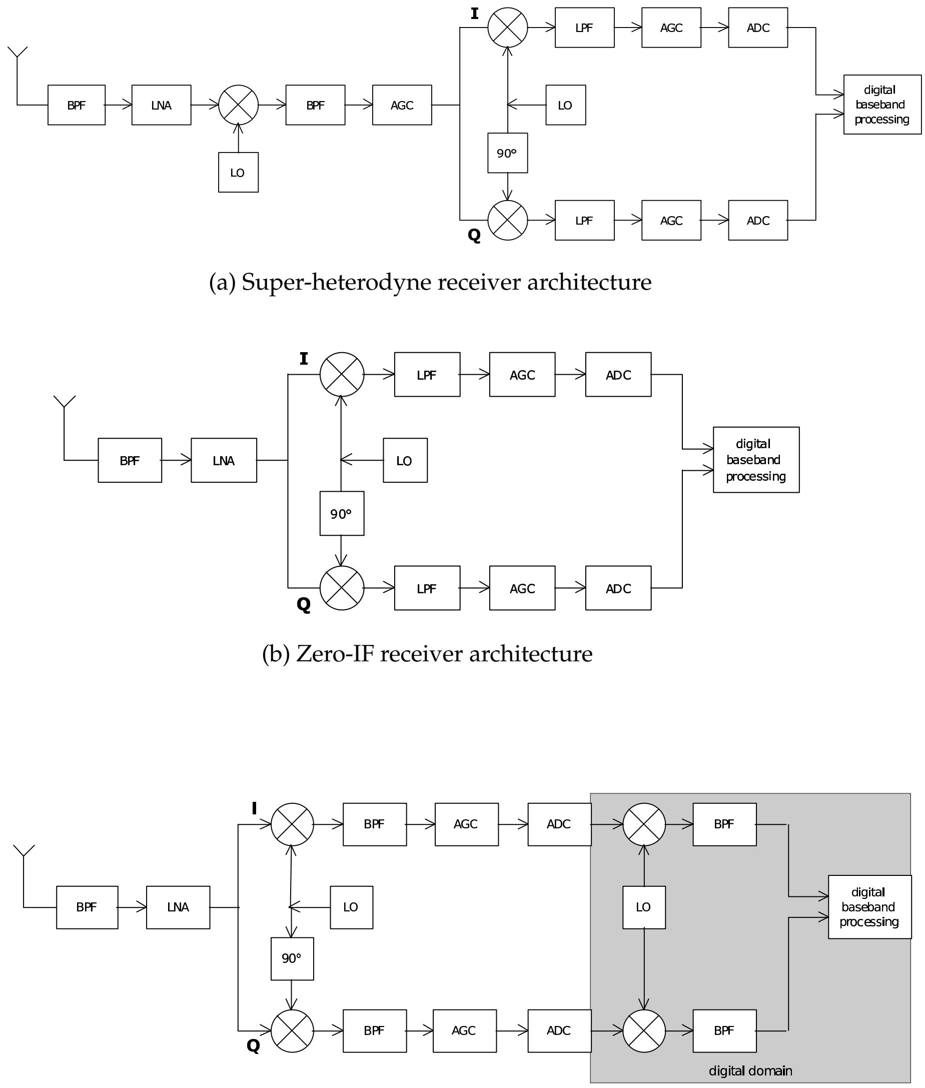
The last receiver architecture is the digital-IF architecture which is very similar to the zero-IF architecture. The wanted signal is not mixed to baseband but to a low frequency e.g. 10 MHz and for that reason does not need to be complex anymore. However, this architecture still requires two ADCs, because there is an unwanted channel at -10 MHz which cannot be removed by the analog front-end[1]. The ADC requirements are higher for this architecture compared with a zero-IF receiver for two reasons, higher dynamic range and higher input bandwidth. The input of the ADCs has a higher dynamic range because the unwanted channel can be much stronger than the wanted signal. Moreover, the input bandwidth of the ADC has to be much larger because the unwanted channel has to be converted as well. An advantage of this receiver type is that the output signal does not contain a DC offset.

4.3.1 Analog front-end architecture

For a flexible receiver, the zero-IF and digital-IF architecture are suitable because they do not need external components and can be integrated into a single chip. Power consumption is important for mobile terminals and according to [51] most power is already consumed in the ADCs ($> 80\%$) for a zero-IF receiver. It is expected that for a digital-IF receiver the power consumption of the ADCs is much higher. Therefore, we have chosen in this project to use a zero-IF receiver design in HiperLAN/2 mode. Goal in the analog part of the project is to integrate the *down-conversion* part in a single chip. The down-conversion part consists of the LNA and mixing stage (See Figure 5.4, page 81).

One of the disadvantages of a zero-IF receiver is DC offset. A small DC offset in combination with a frequency offset can already degrade the performance of OFDM receivers. See Section 7.1.1 for more information. In a

[1] Another option is to use an external SAW filters in the RF front-end with a passband equal to the channel bandwidth which is not a feasible option for SDR applications.



(a) Super-heterodyne receiver architecture

(b) Zero-IF receiver architecture

(c) Digital-IF receiver architecture

Figure 4.3: RF receiver architectures.

normal, dedicated OFDM receiver the analog and digital domain are often separate blocks and DC offset is corrected entirely in the analog domain. In our project we are more flexible and can solve this problem in both domains. In our design, DC offset has to be detected and corrected entirely in the digital domain as long as the output signal of the analog RF front-end does not saturate the ADCs. This relaxes the design of the analog front-end and thus the total design. It should be noted that due to time contraints, we did not implement a DC offset detection and correction algorithm. However, this problem is in the digital domain not difficult to solve and can be solved easily by computing for example the average value of the input signal.

4.3.2 Demonstrator digital functionality

For both standards, the first step after ADC conversion is to filter and decimate the 80 Million Samples Per Second (MSPS) signal to 20 MSPS (i.e. decimation filter). The resulting signal contains one channel in HiperLAN/2 mode and twenty channels in Bluetooth mode. Therefore, the digital part has to select in Bluetooth mode one channel by mixing it to baseband and filter the signal with a LPF filter to suppress unwanted channels. The second function of the digital part is demodulation. This step has to be capable of demodulating both GFSK and OFDM signals. The next section describes how the digital functionality of both standards can be integrated at a functional level. This integration is important for every implementation alternative of the digital part.

4.4 Functional architecture of the SDR receiver

For every implementation of our SDR receiver, integration of both receiver at a functional level is important. However, in our project we have chosen to implement the demodulation function in software. For that reason we have implemented two separate demodulators in software. This does not affect the flexibility of the built SDR receiver. On the other hand, for an implementation of our SDR receiver in hardware, integration of both demodulators is important. For this reason, we derive in this section a functional architecture of the SDR receiver by looking at the required number of operations for each block. Other important aspects, such as throughput rates, latency are left out the discussion because these aspects are less important for a software implementation of both demodulators. The functional architecture of the Bluetooth receiver has been described in Chapter 2 and the HiperLAN/2 receiver in Chapter 3.

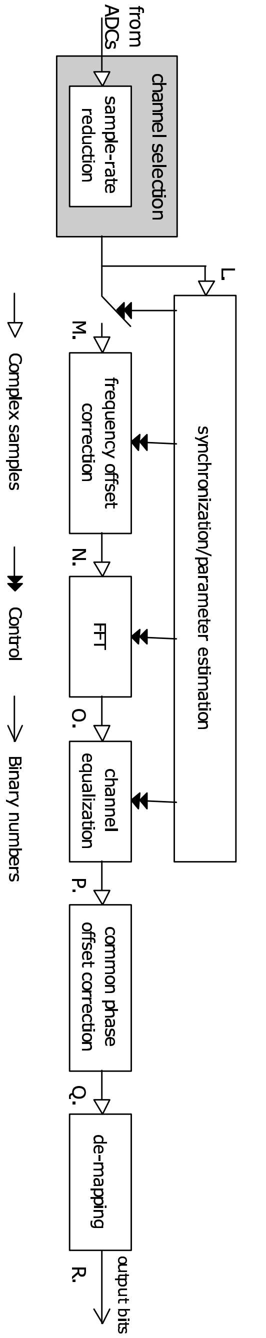


Figure 4.4: Signal processing architecture of the HiperLAN/2 receiver.

function	**MOPS**
sample-rate reduction	1240
synchronization and parameter estimation	4
frequency offset correction	39
FFT	230
channel equalization	39
phase offset detection and correction	40
64-QAM de-mapping	77
total	**1669**

Table 4.1: Computational requirements of the HiperLAN/2 receiver functions (64-QAM mode) using the user scenario of Chapter 5.

4.4.1 HiperLAN/2 receiver

Figure 4.4 depicts such an signal processing architecture for the HiperLAN/2 receiver. The first step is sample rate reduction by filtering the signal with a low-pass filter and decimate it to a 20 MSPS signal. To suppress adjacent channels below the required Signal-to-Noise Ratio (SNR), 31-tap symmetric Finite Impulse Response (FIR) filters are sufficient [52]. See Figure 4.5. The analog filters suppresses far out-of-band regions good enough. Therefore, the design is based on a band-stop design with loose requirements in the frequency range from 30 to 40 MHz (non-adjacent channel) to lower the filter requirements. This filter is implemented as polyphase filter to lower the digital signal processing requirements. Because this filter is fixed in our SDR receiver, it can be implemented in hardware.

The demodulation function starts by searching for the start of a MAC burst and to acquire symbol timing. After that, it estimates the frequency offset and channel parameters. Then, the data OFDM symbols can be demodulated by first correcting the frequency offset, performing an FFT, correcting the channel and detecting and correcting the phase offset by using the pilot tones. The output are Quadrature Amplitude Modulation (QAM) symbols which have to be de-mapped into raw bits. The required number of Million Operations Per Second (MOPS) for each block of the HiperLAN/2 receiver is listed in Table 4.1. It can be seen that the sample-rate reduction filters is the most demanding function of the HiperLAN/2 receiver. As mentioned before, this filter is fixed in our SDR receiver and therefore it can be implemented in hardware without affecting the flexibility. For this reason we did not pay much attention to this part of the receiver.

4.4.2 Bluetooth receiver

Figure 4.6 depicts the signal-processing architecture for the Bluetooth receiver. The first step, channel selection, can be divided in three parts:

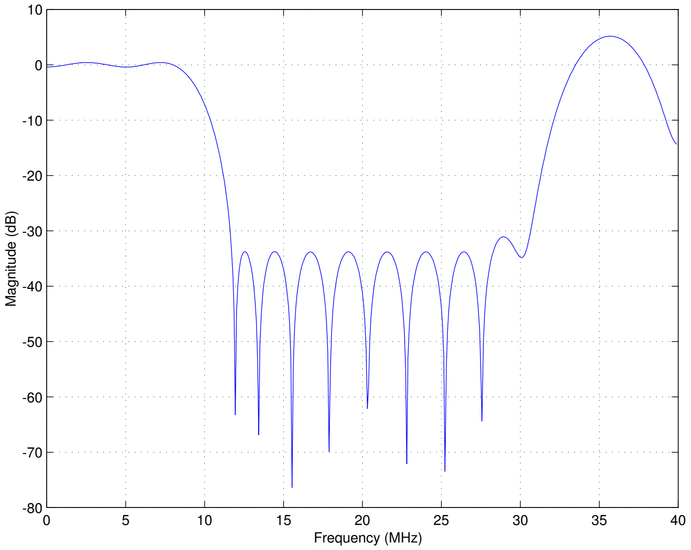


Figure 4.5: Filter characteristics of the sample-rate reduction filter (f_s = 80 MHz).

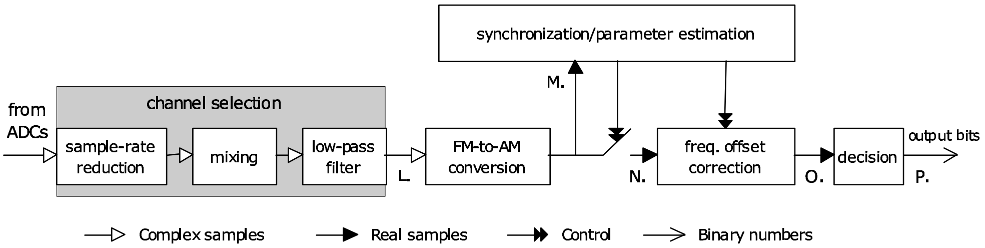


Figure 4.6: Signal processing architecture of the Bluetooth receiver.

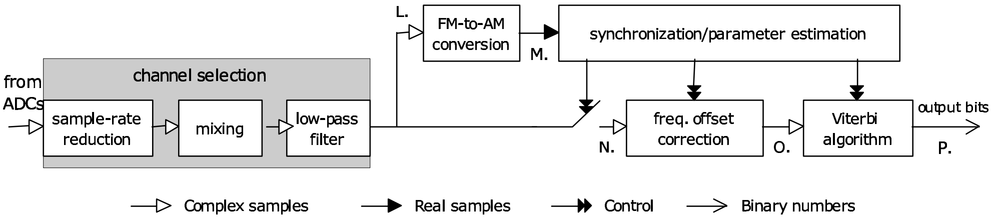


Figure 4.7: Signal processing architecture of the Bluetooth MAP receiver.

function	MOPS
sample-rate reduction	1240
mixing	100
low-pass filtering	260
synchronization and parameter estimation	t.b.d.
frequency offset correction	9
MAP receiver	57
total	**1666**

Table 4.2: Computational requirements of the Bluetooth receiver functions (MAP receiver) using the user scenario of Chapter 5.

- Sample-rate reduction (from 80 to 20 MSPS) (equal to the sample-rate reduction filter in HiperLAN/2 mode)
- Mixing of the wanted channel to baseband
- Removing adjacent channels

The next step is to demodulate the GFSK signal into an AM signal by taking the derivative of the phase. Because this demodulator converts frequency variations into amplitude variations, an frequency offset results in an DC offset in the output signal. This offset has to be corrected before bit detection.

Both the OFDM and the Park's demodulator [30] require a complex input, which is an opportunity for integration. The Park's baseband FM demodulator is used in our project but it has two disadvantages: apparently a large difference with the theoretical performance. This is probably caused by the Gaussian filter that introduces a memory effect in the signal. Single-bit detectors do not utilize this effect. In addition, this demodulator does not specify a pre-detection filter for optimal decision. The latter results in a channel selection filter based on trial and error. For these reasons, we have chosen another demodulator, the MAP receiver (Figure 4.7) which has a better performance and moreover it defines an optimal pre-detection filter. This filter is given by the Laurent decomposition [31] that describes the GFSK signal as a sum of linear, orthogonal, PAM waveforms. The first waveform is only important for Bluetooth signals because it contains more than 98 % of the signal energy. The output is an amplitude-modulated signal which is very similar to QAM modulation and the Viterbi algorithm is used for bit detection. Figure 4.7 shows the signal-processing architecture for the Bluetooth MAP receiver. The required number of MOPS for each block is shown in Table 4.2.

4.4.3 Bluetooth-enabled HiperlAN/2 receiver

In both receivers, the sample-rate reduction function (i.e. decimation filter) requires most computations per second. As its functionality is fixed for both

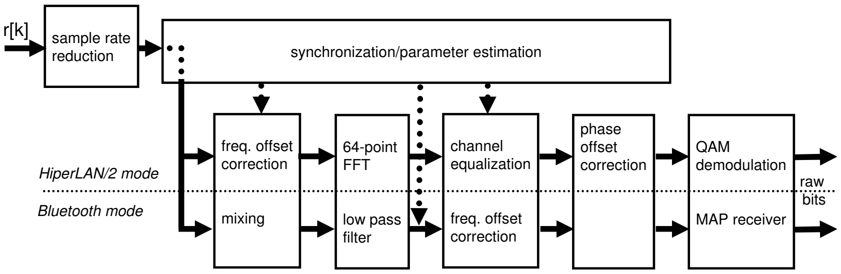


Figure 4.8: Functional architecture of the Bluetooth-enabled HiperLAN/2 receiver.

standards, it can easily be implemented in hardware without affecting the flexibility of the SDR receiver. In the HiperLAN/2 demodulator, the FFT has the highest computational requirements (Table 4.1), whereas, in Bluetooth mode the low-pass filter in the channel selection function requires most processing power (Table 4.2). Both filtering and FFT incorporate multiplications and additions and therefore it is possible to combine them. Low-pass filtering in the frequency domain is not an option because the *overlap-add* method [53] requires more computations[2].

Other combinations are now very straightforward. The HiperLAN/2 equalizer which incorporates complex multiplications can be combined with the Bluetooth frequency-offset correction that also uses complex multiplications. Finally, the MAP receiver can be combined with the QAM demodulator. In our current implementation of the QAM demodulator we use hard decision which means that the output of this function are raw bits. Better performance can be achieved if this demodulator is combined with the FEC decoder which also contains a Viterbi-algorithm [54] just as the MAP receiver.

In this section we have integrated the functional architecture of Bluetooth into the HiperLAN/2 architecture. This integration can also be seen in a wider scope. In fact, every phase-modulation standard can be integrated into an OFDM receiver. So, we expect that this SDR functional architecture can be used for other wireless LAN standards as well.

4.4.4 SDR architecture

The ability of the SDR architecture to support a communication standard is perdominantly determined by [55]:

1. the largest instantaneous signal bandwidth

[2] This method requires for this application at least a 128-point FFT and a small 8-point IFFT (decimation and mixing can be performed after the FFT). As a result more computations are required than *normal* filtering in the time domain.

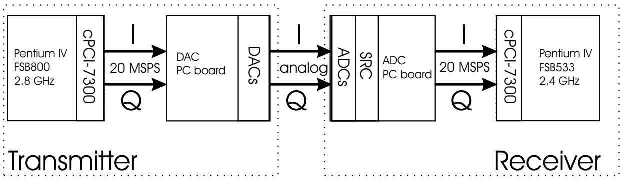


Figure 4.9: Block diagram of the SDR testbed (baseband only).

2. the frequency range and bandwidth of the RF front-end

3. the ADC sampling rate

4. the maximum dynamic range

5. the DSP processing capacity

In our application, the channel bandwidth in HiperLAN/2 is much larger than the bandwidth of a Bluetooth channel. So, a front-end which is capable of receiving both HiperLAN/2 and Bluetooth signals, is mainly designed for the reception of HiperLAN/2 signals which have a signal bandwidth of 20 MHz. Other wireless LAN standards do not have a larger bandwidth than HiperLAN/2, so our front-end is also expected to be suited for other standards.

Wireless LAN standards use the 2.4 and 5 GHz band, so for that reason, the second ability, the frequency range of the RF front-end has to include at least these two bands. The wideband analog SDR front-end has been designed to have a frequency range, from 500 MHz to 6 GHz. So, other wireless standards, such as GSM could be received as well. More information about the bandwidth and other parameters of the wideband analog front-end can be found in the Ph.D. thesis of V.J. Arkesteijn [47].

In Bluetooth mode, the output of the analog front-end is also a 20 MHz wide signal which contains 20 Bluetooth channels. As neighboring Bluetooth channels can be 40 dB stronger than the wanted channel [18], the ADC resolution should be at least 10 bit. (HiperLAN/2 has less stringent requirements.) In our project we use 12-bit COTS ADCs at 80 MSPS that where available at our chair. This is on the safe side but at the cost of a higher power consumption.

The HiperLAN/2 receiver has most DSP requirements compared with the Bluetooth receiver. Therefore, the DSP platform should have enough capacity for the reception of HiperLAN/2 signals. It is expected that other wireless LAN standards do not have more DSP requirements because they are very similar to the Bluetooth or HiperLAN/2 with respect to modulation technique and RF bands.

SDR challenges

In Section 1.4 we posed challenges which have to be solved in an SDR receiver. This section explains which design choices which have been made in order to solve these challenges:

- **Power consumption**. Because one of the goals of our project is to build an SDR which is feasible within a few years we have decided to focus on a flexible radio in a notebook where power consumption is less an issue.
- **Clock generation** In our SDR architecture, a single master clock (LO) is used, which runs at 80 MHz. Besides the Pentium 4, we use a COTS Programmable Down Converter (PDC) for channel selection. The output rates of this Programmable Down Converter (PDC) are derived from this clock, so the sample-rate conversion function is relatively simple. These rates are 20 MHz (HiperLAN/2) and 5 MHz (Bluetooth). The output signal of the PDC is buffered and transfered to the PC where all demodulation functions are performed.
- **Receiver architecture** As the receiver complexity is four times the transmitter complexity [15], we have tried to use for the most demanding standard, HiperLAN/2, algorithms which are rather simple but have good performance.
- **Handset production** Our SDR receiver uses few components because we have developed a CMOS-integrated wideband analog SDR front-end and use the notebook's CPU for signal processing purposes. This solution is relatively cheap to produce.

4.5 Implementation platform for the digital part of the receiver

The functional architecture of the SDR receiver can be mapped on a range of flexible platforms, varying from application specific platforms such as the *Chameleon* project [56] to more general platforms such as a DSPs and General Purpose Processors (GPPs). Our application is a flexible radio in a notebook, so existing processing resources of the notebook can be used for digital signal processing purposes. In our project, we have chosen to investigate to which extent a GPP processor can be used for our receiver algorithms.

A real-time software implementation of the demodulation function of the receiver (and modulation function of the transmitter) on the notebook's processor appeared to be unexpectedly possible. Although a GPP is very power in-efficient, it is already available in a notebook. So, this solution is cheap as it does not require extra DSP hardware. To test the digital part of the SDR receiver, we have built a baseband transceiver that consists of four blocks; a transmitter PC, a DAC Printed Circuit Board (PCB), a receiver PC and an ADC PCB.

In this thesis we have shown that a real-time software implementation of the modulation and demodulation function of the SDR transreceiver is possible using the notebook's processor. Modern CPUs have enough processing capabilities (computational capabilities and I/O speed) to perform this function. This research is an extension of the work of Bose [3] who showed in 1999 for second-generation mobile standards that it is possible to perform most baseband functions of the physical layer on a GPP processor. In his research, I/O was one of the main bottlenecks which is no longer true for current-day computers. Nowadays, an important bottleneck is the latency of the operating system, because wireless standards require fast response times.

Traditional operating systems such as Windows or Linux are non-real-time, e.g. the latency of the system is undefined and can be up to 100 ms for Linux [57]. So, it is possible that our transceiver *program* misses a buffer and data is lost. A solution to this problem is, for example, to apply special patches to the Linux kernel, which reduce this maximal latency to about 5 μs[57][3]. In our testbed we use large sample buffers of 100 ms to avoid the influence of the operating system but additional research is needed to find the maximal allowable latency which is likely determined by the MAC layer. Furthermore, we have to investigate if this value can be achieved in our testbed. So at the moment, our transceiver can only be used for continuous transmission of MAC bursts.

Future standards will probably have higher demands than HiperLAN/2, but on the other hand future GPP will also have more processing capabilities. So, it is unknown if a real-time implementation of the modulation and demodulation function of future standards on a GPP will still be possible. This thesis aims only on current-day wireless LAN standards and current-day GPPs. On the other hand, companies like Intel are integrating wireless LAN support into the chipset of notebooks [58]. This solution is not flexible, but will be sufficient for most consumers. So, it seems that in the near future, computers have already radio functionality onboard.

[3] A HiperLAN/2 MAC frame has a duration of 2 ms and the shortest Bluetooth MAC frame is 0.625 ms long.

Chapter 5

SDR testbed

5.1 Introduction

This chapter describes the testbed that has been built in the SDR project. First, an overview of the testbed is presented which is followed by a description of the developed software. In addition, user scenarios have been derived for both standards in order to estimate and measure the computational power requirements, necessary for executing the software on a Pentium 4. To avoid the influence of the operating system (Slackware Linux), we use in our testbed large sample buffers of 100 ms. This Pentium solution is compared with an implementation on a low-power DSP processor. The second part of this chapter describes the hardware of the testbed at system level.

5.1.1 Overview

Figure 5.1 shows the component architecture of our SDR testbed. The testbed consists of both a transmitter and a receiver. The transmitter contains a Personal Computer (PC) with a DAC board, an Agilent E4438C generator (Section 5.4.2) for up-conversion and an antenna. The receiver consists of an antenna, a wideband SDR analog front-end and a PC with an ADC board.

The transmitter PC continuously generates MAC bursts (HiperLAN/2 or Bluetooth) which are sent in real-time to the DAC board at 20 MSPS by using an Adlink cPCI-7300 digital Input/Output (I/O) PCI card [59]. The DAC board converts this digital signal into a complex analog baseband signal. The last step of the transmitter is up-conversion of the baseband signal by the Agilent E4438C generator and transmission of the RF signal. At the receiver side, the signal is picked up with an antenna that is connected to the analog SDR front-end. Its baseband output is sampled with 80 MSPS by the ADCs on the ADC board. After this conversion, the first step in the digital part is channel selection that is computed by the Intersil ISL5416 PDC [60]. This PDC

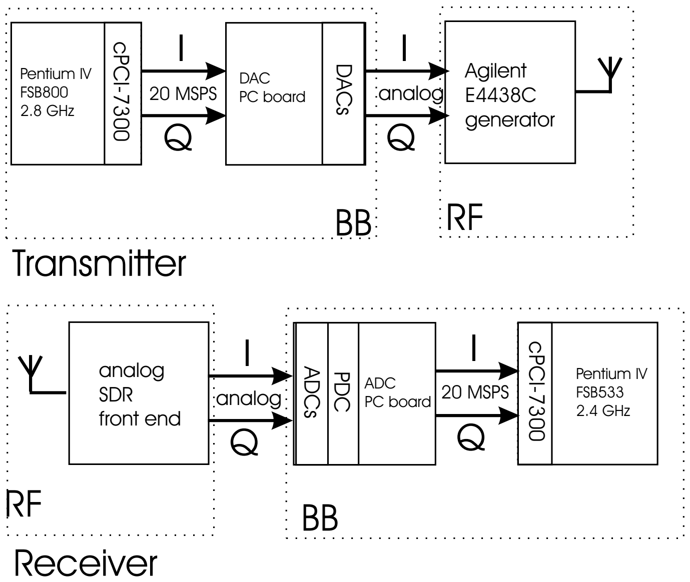


Figure 5.1: Component architecture of the SDR testbed.

decimates the digital signal into a complex 20 MSPS signal in HiperLAN/2 mode and into a 5 MSPS signal (including mixing of the wanted channel to baseband) for Bluetooth. Both the ADCs and PDC are mounted on the ADC board. The signal is then transported to the receiver PC by using another Adlink cPCI-7300 digital I/O PCI card. Finally, the receiver PC performs at the data rate all demodulation functions.

5.2 Software

In the previous chapter, we have presented the functional architecture of a Bluetooth-enabled HiperLAN/2 receiver. Such an architecture is especially important for hardware-oriented implementions. In our testbed, however, we have implemented the demodulation function fully in software in two separate programs and in this implementation alternative, the functional SDR architecture is less important.The channel selection function is computed by the Intersil ISL5416 PDC that is integrated on the ADC board.

Moreover, for the Bluetooth receiver we have implemented both the MAP receiver and Park's FM demodulator. Currently, we do not have a good algorithm for estimating the modulation index [34]. In a testbed environment, the modulation index is known, but in practical situations this is not the case. For these reasons, we have decided to use the Park's FM demodulator in our experiments. Because we have a software implementation of the demodulation function we can use this demodulator without affecting the functionality of SDR receiver.

The source code of the Bluetooth and HiperLAN/2 transmitter and receiver is written in C and compiled with the Intel compiler 8.0 under Linux. We use floating-point precision because floating-point operations are as fast as fixed-point operations on a Pentium 4[1].

Moreover we have used the open-source FFTW library [61] for computing the inverse FFT and FFT. As a DAC requires fixed-point numbers, the transmitter has to convert the floating-point numbers into fixed point. The receiver, on the other hand, receives fixed-point numbers from the ADCs, so it has to do the inverse process. It was observed that these conversions take relatively a long time to compute and therefore special Single Input Multiple Data (SIMD) instructions [62] are used for acceleration. Of course, when the transmitter and receiver functions are rewritten to fixed-point operations, this step can be omitted.

5.3 Computational power requirements

5.3.1 User scenarios

For both standards, Bluetooth and HiperLAN/2, we have derived as a test setup, a user scenario to estimate and measure the computational requirements, assuming continuous transmission. This scenario can be compared with a realistic scenario that includes the influences of the higher OSI layers on the physical layer.

Bluetooth user scenario

The Bluetooth symbol duration is 1 μs and data is transmitted in time slots with a duration of 625 μs [18]. For estimating computational requirements, we assume maximum transfer rate. In this mode, Bluetooth uses a packet (i.e. DH5 packet) which spans 5 time slots and 1 time slot is used for uplink communication.

HiperLAN/2 user scenario

A HiperLAN/2 MAC frame has a maximum duration of 2 ms and consists of 6 logical channels (Section 3.1). In our scenario, we assume that there is

[1] We expect that floating-point instructions will have a higher power consumption than similar fixed-point instructions. However, we expect that this difference is very small compared with the total power consumption of a Pentium 4, because most power is consumed in this processor by the caches.

no Random access CHannel (RCH) and that other channels have equal duration. Furthermore, we assume that we have to demodulate two channels (one common and one user part).

5.3.2 Requirements

In the testbed we have chosen to map the demodulation (and modulation) functions of the Bluetooth and HiperLAN/2 transceiver on a Pentium 4. To compare the testbed with other implementation alternatives we have estimated the computational power requirements for a low-power DSP (TI C64x DSP) as well.

Measurement method used in the testbed

Time measurements are performed on a Pentium 4 at 2.8 GHz by counting the number of cycles for each function. A Pentium 4 processor is a very complex design and therefore the number of cycles needed for computing a particular function, is influenced by many parameters such as cache misses, memory alignment, etc. It is for that reason that we use average values in these time measurements. The number of cycles required for the whole receiver or transmitter function (total values) is measured separately and is not determined by summing up the measurement results of the individual components.

Estimation method used for the TI C64x DSP

The TI C64x DSP is a fixed point DSP family and computations can be performed with 16-bit or 32-bit precision. Simulations of our HiperLAN/2 transceiver implementation in 64-QAM mode revealed that the receiver requires at least 7-bit quantized input values for error-free reception (raw bits). So, we expect that 16-bit fixed-point calculations can be used for the receiver functions. We have estimated the computational power for each transceiver function by using available benchmarks for the TI C64x DSP [63], [64] or if there are no benchmarks available for the used algorithms, we use the instruction set manual [65]. The total computational load is estimated by the sum of all functions, multiplied with 25 % for overhead costs.

Results

Table 5.1 and Table 5.2 list for each function of the Bluetooth transmitter and receiver, the number of required Million Operations Per Second (MOPS)[2] (multiplications, additions, etc.) and how much cycles this function needs on

[2] These values are derived by looking at the used algorithms in each part of the transceiver and is not determined by the C-code. A operation can be a multiplication, addition or shift...

Bluetooth **TX function**	**MOPS**	**M cycles/s** **[Pentium 4]**	**M cycles/s** **[TI C64x DSP]**
MAC burst generation	1	1	1
GFSK modulation	2100	440	50
float-to-int conversion	80	320	-
total	**2181** (analyzed)	**714** (measured)	**64** (estimated)

Table 5.1: Computational load of Bluetooth transmitter functions.

a Pentium 4 and TI C64x DSP. Especially the GFSK modulation, conversion to fixed point numbers of the Bluetooth transmitter and FM-to-AM conversion of the receiver require most cycles. In the GFSK modulation function, a 60-tap Gaussian filter is used that requires 1000 million additions plus multiplications per second. In our implementation we have replaced this filter by lookup tables as the output value of the filter depends only on the last 4 BPSK symbols. This optimization reduces the amount of computations significantly. Furthermore, the estimated number of cycles on a TI C64x DSP is almost a factor 10 smaller than the measured number of cycles on the Pentium 4. This may have several reasons, the first one is that a DSP is designed for digital signal processing algorithms for embedded low-power systems and a Pentium 4 is not. Furthermore, several steps in our transceiver code operate on large buffers, which reduce the performance gain of the CPU cache. Moreover, the Pentium 4 implementation is written in C code without the use of optimized signal processing libraries and for estimating the load on the DSP, we use values of optimized libraries.

Table 5.3 and Table 5.4 show for HiperLAN/2, the number of required MOPS and cycles for each function of the transmitter and the receiver. Computational intensive parts are the conversion to floating-point precision, FFT and 64-QAM de-mapping in the receiver and conversion to fixed-point numbers in the transmitter. Although more bits are transmitted by the HiperLAN/2 transmitter, it requires less computational power than the Bluetooth transmitter. The HiperLAN/2 receiver requires on the other hand more cycles per second than the Bluetooth receiver, but the latter operates at a lower sample rate. Again, the estimated number of cycles on the TI C64x DSP is almost a factor 10 smaller than the measured number of cycles on the Pentium 4.

Bluetooth **RX function**	**MOPS**	**M cycles/s** **[Pentium 4]**	**M cycles/s** **[TI C64x DSP]**
int-to-float conversion	20	93	-
FM-to-AM conversion	75	176	40
synchronization	1	35	3
freq. offset corr. and bit decision	4	37	2
total	**100** (analyzed)	**381** (measured)	**56** (estimated)

Table 5.2: Computational load of Bluetooth receiver functions (Park's FM demodulator).

HiperLAN/2 **TX function**	**MOPS**	**M cycles/s** **[Pentium 4]**	**M cycles/s** **[TI C64x DSP]**
QAM mapping	38	85	15
iFFT	230	128	32
float-to-int conversion	80	215	-
total	**348** (analyzed)	**500** (measured)	**59** (estimated)

Table 5.3: Computational load of HiperLAN/2 transmitter functions for 64-QAM mode.

HiperLAN/2 **RX function**	**MOPS**	**M cycles/s** **[Pentium 4]**	**M cycles/s** **[TI C64x DSP]**
int-to-float conversion	80	351	-
synchronization and parameter estimation	4	60	8
freq. offset corr.	39	120	21
FFT	230	128	32
channel equalization	39	79	21
phase offset detection and correction	40	127	32
64-QAM de-mapping	77	204	48
total	**509** (analyzed)	**1225** (measured)	**203** (estimated)

Table 5.4: Computational load of HiperLAN/2 receiver functions for 64-QAM mode.

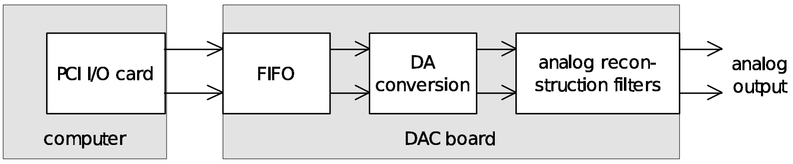


Figure 5.2: Functional architecture of the DAC board.

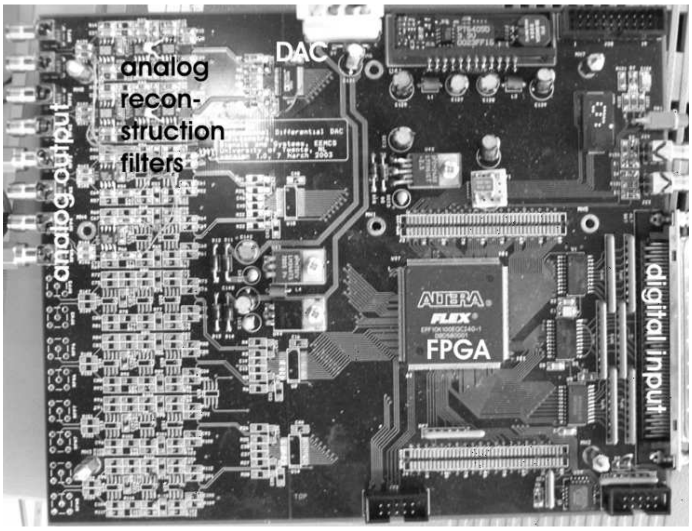


Figure 5.3: Photograph of the DAC board.

5.4 Hardware

5.4.1 DAC board

The function of the DAC board is to convert the incoming digital samples to an analog signal. It consists of an First In, First Out (FIFO) buffer, two DACs and analog reconstruction filters (Figure 5.2). The FIFO is implemented on an Field Programmable Gate Array (FPGA) together with control functions. The purpose of the FIFO is to provide a continuous stream of samples to the DAC. The DAC (Analog Devices AD9761) has a resolution of 10-bit and operates at 20 MSPS. To suppress images, the analog output of the DAC is filtered with 3^{rd} order analog reconstruction filters with a bandwith of 8.5 MHz. The DAC board is developed by G.J. Laanstra and H. Kuipers and a photograph of this board is shown in Figure 5.3.

parameter	value
Bandwidth (3 dB)	0.2 - 2.2 GHz
Noise Figure (NF)	6.5 dB
Third-Order Intermodulation Intercept Point (IIP_3)	+ 1 dBm
Power	200 mW

Table 5.5: Analog front-end characteristics (voltage supply = 1.8 V).

5.4.2 Agilent E4438C generator

The Agilent E4438C generator in Figure 5.1 provides up-conversion of the analog baseband signal. This generator is capable of generating RF signals up to 6 GHz at a large amplitude range (-136 to +7dBm) [66]. It allows proper testing of the analog and digital front-end.

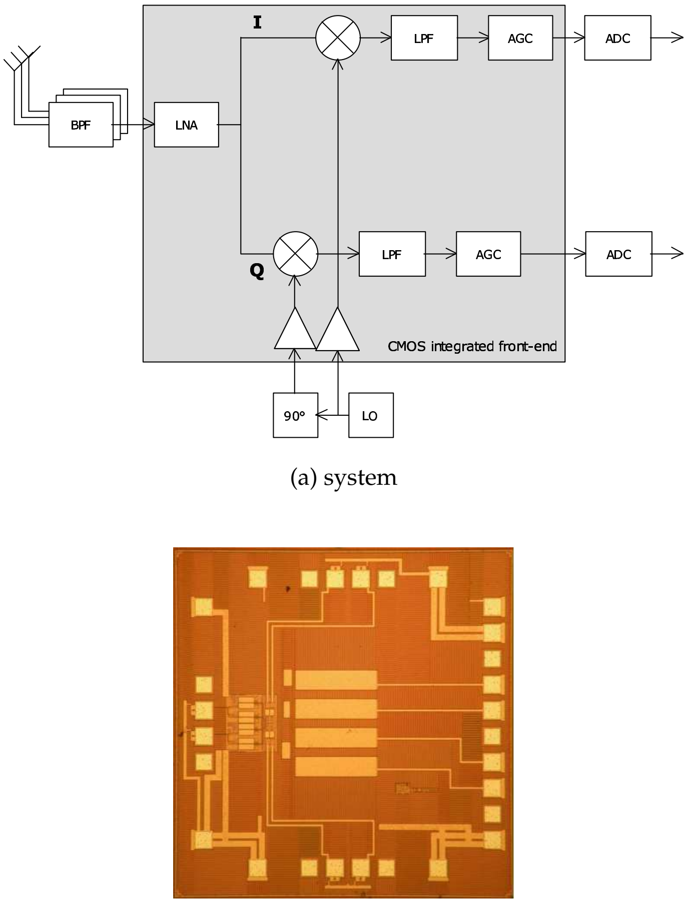
5.4.3 Analog RF front-end

This section discusses the analog part of the SDR front-end [67] . A block schematic can be seen in Figure 5.4. The front-end uses separate antennas and RF filters for the various bands [23]. The rest of the receiver, however, is integrated. In this way, the integrated circuit can still be used for a large number of applications, and only a new PCB with filters and possibly the antenna has to be designed when a receiver for a new frequency band is required. The rest of the discussion in this section will focus on the part in Figure 5.4 inside the dashed rectangle. For this part, an integrated circuit has been designed using a standard 0.18 μm CMOS process [67].

The circuit starts with a wideband Low-Noise Amplifier (LNA). This LNA employs noise canceling [68], which is useful for obtaining a low noise figure over a wide bandwidth. The LNA is followed by two mixers. Two LOs signals with a phase difference of 90 degrees are present. These implement quadrature downconversion, enabling both zero-IF and low-IF architectures.

The two mixers are both followed by a baseband amplifier combined with a low-pass filter with a cut-off frequency of around 10 MHz. This is more than enough for Bluetooth and sufficient for HiperLAN/2 signals. The circuit's correct functioning has been verified, both stand-alone and in combination with the digital front-end.

The main characteristics of the (first) prototype are shown in Table 5.5. The cut-off frequency of the analog front-end is 2.2 GHz and not 6 GHz as required by the HiperLAN/2 standard. However, the front-end still functions at higher frequencies, but the attenuation will be larger than 3 dB. As a consequence, it is expected that the required sensitivity will not be met. Power consumption of the SDR front-end is 200 mW (excluding AD conversion) whereas a dedicated front-end consumes about 50 mW. According to



(a) system

(b) chip

Figure 5.4: Analog front-end [67].

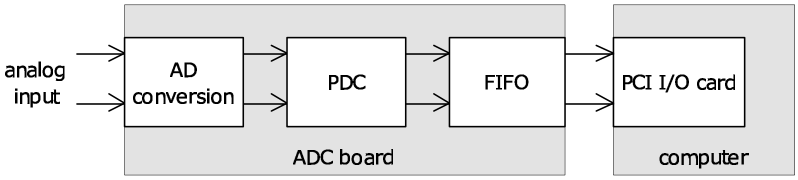


Figure 5.5: Functional architecture of the ADC board.

[51], most power is consumed in the ADCs ($>$ 80%), so power consumption of this multi-band analog front-end is large, but its influence on the total power consumption is expected to be small as the typical power consumption of a Pentium 4 is about 35 W [83]. The results of the first SDR analog front-end are promising but not good enough. Therefore, a second prototype will be designed using another topology. This new design aims at more bandwidth and less power consumption.

5.4.4 ADC board

The function of the ADC board is to convert the incoming analog baseband signal to a digital signal. It consists of two ADCs, a Programmable Down Converter (PDC), and a FIFO buffer (Figure 5.5). The analog anti-aliasing filters are provided by the analog front-end and are for this reason not integrated on this board. The ADCs (Analog Devices AD9432) have a resolution of 12-bit and operates at 80 MSPS. The outputs are connected to the Intersil ISL5416 PDC. This PDC is a programmable Applications-Specific IC (ASIC) which performs the channel selection function of the SDR receiver.

In HiperLAN/2 mode, it decimates and filters the input signal to a 20 MSPS signal and in Bluetooth mode, the PDC mixes the wanted channel to baseband and decimates the signal to a 5 MSPS signal. The PDC is a very complex and flexible chip and its functional architecture includes a mixing stage, multiple filtering stages and an AGC stage (Figure 5.6).

The output of the PDC is buffered in a FIFO and transmitted in bursts to the computer for demodulation. The ADC board is developed by G.J. Laanstra and H. Kuipers and a photograph of this board is shown in Figure 5.7.

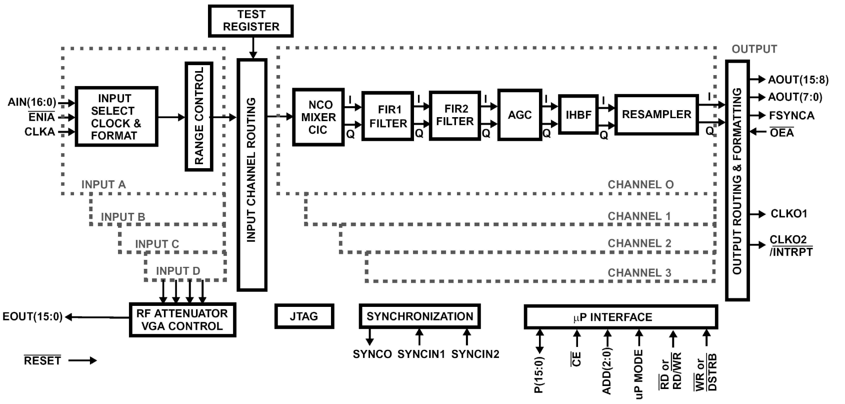


Figure 5.6: Functional architecture of the Intersil ISL5416 PDC [60].

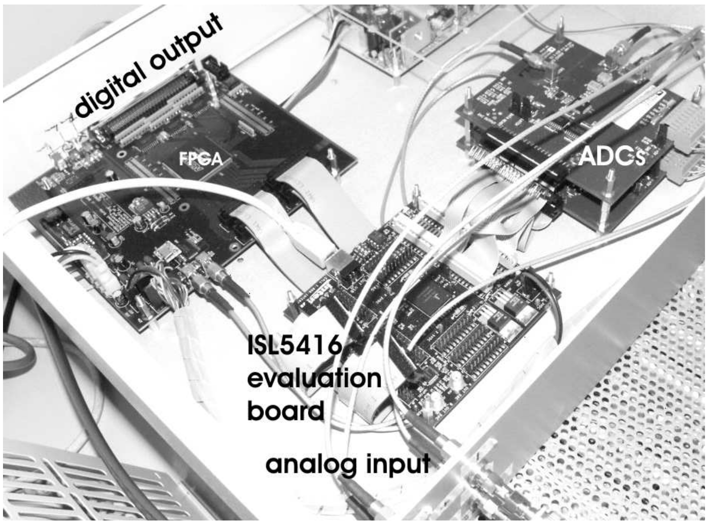


Figure 5.7: ADC board.

Chapter 6

Experiments

This chapter describes experiments which have been carried out with the SDR testbed. These experiments have two goals: *verification* of the designed and realized receiver and *demonstration* of realized real-time SDR receiver. Verification has to be performed at multiple stages of the receiver. The first stage, which is described in this chapter, is to test and verify the performance of the system. For this reason, we have built a baseband simulation model of both transceivers. This model uses the same source code of the transceivers as used in the testbed. An important goal of this simulation model is to compare the performance of both receivers for an Additive White Gaussian Noise (AWGN) channel with pertinent theoretical results. Moreover, these simulations have the purpose to fine-tune the receiver algorithms.

The second stage in the verification process is to perform baseband experiments. Goal of this step is to verify the proper functioning of the built DAC and ADC board. This is achieved qualitatively by looking at received constellations, eye-diagrams, etc. and quantitatively by transmitting pseudo random bits and calculate the BER during the transmission experiment. The last stage includes experiments with the RF part of the receiver. Goal of this step is to verify the proper functioning of the built SDR receiver. This is performed by using the same qualitative and quantitative tests of the baseband experiments.

The last part of this chapter describes an webcam application that has been developed for demonstration purposes.

6.1 Verification

6.1.1 Theoretical performance of Bluetooth

The Gaussian pre-modulation filter in the Bluetooth transmitter removes higher frequencies of the modulating signal (as can be seen in Figure 2.6). This re-

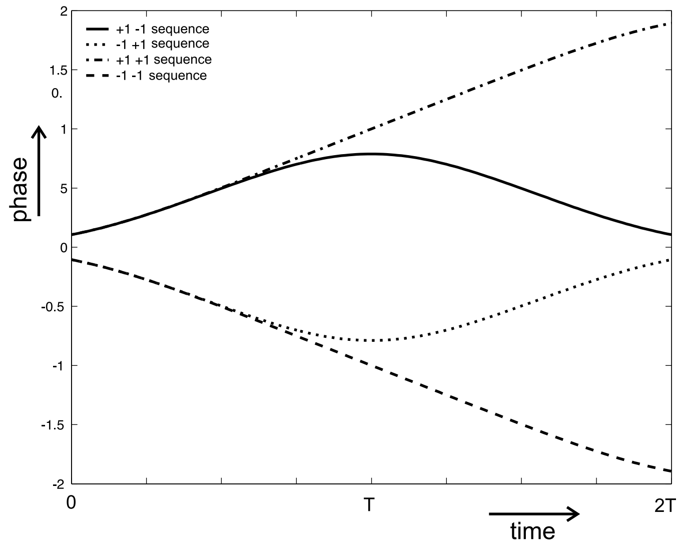


Figure 6.1: Phase plot of the transmitted sequences.

duces the bandwidth of the VCO output signal but also reduces the frequency deviations (i.e. introducing ISI) which has a negative effect on the BER.

In our literature search for GFSK demodulation we did not find a theoretical relation between the BER and SNR reported. However, most designers use 21 dB [69] in order to meet a BER of 0.1%. To verify this assumption, we have estimated the theoretical performance. The theoretical performance of phase modulated systems is determined by the Euclidean distance between the different transmitted symbol sequences. This distance is given by [70]:

$$d^2 = log_2\left(\tfrac{2}{T} \int\limits_{0}^{NT} [1 - \cos(\phi(t)]dt \right) \tag{6.1}$$

with N the observation period in bit times, $t = 0...NT$ and $\phi(t)$ the phase difference between the two phase functions of Equation 2.4.

The corresponding BER is given by [70]:

$$P_b \approx Q\left(\sqrt{d^2 \tfrac{E_b}{N_0}} \right) \tag{6.2}$$

with

$$Q(z) \triangleq \frac{1}{\sqrt{2\pi}} \int_z^{\infty} e^{-\frac{\lambda^2}{2}} d\lambda \tag{6.3}$$

Figure 6.1 depicts the phase plot of Bluetooth signals after 2 bit times for all possible sequences. Both the symbol sequence $+1, -1$ and $-1, +1$ more or less return to the original begin state. As a result of this, a receiver algorithm

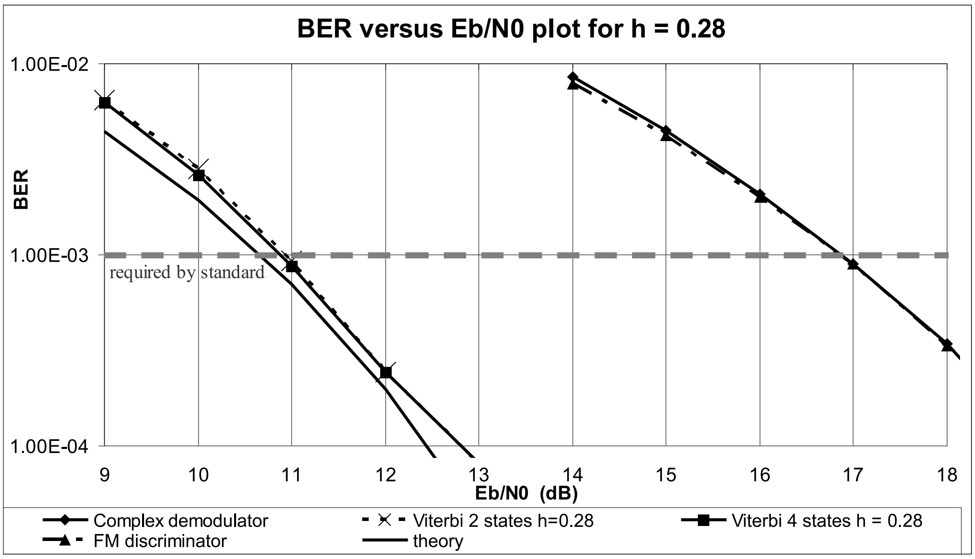


Figure 6.2: BER versus $\frac{E_b}{N_0}$ for several Bluetooth demodulators.

cannot distinguish after time n between the paths $...\alpha_{n-4}, \alpha_{n-3}, -1, +1, \alpha_n$ and $...\alpha_{n-4}, \alpha_{n-3}, +1, -1, \alpha_n$. Therefore, the observation period N should be at least 2 for Bluetooth systems which means that the receiver makes an decision for bit α_n at time $n+1$. Larger observation intervals will not increase performance.

Figure 6.1 is symmetrical around the x-axis, so in our analysis we focus only on the upper part. In this half, there are two symbol sequences, $+1, +1$ and $+1, -1$. In case of the sequence $+1, +1$, the receiver makes an error if the path $-1, +1$ or $-1, -1$ is more likely. The distances between the transmitted sequence and the paths which introduce an error, is 2.10 and 3.12 (Equation 6.1). For the other sequence, $+1, -1$, the same paths introduce an error and the according distances are 0.71 and 2.10.

The total BER is the sum of the individual BERs multiplied with the likelihood of the transmitted sequence (i.e 1/4):

$$P_b = 2\left(\frac{1}{4}Q\left(\sqrt{0.71\frac{E_b}{N_0}}\right) + \frac{1}{4}Q\left(\sqrt{2.10\frac{E_b}{N_0}}\right) + \frac{1}{4}Q\left(\sqrt{2.10\frac{E_b}{N_0}}\right) + \frac{1}{4}Q\left(\sqrt{3.12\frac{E_b}{N_0}}\right)\right) \tag{6.4}$$

Due to the non-linear nature of the Q-function, the minimum distance (i.e. 0.71) dominates the BER. The theoretical performance is depicted in Figure 6.2. To achieve a BER of 0.1%, $\frac{E_b}{N_0}$ should be at least 10.7 dB. Moreover, this figure shows the simulation results of several Bluetooth demodulators. The MAP receiver has a performance about 0.3 dB above the theoretical per-

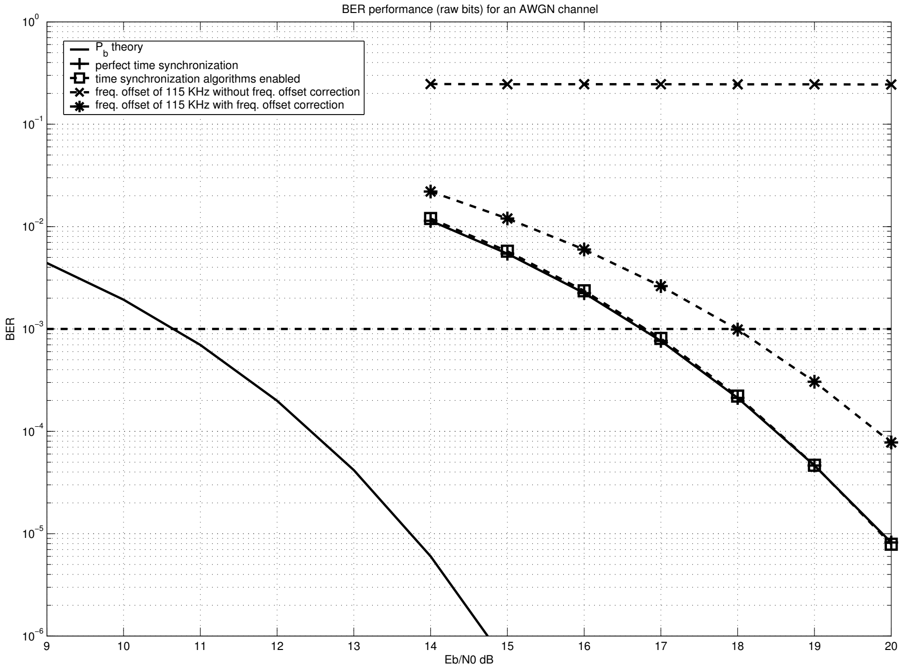


Figure 6.3: Bluetooth performance (BER versus $\frac{E_b}{N_0}$).

formance. As expected there is little difference between a MAP receiver with 2 and 4 states (i.e. observation period is 2 or 3 bit times). Finally, this figure shows also the performance of single-bit demodulators such as the Park's demodulator. Compared with the theoretical lower bound, these single-bit demodulators perform about 6 dB worse. Literature, such as [71], report the same performance for these types of demodulators.

6.1.2 Simulations

Figure 6.3 depicts the performance of the Bluetooth transceiver algorithms for an AWGN channel. For each simulation point, 10 million bits were transmitted. In the testbed we have implemented the Park's demodulator. The performance of this demodulator with perfect time synchronization is the same as in Figure 6.2. Enabling the time synchronization algorithm, does not decrease the performance of the system in the interesting area. The standard allows a maximum frequency offset of 115 kHz and simulations were also performed for an AWGN channel with this frequency offset. Without frequency offset correction, the system does not function properly as can be seen in Figure 6.3. Enabling frequency-offset correction, has a performance penalty of about 1 dB compared with a normal AWGN channel. From these simulations, it can be concluded that our implementation of the Bluetooth

Simulation	Min. $\frac{E_b}{N_0}$ [dB]	Min. SNR [dB]
P_b theory	10.7	10.7
perfect time synchronization	16.8	16.8
time synchronization algorithm enabled	16.8	16.8
freq. offset of 115 KHz without corr.	NA	NA
freq. offset of 115 KHz with corr.	18.0	18.0

Table 6.1: Minimum requirements for the Bluetooth receiver to reach a BER of 0.1%.

receiver requires a minimum $\frac{E_b}{N_0}$ of 18.0 dB to meet the requirements of the standard. The relation between SNR and $\frac{E_b}{N_0}$ is given by:

$$SNR = \frac{E_b R}{N_0 B} \tag{6.5}$$

with R the bitrate and B the bandwidth in Hz.

For analog front-end design, the required SNR is a more convenient parameter. As R equals B in Bluetooth, the minimum required SNR is also equal to 18.0 dB.

6.1.3 Theoretical performance of HiperLAN/2

In an AWGN channel, the following relation exists between $\frac{E_b}{N_0}$ and BER for a BPSK modulated system ([40], [54]):

$$P_b = Q\left(\sqrt{2\frac{E_b}{N_0}}\right) \tag{6.6}$$

with P_b the Bit-Error Rate (BER).

Compared to a *regular* BPSK system, an OFDM system uses 48 BPSK channels in parallel and the system *wastes* 1/5 of the transmitted power to the cyclic prefix. So, the total signal power P_{s_OBPSK} for an OFDM system using BPSK modulation compared to the signal power, P_{s_BPSK}, of a BPSK system is:

$$P_{s\_OBPSK} = \frac{5}{4} \cdot (48 \cdot P_{s\_BPSK} + 4 \cdot P_{s\_pilot}) \tag{6.7}$$

The four pilot carriers are modulated BPSK, with the same power as the information bearing subcarriers[1]. Thus the equation above evaluates to:

$$P_{s\_OBPSK} = 65 \cdot P_{s\_BPSK} \tag{6.8}$$

[1] This is true for all HiperLAN/2 modes.

Furthermore, the bitrate of the OFDM system is higher than of the BPSK system:

$$R_{OBPSK} = \frac{4}{5} \cdot 48 \cdot R_{BPSK} \tag{6.9}$$

Hence, the resulting expression for the average bit energy of OFDM with BPSK modulated carriers is:

$$E_{b\_OBPSK} = \frac{P_{s\_OBPSK}}{R_{OBPSK}} = \frac{65 \cdot P_{s\_BPSK}}{48 \cdot \frac{4}{5} \cdot R_{BPSK}} \approx 1.69 \cdot E_{b\_BPSK} \tag{6.10}$$

Due to the cyclic prefix, the noise power is also scaled by 5/4. Thus, the relation between normal BPSK modulation and HiperLAN/2 using BPSK modulated carriers is:

$$\frac{E_{b\_OBPSK}}{N_{0\_OBPSK}} = \frac{65}{48 \cdot \frac{4}{5} \cdot \frac{5}{4}} \cdot \frac{E_{b\_BPSK}}{N_{0\_BPSK}} = \frac{65}{48} \cdot \frac{E_{b\_BPSK}}{N_{0\_BPSK}} \tag{6.11}$$

From this, it can be concluded that the OFDM system using BPSK as subcarrier modulation technique, needs an extra ≈ 1.3 dB to reach the same raw bit error rate as a *regular* BPSK system.

Thus, the theoretical BER as function of $\frac{E_b}{N_0}$ for a HiperLAN/2 BPSK system is given by:

$$P_{b\_OBPSK} = Q\left(\sqrt{2 \cdot \tfrac{48}{65} \cdot \tfrac{E_b}{N_0}}\right) \tag{6.12}$$

This computation is valid for all other subcarrier mapping techniques, because the pilot carrier energy is kept equal to the average subcarrier energy.

The theoretical BER as function of $\frac{E_b}{N_0}$ for a HiperLAN/2 Quadrature Phase-Shift Keying (QPSK) system is equal to the performance for BPSK modulation:

$$P_{b\_OQPSK} = Q\left(\sqrt{2 \cdot \tfrac{48}{65} \cdot \tfrac{E_b}{N_0}}\right) \tag{6.13}$$

The theoretical BER as function of $\frac{E_b}{N_0}$ for a HiperLAN/2 16-QAM system is [40]:

$$P_{b\_O16QAM} = 4(1 - \frac{1}{\sqrt{16}})Q\left(\sqrt{\frac{48}{65} \cdot \frac{12 \cdot E_b}{(16-1)N_0}}\right) \tag{6.14}$$

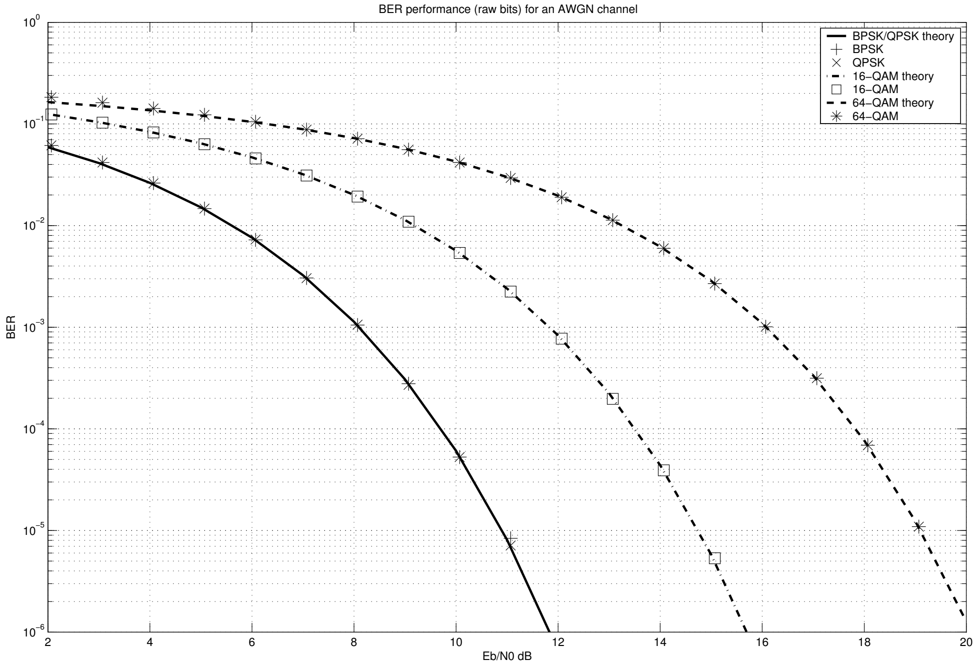


Figure 6.4: HiperLAN/2 raw BER performance (BER versus $\frac{E_b}{N_0}$).

And the performance of a HiperLAN/2 64-QAM system is:

$$P_{b\_O64QAM} = 4(1 - \frac{1}{\sqrt{64}})Q\left(\sqrt{\frac{48}{65} \cdot \frac{18 \cdot E_b}{(64-1)N_0}}\right) \tag{6.15}$$

6.1.4 Simulations

Figure 6.4 depicts the simulated raw BER for the HiperLAN/2 system. The channel is AWGN and in these simulations all synchronization and parameter estimations algorithms are disabled. The simulation results match the theoretical performance (derived in the previous section). Therefore, we can assume that we have implemented our simulation model correctly.

Packet-error ratio

The HiperLAN/2 standard requires a Packet-Error Rate (PER) of less than 10% for packets with a size of 54 bytes [17]. A packet is received incorrectly, if one or more bits after error correction are wrong. The required BER can be

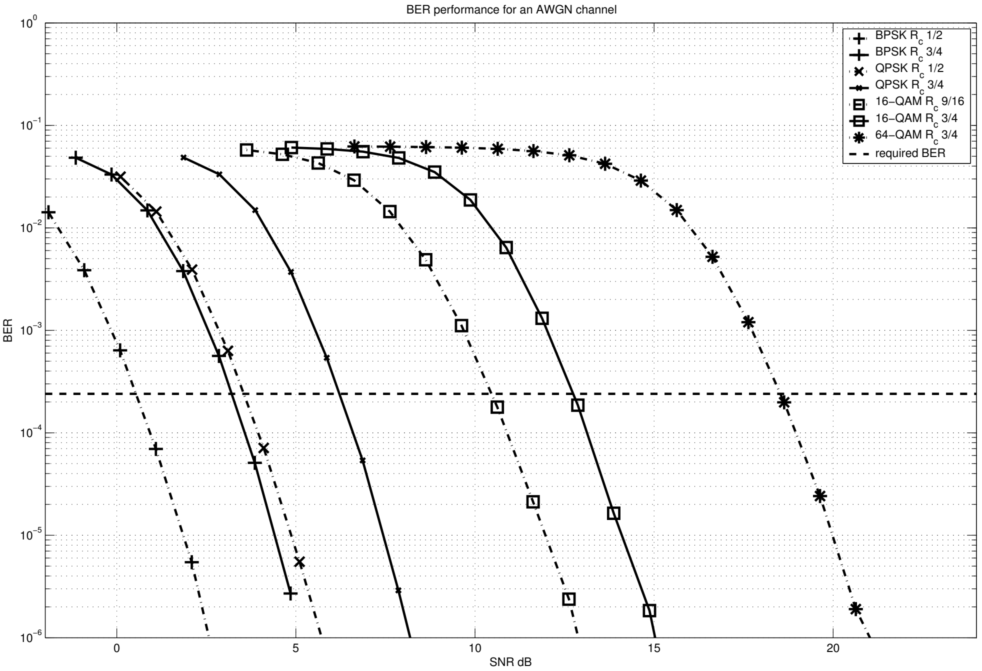


Figure 6.5: HiperLAN/2 BER performance (BER versus SNR).

calculated from the PER as follows (see [72]):

$$P_b = 1 - \left(1 - P_e\right)^{\frac{1}{54 \cdot 8}} \tag{6.16}$$

with P_b the BER and P_e the PER.

Figure 6.5 depicts the BER after error correction and Table 6.2 summarizes the minimum requirements for the different bitrate modes of the HiperLAN/2 system.

In the second experiment, the channel is replaced by an AWGN channel plus a frequency offset of 240 kHz. In addition, all receiver algorithms are enabled; time synchronization, frequency-offset estimation & correction, channel transfer function estimation & correction and common-phase-offset estimation & correction. Especially, the value of the threshold parameter in the time synchronization algorithm is a critical parameter. Its optimal value depends on the noise level present in the system. At a high noise level, the maximum correlation peak of the time synchronization algorithm is much lower than for a low noise level. In our simulations, we have optimized this value for the interesting $\frac{E_b}{N_0}$ area.

Figure 6.6 depicts for this channel, the performance of the system after error correction. See also Table 6.3. The channel AWGN, is a perfect channel,

Bitrate mode	**Subcarrier modulation**	R_c	**Minimal** $\frac{E_b}{N_0}$ **[dB]**	**Minimal SNR [dB]**
A	BPSK	1/2	5.7	0.5
B	BPSK	3/4	6.7	3.2
C	QPSK	1/2	5.7	3.5
D	QPSK	3/4	6.7	6.2
E	16QAM	9/16	9.2	10.5
F	16QAM	3/4	10.1	12.7
G	64QAM	3/4	14.1	18.5

Table 6.2: Minimum $\frac{E_b}{N_0}$ requirements for an AWGN channel to reach a PER of 10% using packet length of 54 bytes.

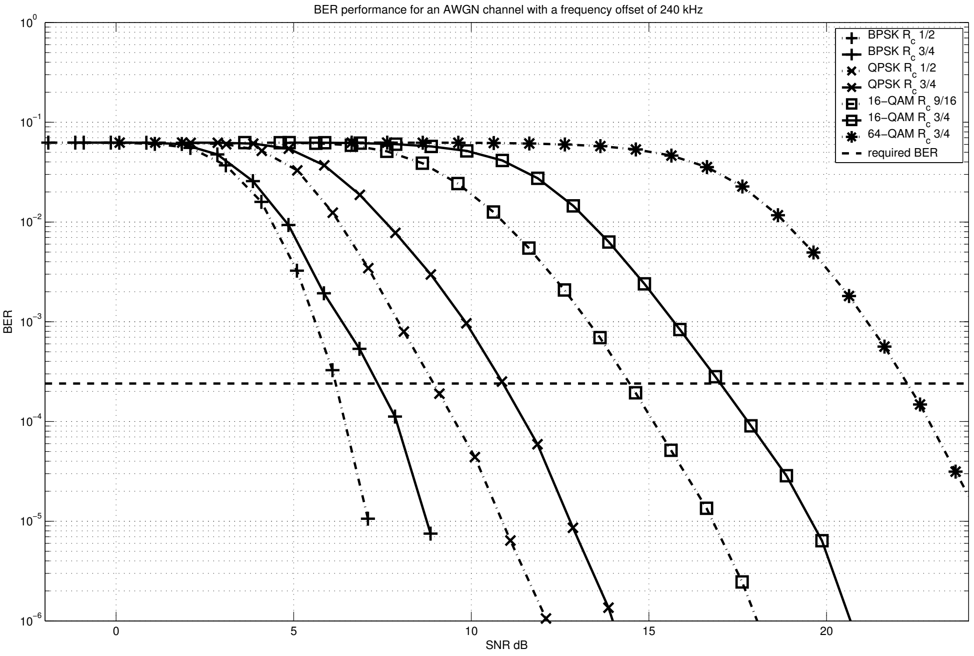


Figure 6.6: HiperLAN/2 BER performance for an AWGN channel plus a frequency offset of 240 kHz (BER versus SNR).

Bitrate mode	**Subcarrier modulation**	R_c	**Minimal** $\frac{E_b}{N_0}$ **[dB]**	**Minimal SNR [dB]**
A	BPSK	1/2	11.4	6.2
B	BPSK	3/4	10.9	7.4
C	QPSK	1/2	11.1	8.9
D	QPSK	3/4	11.4	10.9
E	16QAM	9/16	13.2	14.5
F	16QAM	3/4	14.4	17.0
G	64QAM	3/4	17.9	22.3

Table 6.3: Minimum $\frac{E_b}{N_0}$ requirements for an AWGN channel with a frequency offset of 240 KHz to reach a PER of 10% using packet length of 54 bytes.

so there is no need to estimate the channel transfer function, frequency offset and common phase offset. In this experiments, however, we have enabled the estimation algorithms for these parameters, to investigate its influence on the system performance. For higher bitrate modes, the difference is smaller than for lower modes because the preamble section is scaled with the used modulation. This means that for lower modes, the preamble section is more affected by noise than for higher modes. As a consequence, parameter estimation will be less accurate. This can be seen in Figure 6.6.

In a more realistic experiment, a multi-path channel should be used. The ETSI defines several multi-path channel, ranging from typical office environments to large indoor areas such as a exhibition hall [73], [46]. We did not implement these channels in our simulation model mainly due to time constraints. Moreover, these channels are not defined at 20 MSPS but at 100 MSPS. So, in order to simulate these channels, we have to extend our baseband simulation model to this higher sample rate and include channel selection filters. Another option is to down-sample the channel model to 20 MSPS. In [74] simulation results are shown for the first channel model of the ETSI, channel model A. Because, it is unclear which algorithms are used in the receiver and how the multi-path channel is implemented we cannot use these results to estimate the performance penalty in our receiver if these multi-path models are used. So, implementation of a multi-path channel requires time to implement which we did not have and the result cannot easily be compared with values from literature. As the most important goal of the simulations in this section is to verify the designed and realized transceiver, implementation of multi-path channels is not required.

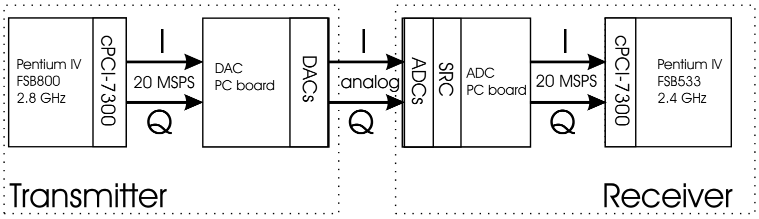


Figure 6.7: Setup for baseband experiments.

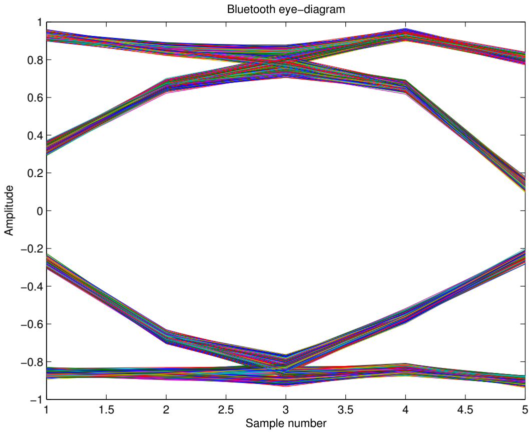


Figure 6.8: Baseband experiments: Eye-diagram of received Bluetooth signal.

6.1.5 Baseband experiments

In baseband experiments, the output of the DAC board is connected directly to the input of the ADC board (Figure 6.7). In these experiments, there is a negligible frequency offset[2]. So to obtain optimal results, we have disabled the frequency-offset detection and correction algorithms in both receivers. In Bluetooth mode, the transmitter generates Bluetooth signals at 4 MHz because the analog circuit before the ADCs has a high-pass filter characteristic. (The cut-off frequency is 400 kHz [75].) After digitalization of the Bluetooth signal, the Intersil PDC mixes the signal back to baseband and applies a low-pass filter with a passband of 0.5 MHz. In these experiments, the receiver does not produce bit errors and the eye-diagram of the received signal is very good (Figure 6.8).

In HiperLAN/2 mode, the transmitter generates HiperLAN/2 signals at baseband. The receiver samples this signal and the Intersil PDC applies a low pass filter with a cut-off frequency of 10 MHz. In BPSK and QPSK mode, the receiver works without bit-errors. However, in 16-QAM and 64-QAM mode

[2] The LOs operate at 80 MHz and not 6 GHz. For this reason, the frequency difference between both LOs can be neglected.

bit-errors occur at carrier -1 and carrier 1. Figure 6.9 depicts the constellation diagrams of several carriers in 64-QAM mode at point Q in Figure 3.10. Carrier 1 is very noisy whereas carrier 5 has a perfect constellation. The same applies for carrier 14 and 26.

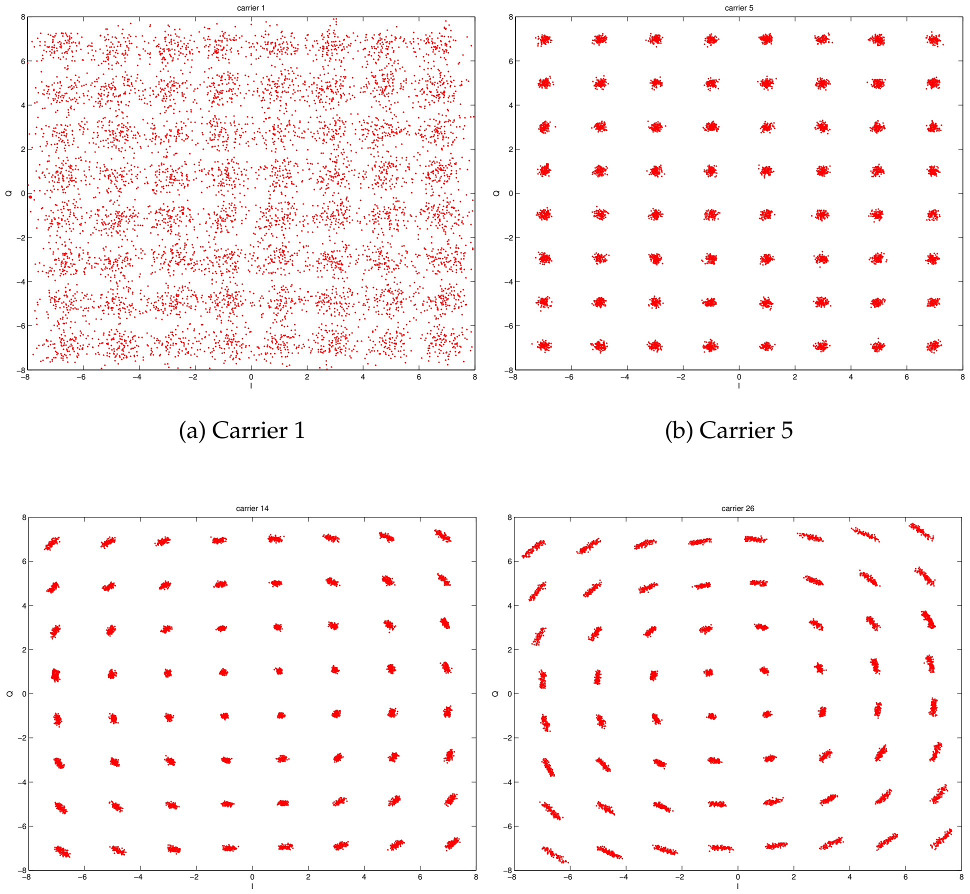
Why carrier 1 and -1 are distorted is unknown. If all carriers are set to zero, except carrier 1, the constellation of carrier 1 is much better (Figure 6.10). Additional testing is needed, to find the cause of this distortion, but experiments indicate that this distortion is introduced in the ADC board. Furthermore, the analog circuit before the ADCs has a high-pass filter characteristic and its cut-off frequency is 400 kHz [75] whereas the carrier frequency of carrier 1 is 312.5 kHz. However, measurements in the analog circuit showed only a 0.1 dB attenuation at 312.5 kHz. Probably there are other reasons for this cut-off frequency and replacing the analog circuit may solve the problem. The constellation diagrams of carrier 14 and 26 have a small phase offset which is probably introduced by an I/Q imbalance. See Section 7.1.1 for more information.

Moreover, the 64-QAM constellation of carrier 26 seem to rotate about 7 degrees during the transmission of a burst. This effect is also present in the constellation for carrier 14, but less prominent. We assume that phase noise of the LOs introduces this distortion [49] which is discussed in more detail in Section 7.1.1. Finally, Figure 6.11 shows the output of the channel estimation algorithm. For each carrier, the length of the vertical line indicates the variation in channel estimation. In baseband experiments, the channel is mainly determined by the analog reconstruction filters in the DAC board. These filters are designed to have a maximum passband ripple of 0.5 dB. This is verified by the output of the channel estimation algorithm which is very flat.

6.1.6 RF experiments

RF experiments have been performed with the setup of Figure 6.12 (without antennas) and a picture of the testbed is shown in Figure 6.13. Because the used LOs are very accurate, the frequency offset can be neglected. Therefore we have disabled the frequency-offset detection and correction algorithms in both receivers to achieve optimal results. Furthermore, the setup of Figure 6.12 is not complete because fully functional receivers also contain an analog AGC. In our setup, only the AGC of the PDC in the digital domain is enabled, so weak signals will not use the full scale of the ADCs. For this reason, the sensitivity requirements of both standards cannot be met. More research is required to implement this feature because it needs to know for example, the signal strength at the input of the ADCs.

In Bluetooth mode, the receiver operates from -20.0 to -69.4 dBm signal



(a) Carrier 1

(b) Carrier 5

(c) Carrier 14

(d) Carrier 26

Figure 6.9: Baseband experiments: 64-QAM constellation diagrams for several carriers.

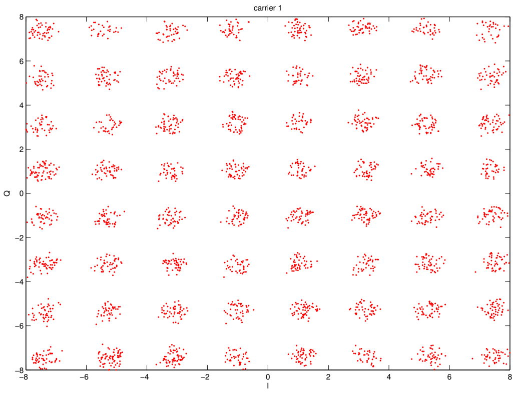


Figure 6.10: Baseband experiments: 64-QAM constellation diagram for carrier 1 with other carriers set to zero.

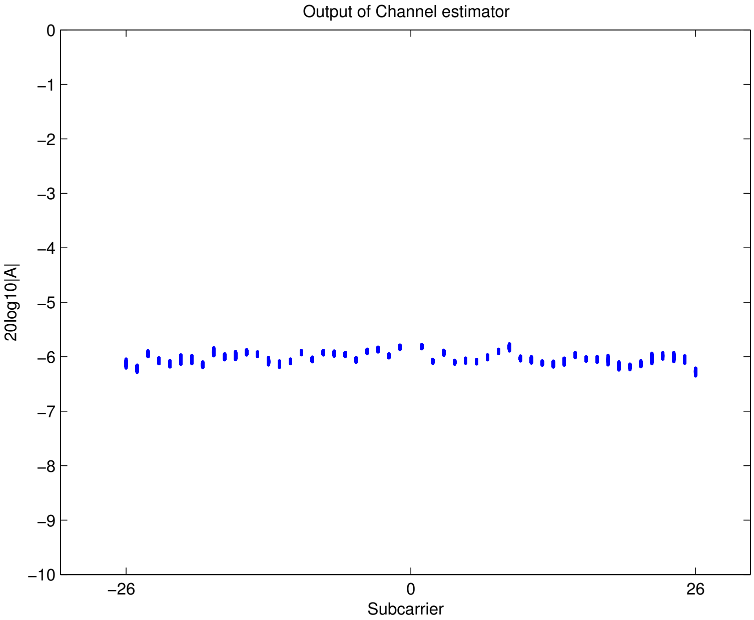


Figure 6.11: Baseband experiments: Output of the HiperLAN/2 channel estimation algorithm.

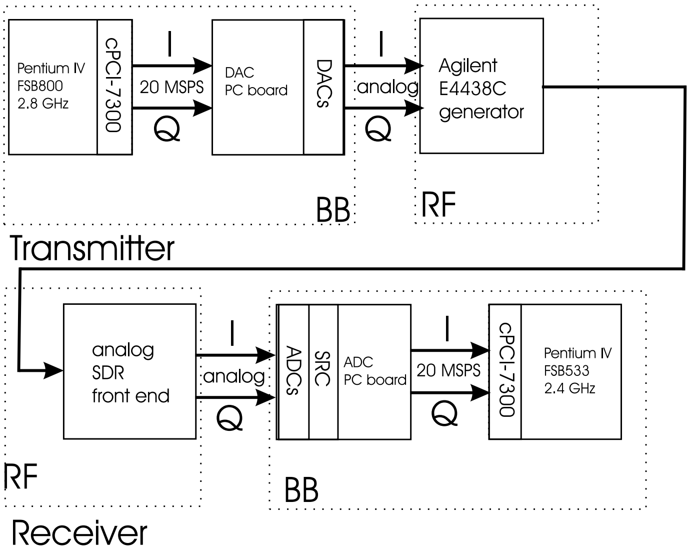


Figure 6.12: Setup for RF experiments.

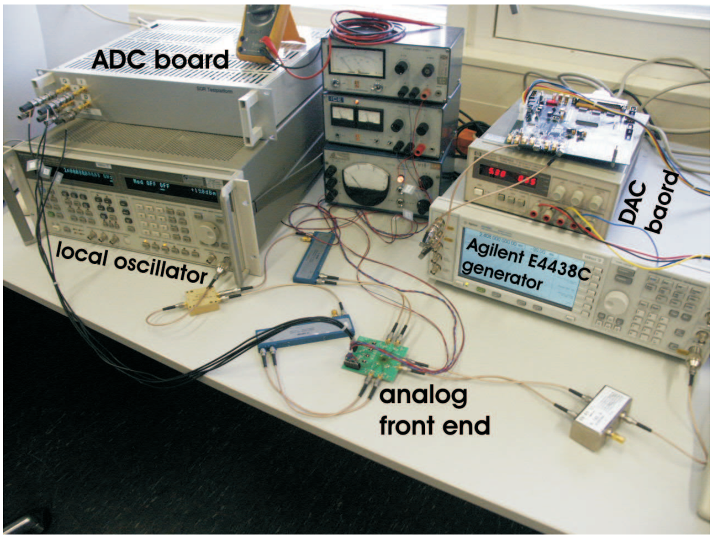
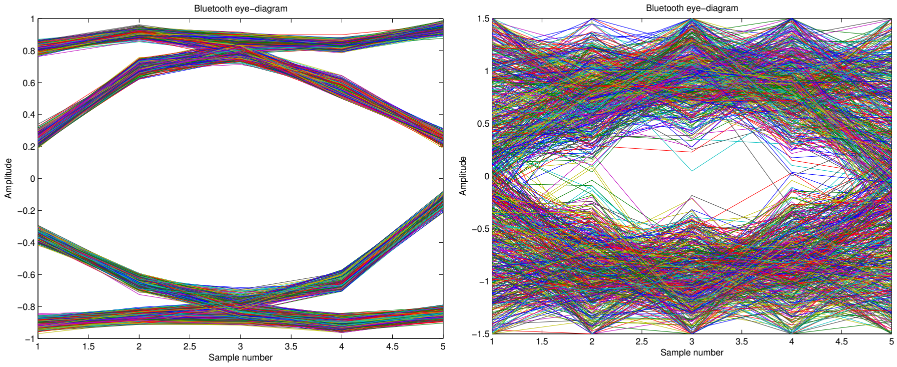


Figure 6.13: Photograph from the setup used in RF experiments (both computers are not shown).



(a) -20.0 dBm

(b) -69.4 dBm

Figure 6.14: RF experiments: Eye-diagram of received Bluetooth signal.

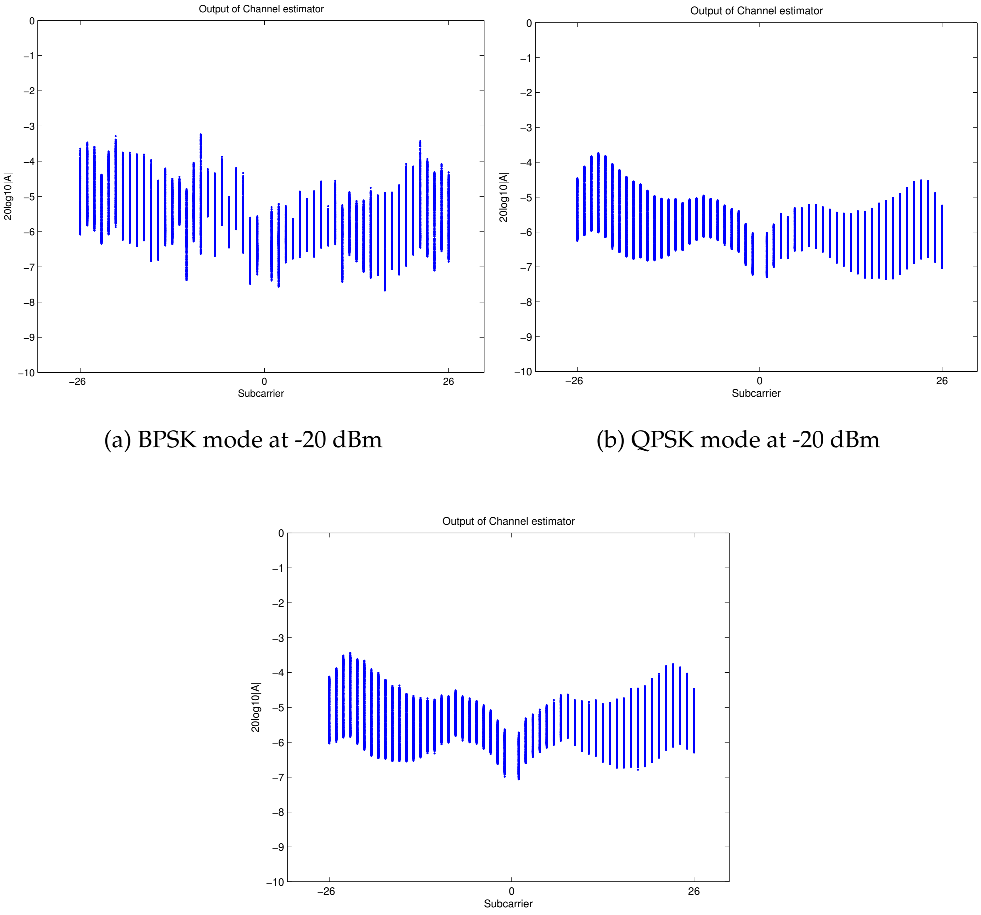
strength[3] within the specifications of the standards i.e. the BER is below 0.1%. Figure 6.14 depicts the eye-diagram for both −20.0 and −69.4 dBm Bluetooth signal. The first eye-diagram is very open whereas with the weak signal at −69.4 dBm, the eye-diagram is almost closed.

In HiperLAN/2 mode, the receiver works only for BPSK, QPSK and 16-QAM mode. In 64-QAM mode, the Packet-Error Rate (PER) is larger than 10%. Figure 6.15 shows the output of the channel estimation algorithm for the three lower modes of HiperLAN/2. The variation of the estimations are much higher than in Figure 6.11. So, it seems that the output of the SDR analog front-end introduces more noise in the wanted signal bandwidth. Moreover, the variations of the outer carriers is higher than for lower carriers.

Figure 6.16 depicts for the constellation of several carriers in 16-QAM mode at point Q of Figure 3.10. The same carriers are plotted as in Figure 6.9, however the used modulation mode is different! It seems that for all carriers, the constellation points at the border of the constellation diagrams are more noisy than the inner points. We do not know which distortion can introduce this effect. Another observation is that carrier -1 is not significantly worse than other carriers. This means that other noise sources are much stronger than the effect observed in the baseband experiments. Additional research is needed to analyze the above mentioned effects. More information can be found in the Ph.D. thesis of V.J. Arkesteijn [47].

Moreover, we have also conducted sensitivity tests for both standards, see Table 6.4. In Bluetooth mode, the specification of the standard are almost met

[3] We have calibrated the output of the Agilent generator by manually measuring the RMS value of the generator output.



(a) BPSK mode at -20 dBm

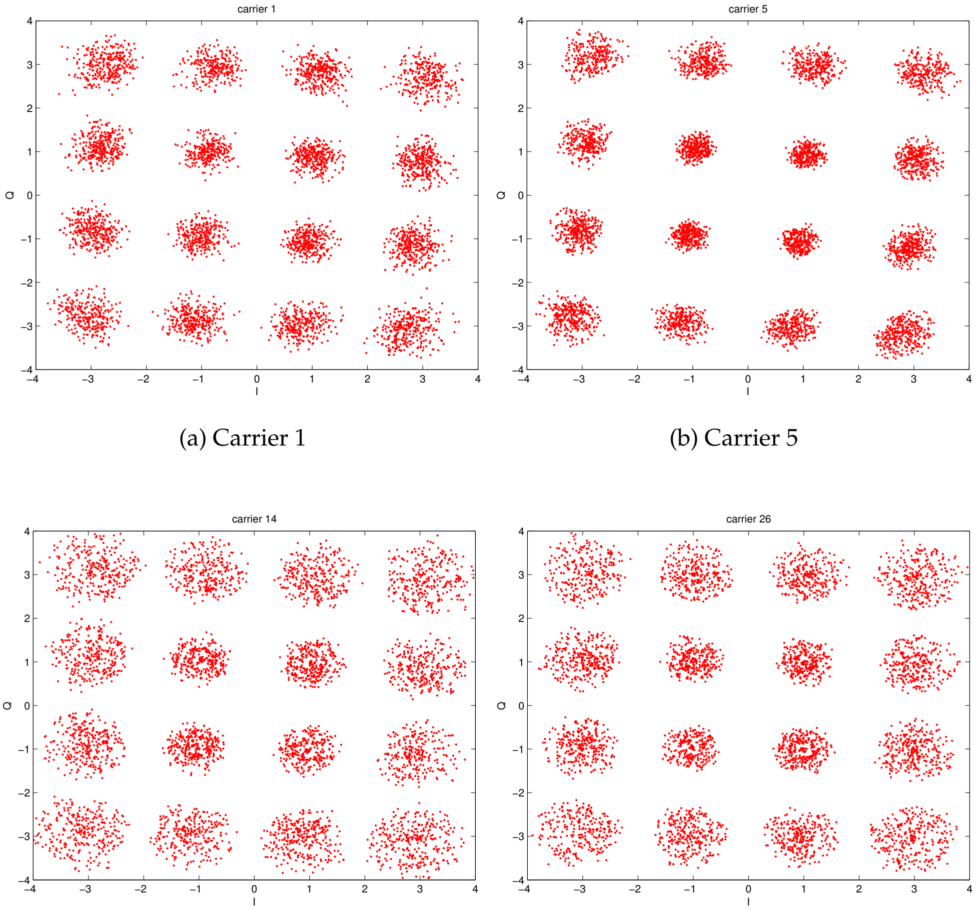
(b) QPSK mode at -20 dBm

(c) 16-QAM mode at -40 dBm

Figure 6.15: RF experiments: Output of the HiperLAN/2 channel estimation algorithm.

Mode	**Required sensitivity**	**Measured sensitivity**
Bluetooth	−70 dBm	−69.4 dBm
HiperLAN/2 BPSK	−83 dBm	−71.0 dBm
HiperLAN/2 QPSK	−79 dBm	−66.0 dBm
HiperLAN/2 16-QAM	−73 dBm	−53.8 dBm
HiperLAN/2 64-QAM	−68 dBm	NA

Table 6.4: Sensitivity tests



(a) Carrier 1

(b) Carrier 5

(c) Carrier 14

(d) Carrier 26

Figure 6.16: RF experiments: 16-QAM constellation diagrams for several carriers at -40 dBm signal strength.

and in HiperLAN/2 mode there is a large gap between requirements[4] and the performance of our SDR receiver. It is expected that implementation of an analog AGC in the testbed setup will improve these results dramatically. However, the design of an AGC is not easy, because it is a feedback system in both the analog and digital domain.

6.2 Applications

6.2.1 Webcam application

To demonstrate our SDR receiver, we have developed a application which demonstrates our SDR receiver. In this application images are grabbed real-time from a webcam. Then, the pictures of the webcam are scaled down, depending on the used standard and mode. After down-scaling of the picture the data is modulated (without error correction) with Bluetooth or HiperLAN/2 modulation, up-converted to RF frequencies and transmitted through air (Figure 6.17). At the receiver side, the signal is picked up with an antenna that is connected to the SDR front-end and the raw bits, i.e. video signal, is shown on screen. Because both the transmitter and receiver program use 2 buffers (ping/pong), the total latency of the system is about 400 ms. In the application, the user can select at run-time the standard and used mode. Currently no experiments have been carried out with the webcam application, but initial experiments have shown that we can receive Bluetooth signals successfully at a distance of 30 cm.

[4] The maximum PER is 10% in HiperLAN/2 which equals a raw BER of about 10^{-3}. This value is an approximation because it depends strongly on the nature of the raw bit-errors.

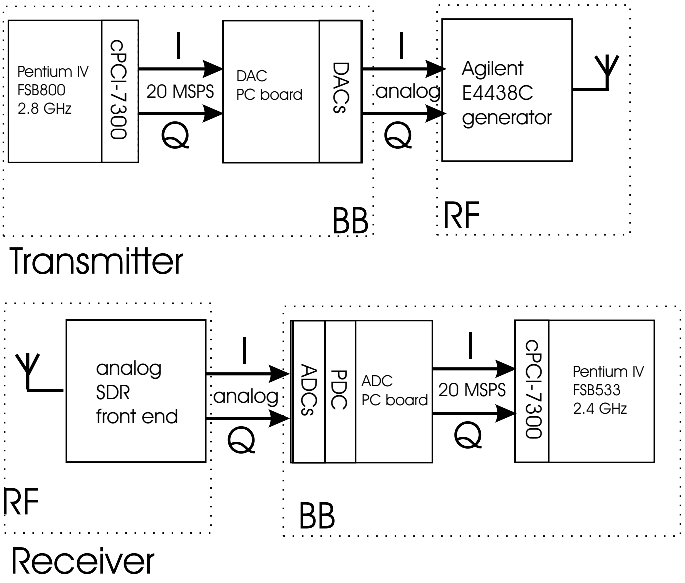


Figure 6.17: Setup for the webcam application.

Chapter 7

Discussion

The final chapter of this thesis consists of three parts. First, the designed and realized test-bed is evaluated. We consider questions as for example, is our receiver capable of decoding HiperLAN/2 and Bluetooth MAC bursts when there is a DC offset in the output of the analog front-end? Moreover, an overview is given of major sources of distortions in the analog front-end. The second part of this chapter discusses the utilization of the result of our project. Can our SDR receiver be used in commercial products and moreover can it be extended to other standards? Finally, conclusions are drawn and recommendations are given for further research.

7.1 Retrospection

At the start of the project, we knew little about Bluetooth and HiperLAN/2 receivers. In order to gain knowledge, we have built a testbed which is capable of transmitting and receiving Bluetooth and HiperLAN/2 signals real-time. Now, at the end of the project we evaluate this testbed. Does it meet the specifications set by both standards? Chapter 6 has shown that this is not entirely the case. To find the cause for this, the main distortions in the analog front-end have to be identified. In addition, their influence on both demodulators have to be investigated. Our purpose is to solve these distortions in the digital domain if it is necessary to do so.

7.1.1 Mixed signal design

Bluetooth is designed for low-cost and robustness against interferers. For these reasons, it allows parameters in the radio system to have relatively large variations. So, the requirements on the analog front-end are less severe than for the high-speed HiperLAN/2 standard. Therefore, in the rest of this section, only the influence of analog distortions is investigated for the

HiperLAN/2 receiver. Four distortions in the analog front-end are discussed: DC offset, phase noise, I/Q mismatch and non-linearities.

DC offset

In our project we use a zero-IF receiver architecture and one of the disadvantages of this architecture is DC-offset (Section 4.3). An DC offset introduces several effects in the HiperLAN/2 receiver. The first effect is corruption of the time synchronization algorithm. The implemented algorithm is correlation-based and an DC offset biases the output values of the correlation. Moreover, the estimation of the frequency offset is affected as it estimates the average phase difference between two parts of the preamble structure. If there is a DC offset, the average phase difference will be biased. In addition, an DC offset in combination with a frequency offset, leaks into neighboring carriers.

Assume that there is a difference between the carrier frequency in the transmitter and the receiver of δf[1]. Moreover, assume that the output of the analog front-end contains a DC offset. Let the received signal be defined as (Equation 3.16):

$$\widetilde{r}[k] = \widetilde{s}[k]e^{\jmath(2\pi \frac{k}{T}\delta f)} + A_{DC} \tag{7.1}$$

with $\widetilde{s}[k]$ the transmitted complex baseband signal, δf the frequency offset, T the symbol time and A_{DC} the DC offset.

The received signal is corrected in the receiver by multiplying with the negative frequency of the estimated frequency offset ($\widehat{\delta f}$):

$$\widetilde{r}_c[k] = \widetilde{r}[k]e^{-\jmath(2\pi \frac{k}{T}\widehat{\delta f})} + A_{DC}e^{-\jmath(2\pi \frac{k}{T}\widehat{\delta f})} \approx \widetilde{s}[k] + A_{DC}e^{-\jmath(2\pi \frac{k}{T}\widehat{\delta f}} \tag{7.2}$$

So, a DC offset is combination with a frequency offset is converted to a frequency equal to the negative value of the frequency offset. In the next step of the receiver, an FFT is applied:

$$\begin{aligned} \widehat{C}_{m,n} &= \textstyle\sum_{x=0}^{N-1} (\widetilde{r}_c[x] + A_{DC}e^{-\jmath(2\pi \frac{k}{T}\widehat{\delta f}})e^{-\jmath 2\pi \frac{xm}{N}} \\ &= N \cdot DFT(\widetilde{r}_c[x]) + N \cdot DFT(A_{DC}e^{-\jmath(2\pi \frac{k}{T}\widehat{\delta f}}) \end{aligned} \tag{7.3}$$

Because the DC offset is not orthogonal to the data carriers, part of it will leak into neighboring carriers (last term of Equation 7.3). Figure 7.1 depicts this effect for an frequency offset of 240 kHz.

In Figure 7.2, the power of the DC offset that spills into a neighboring subcarrier in the direction of the frequency offset is plotted. This is calculated with Equation 7.3 (see also [76]).

[1] Note that this frequency difference is *not* time dependent. Any time dependent changes are represented as *phase noise*.

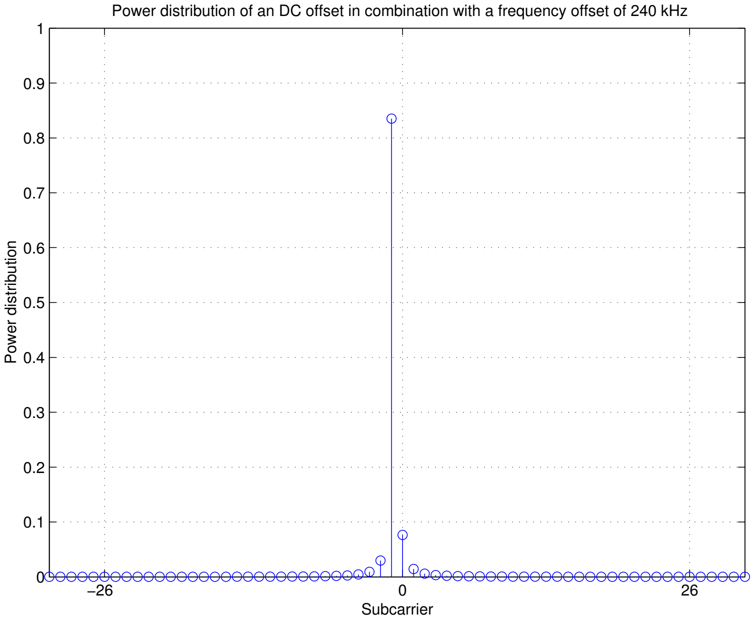


Figure 7.1: Power distribution of a DC offset in combination with a frequency offset of 240 kHz on the subcarriers.

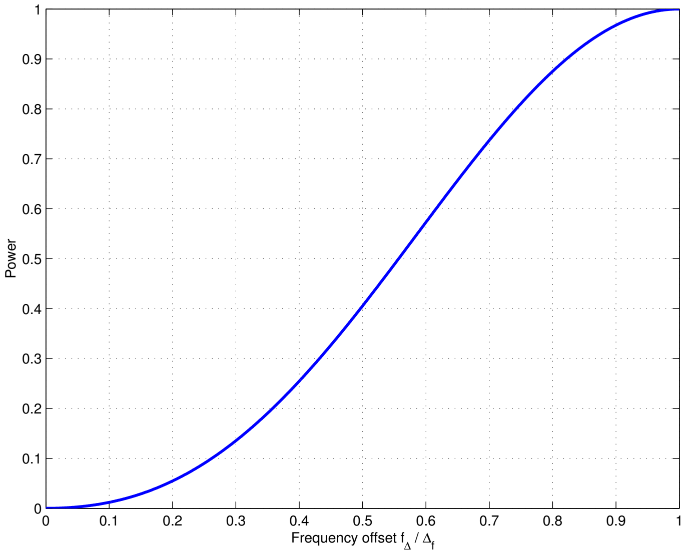


Figure 7.2: Influence of a DC offset in combination with a frequency offset on neighboring subcarrier in direction of the frequency offset.

In a worst case scenario with an frequency offset of 240 kHz, approximately 83% of the DC offset power is spilled into carrier -1. The highest mode in HiperLAN/2 (64-QAM) requires a SNR of 21.1 dB (Table 6.3) in order to have a PER of 10%. An DC offset should be much lower than this noise floor because it affects both the frequency offset estimation algorithm and spills into neighboring subcarriers:

$$C/I_{DC} = \frac{E_b \cdot R \cdot \frac{5}{4}}{0.83 P_{DC}} \ll SNR_{required} \tag{7.4}$$

with E_b the bit energy (without prefix), R the bitrate, P_{DC} the power of the DC component and $SNR_{required}$ the required SNR to meet a PER of 10%.

Simulations for HiperLAN/2 with the highest mode, 64-QAM, have been carried out with a frequency offset of 240 kHz and different levels of DC offset. See Figure 7.3. If the DC offset is 34 dB lower than the signal energy, its influence can be neglected. The maximum allowable DC component entering the digital part of the receiver is about 10% of the ADC resolution[2].

Moreover, the HiperLAN/2 receiver requires at least 7 bit for error free reception (Section 5.3.2). In our testbed we used 12-bit ADCs, so the DC component is in a worst case scenario $10\% \cdot 2^{12}/2^5 \approx 3$ ($=$ 10 dB) times larger than the HiperLAN/2 signal. Thus, a DC-offset has to be suppressed by 44 dB (10 + 34) in the digital part. At this moment, the demonstrator software does not contain DC-offset cancellation algorithms, but we expect that this suppression can be met by using DC-offset cancellation at multiple stages in the digital part of the receiver, if this is necessary.

For example, the DC can be estimated at the signals entering the digital domain. In addition, it can be estimated more precise after time synchronization is achieved and before frequency offset is estimated[3]. Because the analog baseband signal can have a maximum frequency offset of 240 kHz and the first carrier has a value of 312.5 kHz. it is not possible to use a high pass filter just before AD conversion.

Phase noise

Phase noise degrades the performance of the system by two effects: loss of orthogonality of the subcarriers (similar to a *time dependent* frequency offset) and a *common phase offset* to all subcarriers [37]. The latter effect, the common phase rotation, is already detected and corrected in the receiver (point P in Figure 3.10, page 52.).

To describe these two effects independently, we will divided the phase of the mixer compared to the mixer in the transmitter into two parts: a part

[2] This value has been set in one of the project meetings with the motivation that an DC offset should not increase the resolution requirements of the ADCs too much.

[3] The preamble structure of HiperLAN/2 does not contain a DC component.

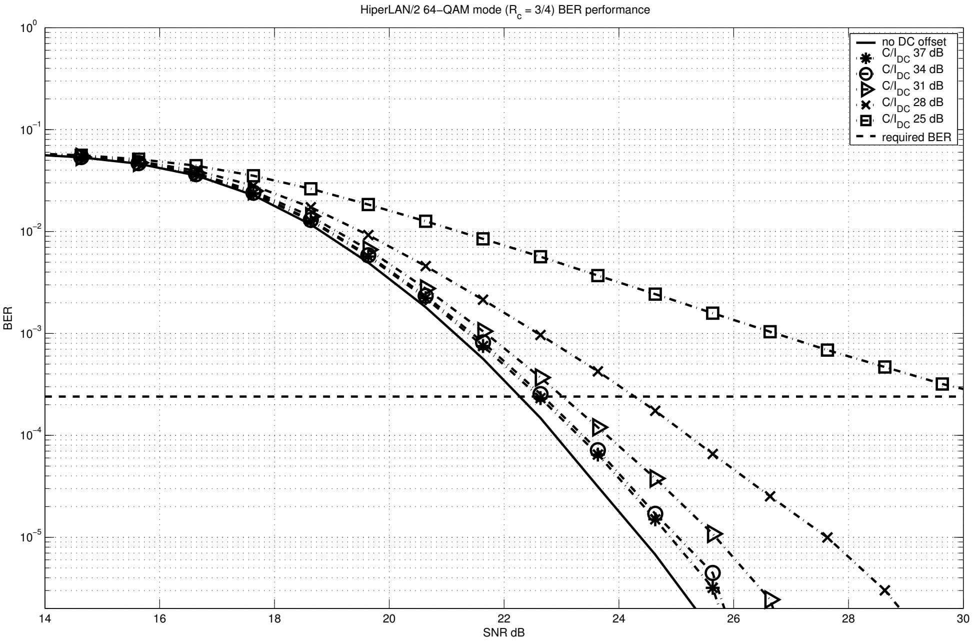


Figure 7.3: HiperLAN/2 BER performance with an frequency offset of 240 kHz and a DC offset (BER versus SNR).

that remains equal throughout an OFDM symbol, ϕ_0 and a time varying part, $\phi(t)$.

The time dependent phase offset, also called *phase noise* or *phase jitter* depends on the actual implementation of the VCO. This effect is described in [76] by a Gaussian random process with zero mean and variation ${\sigma_{\phi(t)}}^2$.

A time-dependent phase offset in the system can be written as follows:

$$\widetilde{r}[i] = \widetilde{s}[i]\, e^{\jmath\, \phi[i]} \tag{7.5}$$

with $\widetilde{r}[i]$ the received signal, $\widetilde{s}[i]$ the transmitted signal and $\phi[i]$ the time-dependent phase offset.

This results in the following demodulated subcarrier values (see also Equation 3.19 and 3.7):

$$\widehat{C}_{m,n} = \sum_{i=0}^{N-1} C_{l,n}\, e^{\jmath\, 2\pi \frac{i}{N}(m-l)+\jmath\, \phi[i]} \tag{7.6}$$

with $\widehat{C}_{m,n}$ the received carrier m of the n^{th} OFDM symbol, $C_{l,n}$ the transmitted carrier l of the n^{th} OFDM symbol and N the FFT length.

Figure 7.4 shows the inter-subcarrier interference that is caused by a phase jitter with ${\sigma_\phi}^2 = 0.01$ $[rad^2]$. In the figure only one subcarrier value is transmitted. When these results are compared with the results of a DC offset in

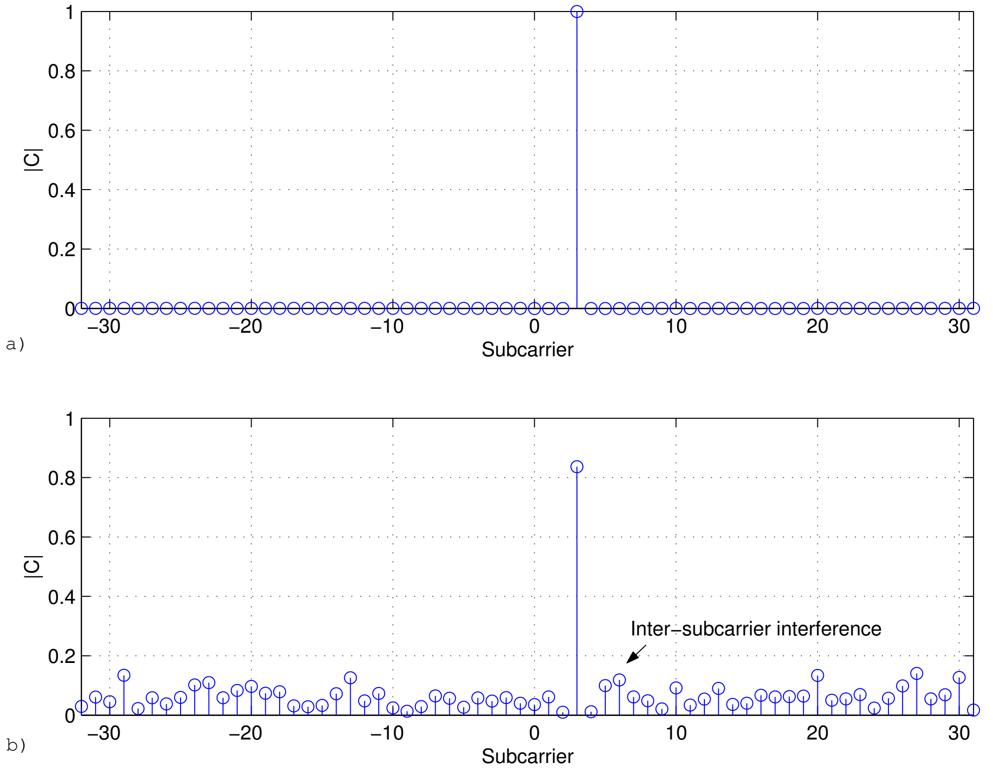


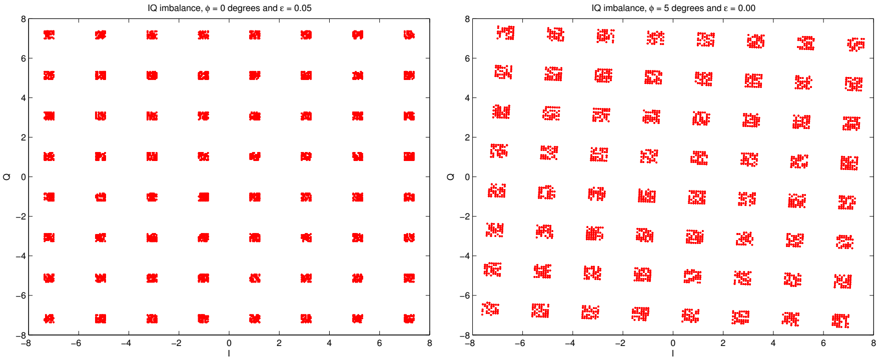
Figure 7.4: Influence of time dependent phase jitter on subcarriers; a) No phase noise present and b) Gaussian phase noise ($\sigma_\phi{}^2 = 0.01$ [rad^2]). These results are calculated with equation 7.6. Taken from [37].

combination with a frequency offset, we see that the first effect introduces inter-subcarrier interference locally, while phase jitter causes inter-subcarrier interference in all subcarriers.

In the designing of the transceiver hardware, mixers with great spectral purity should be chosen. Especially, because it is difficult to compensate for this effect in the receiver. The LOs which are used in our demonstrator do not have sufficient spectral purity (Section 6.1.5), so they have to be replaced by better ones.

I/Q imbalance

Another important distortion in OFDM systems is I/Q imbalance. This is introduced by differences in the analog domain between the I and Q path both in the transmitter and receiver [77]. Especially, differences in the reconstruction and anti-aliasing filters cause this effect and it is for this reason frequency dependent. For each carrier, I/Q imbalance consists of an amplitude



(a) amplitude mismatch

(b) phase mismatch

Figure 7.5: I/Q imbalance: effect of phase and amplitude mismatch.

and phase mismatch [48], [78]:

$$\widetilde{r}[i] = \alpha\widetilde{s}[i] + \beta\widetilde{s}^*[i] \tag{7.7}$$

with α:

$$\alpha = \cos\Delta\phi + \jmath\varepsilon\sin\Delta\phi \tag{7.8}$$

and β:

$$\beta = \varepsilon\cos\Delta\phi - \jmath\sin\Delta\phi \tag{7.9}$$

with ε the relative amplitude imbalance ($I = 1 + \varepsilon$ and $Q = 1 - \varepsilon$), $\Delta\phi$ the phase imbalance and $\widetilde{s}[i]$ the complex transmitted signal.

If there is no I/Q imbalance, the received signal is equal to the transmitted signal ($\varepsilon = 0.0$ and $\Delta\phi = 0°$). I/Q imbalance introduces two effects in an OFDM receiver. The first effect is crosstalk between positive and negative carriers (Equation 7.7). Figure 7.5 and Figure 7.6 illustrate this in the case that there is no noise present in the system. The shown constellations are from one carrier after channel equalization, just before de-mapping, point Q in Figure 3.10, page 52.

Figure 7.5 depicts separate the effects of a amplitude or phase imbalance. From this figure it can be concluded that a phase imbalance also biases the channel estimation algorithm -the constellation diagram is rotated a little-which is the second effect of an I/Q imbalance. Figure 7.6 shows for 64-QAM mode, the constellation diagram of a single carrier in case that there is both a phase and amplitude imbalance. The small constellations are now rotated.

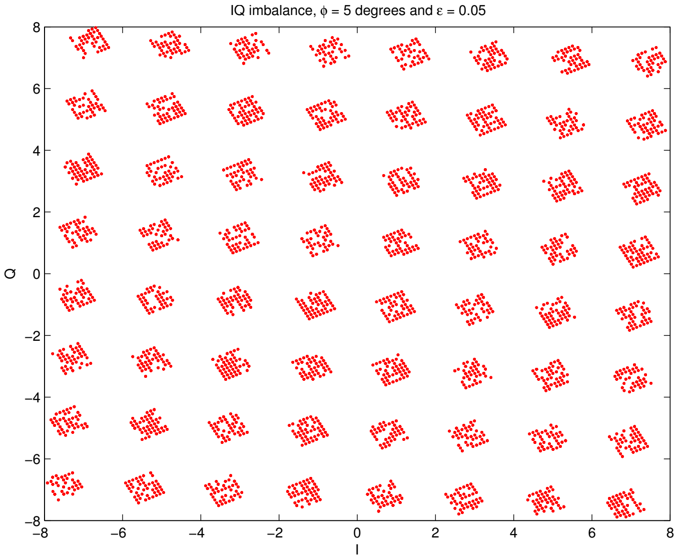


Figure 7.6: I/Q imbalance: $\phi = 5^{\circ}$ and $\varepsilon = 0.05$.

In order to have a PER of 10%, the highest bitrate mode in HiperLAN/2 (64-QAM) requires a SNR of 21.1 dB (Table 6.3). An I/Q imbalance should be much lower than this noise floor because it affects both the channel estimation and introduces crosstalk between positive and negative subcarriers:

$$C/I_{IQ} = \frac{\alpha}{\beta} \ll SNR_{required} \tag{7.10}$$

with E_b the bit energy (without prefix), R the bitrate, P_{DC} the power of the DC component and $SNR_{required}$ the required SNR to meet a PER of 10%.

Simulations for different levels of I/Q imbalances have been carried out in an AWGN channel with 240 kHz frequency offset. See Figure 7.7. If the I/Q imbalance is 24.3 dB ($\varepsilon = 0.03$ and $\phi = 3^{\circ}$) lower than the signal energy, it's influence can be neglected ($\approx$ 0.6 dB performance degradation).

Measurements in the testbed have shown that it is in our case necessary to implement a compensator for I/Q imbalance. At this moment, the testbed software does not contain such an algorithm. Further research is needed to research whether we can use I/Q imbalance compensators from literature, such as [78] in our HiperLAN/2 receiver. Another option is to implement a calibration phase in our testbed which estimates the I/Q imbalance.

Non-linearities

OFDM systems require linear analog components [49] [50]. Non-linear components in combination with feedback circuits, such as integrating filters may result in inter-carrier distortion. Also, clipping of the OFDM signal degrades

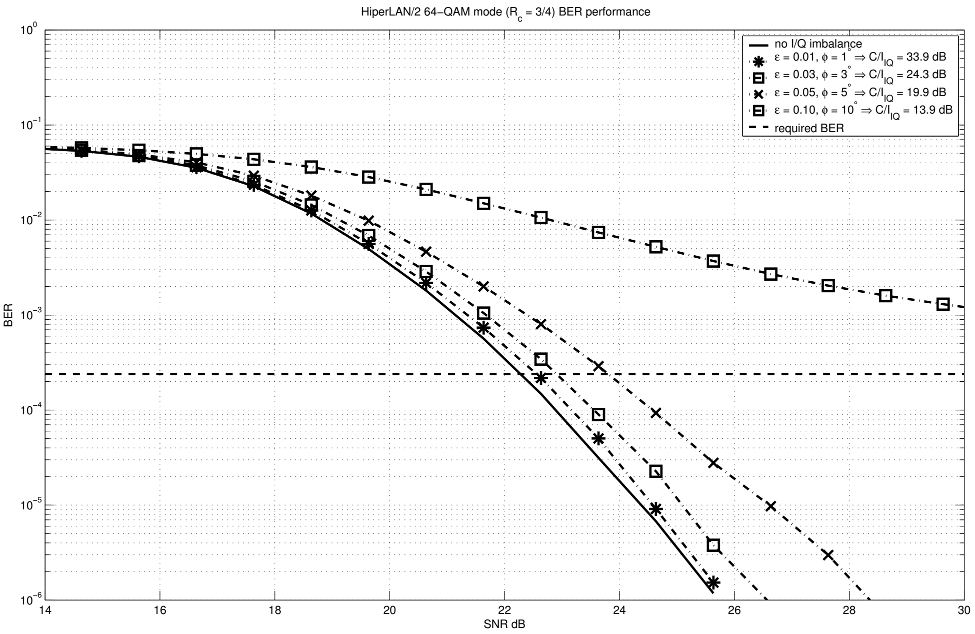


Figure 7.7: HiperLAN/2 BER performance with I/Q imbalance (BER versus SNR).

the performance of the system [79]. Currently, the dominant non-linearities in the analog front-end are not known. So, additional research is needed to identify them and to investigate if they can be solved in the digital domain. It is expected that the interference in the carriers -1 and 1 are also caused by non-linearities. [79] reports about non-ideal front-ends for OFDM systems.

7.2 Utilization

In our project, we have chosen to research to what extent a GPP processor (i.e. Pentium 4) can be used for our receiver algorithms. For the digital baseband functions, we have built a baseband transceiver that consists of four components; a transmitter PC, a DAC Printed Circuit Board (PCB), a receiver PC and an ADC PCB.

In a commercial product this would result in a PCI card equipped with 2 ADCs and 2 DACs plus analog reconstruction filters and digital anti-aliasing filters. See Figure 7.8(a). This implementation requires no extra DSP hardware which saves manufacturing costs. However, the PCI interface is fully utilized[4], disabling the use of other applications. In addition, the CPU is very

[4] The maximum transfer rate of the PCI standard is 133 Mbyte/s and the sustained data rate is about 80-90 Mbyte/s. Upcoming I/O standards like PCI Express [80] will solve this

power inefficient relative to an ASIC, although one can argue that the CPU is always consuming power whether it is used for our *program* or not.

To overcome the large load on the PCI bus we propose an implementation alternative where the ADCs, DACs and filters are implemented in the chipset of the notebook (Figure 7.8(b)). As there is already a large bandwidth available between chipset and CPU, this SDR application can easily be added. Power consumption will be slightly less than the first alternative due to less inter-chip communication. Moreover this integrated solution is more cost effective than the first option.

The last implementation alternative (Figure 7.8(c)) is a PCI board equipped with a low power DSP (e.g. a TI C64x DSP), ADCs, DACs and filters. So all receiver functions are performed by this PCI card. As DSPs are designed to compute digital signal processing algorithms at a low power consumption ($< 1W$), this solution is the most power efficient of the considered alternatives.

However, the future of this alternatives is uncertain because after integration of the sound, network and graphics, new computer chipset generations of Intel will have wireless LAN support [81] [82]. In these markets, every dollar counts and multi-standard support is not an important feature (low-budget solution). For the traveling businessman who encounters many wireless standards, an SDR mobile terminal has benefits. However, it is expected that this will only be a very small part of the market as the largest share of the wireless LAN products will be provided by these "chipset solutions"[5].

Power consumption

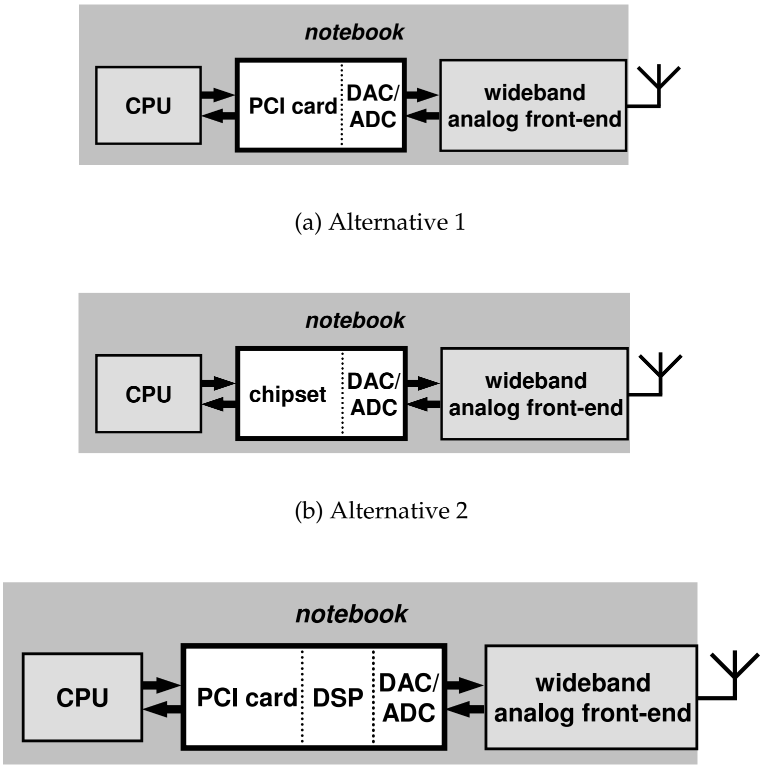
This section assesses the power consumption for the implementation alternatives. All alternatives share a common part which is the interface to the analog domain. This common part consists of 2 ADCs, 2 DACs and filters which is estimated to consume about 0.5 W[6]

Alternative 1, the testbed, and alternative 2, integration in the chipset, will have similar power consumption because they both use the Pentium-4 for the demodulation functions. Typical power consumption for a Pentium 4 is about 35 W at 2.4 GHz clock frequency [83]. So, power consumption will vary between $\frac{381}{2400} \cdot 35 + 0.5 \approx 6$ W (Bluetooth receiver program, Table 5.2, page 78) to $\frac{1225}{2400} \cdot 35 + 0.5 \approx 19$ W (HiperLAN/2 receiver program, Table 5.4, page 78).

problem.

[5] For example, nowadays almost computer chipsets contain sound-card functionality and the market for sound cards has almost vanished.

[6] We have assessed that 12-bit 40 MSPS ADCs and 10-bit 20 MSPS DACs can be used. Current ADCs, for example the dual AD9238 of Analog Devices consumes about 400 mW (200 mW per channel) [75] and current DACs such as the AD9740 DAC consumes about 35 mW per channel [75].



(a) Alternative 1

(b) Alternative 2

(c) Alternative 3

Figure 7.8: Implementation alternatives.

This power consumption can be reduced significantly if a low-power CPU, for example the Pentium-M [62] is used. Typical power consumption of a Pentium-M at 1.2 GHz is 12 Watt. Because the architecture of this processor is much more efficient that a normal Pentium 4 CPU, it is not possible to use the values of Tables 5.1, 5.2, 5.3 and 5.4 for this processor.

Alternative 3 uses a low power DSP instead for the demodulation algorithms. Typical power consumption for a TI C64x DSP running at 500 MHz is 1.0 W [63]). In this case, the power consumption varies between $\frac{56}{500} \cdot 1.0 + 0.5 \approx 0.6$ W (Bluetooth receiver program) to $\frac{203}{500} \cdot 1.0 + 0.5 \approx 1$ W (HiperLAN/2 receiver program). This alternative is more than a factor 10 energy efficient, but requires additional hardware which increases manufacturing costs[7].

[7] The price of a TI C64x DSP varies from \$ 25 to \$ 150 [63].

7.3 Conclusions

In the research of wireless radio communication one can identify several research areas. The first area is the theoretical view where for example, new modulation techniques are investigated. This thesis contributes to the second, more engineering-oriented view. In this field of research, one tries to invent smart solutions given an existing situation such as an abundance of standards. As each wireless communication standard needs its own radio, the SDR concept is emerging as a potential pragmatic solution: a software implementation of the user terminal, able to dynamically adapt to the radio environment in which the terminal is located, given the user demands at hand. In this thesis, we have described an SDR receiver for wireless LAN and PAN standards.

In our opinion, SDR is an implementation technology, mostly invisible for consumers. To validate and demonstrate our ideas about SDR, we have designed an SDR receiver at an functional level which is capable of receiving wireless LAN and PAN standards that use phase-modulation or OFDM. Moreover, an important goal in this project is to create a design that is feasible for a product within a few years with respect to manufacturing costs and power consumption. Due to *Moore's* law [4], current GPP have relatively large processing capabilities. Therefore, a third goal in this research is to investigate in which extent we can use the notebook's GPP processor (e.g. Pentium 4) for digital signal processing purposes.

First, we have successfully integrated the receiver of a phase-modulation based standard (i.e. Bluetooth) with an OFDM receiver (i.e. HiperLAN/2) at a functional level. By using the *Laurent* transformation, phase-modulated signals can be written as an sum of amplitude-modulated signal. This transformation has two benefits, first we can derive an optimal receiver, the MAP receiver and furthermore this type of receiver is more similar to the architecture of an OFDM receiver. However, a disadvantage of the MAP receiver is, that it requires exact knowledge of properties of the received signal, such as modulation index and frequency offset. The functional SDR receiver architecture has fulfilled the narrow scope objectives on page 14; it has enabled us to define reference points, interfaces and has delimited our demonstrator. In addition, the first wide-scope objective has been achieved because it has provided us to identify systems parts which can be integrated. However, the second wide-scope objective is not entirely met, because we have a fully software implementation of both demodulation functions. So, in order to swith functionality, the receiver has to run just a different "program".

A second goal of this project is to research to what extent we can use the existing processing capabilities of the notebook. We have shown in our testbed that all demodulation functions of the physical layer can be mapped on the CPU of the notebook. In the research of Bose [3], I/O was a bottleneck which is no longer valid for our SDR receiver because current-day computers have high-speed I/O standards. Nowadays, an important bottleneck is the latency of the operating system. This aspect of our SDR receiver is out of the scope of the project and in our testbed we have used large sample buffers of 100 ms to avoid the influence of the operating system.

In the introduction of this thesis we have posed SDR-specific challenges:

- **Power consumption**: Because one of the goals of our project is to build an SDR which is feasible within a few years we have decided to focus on a flexible radio in a notebook where power consumption is less an issue. In addition, SDR has already been adopted by the industry for basestations and for that reason SDR in mobile terminals seems to be a more interesting research area.
- **Clock generation**: In our SDR architecture, a single master clock (LO) is used, which runs at 80 MHz. Besides the Pentium 4, we use a COTS Programmable Down Converter (PDC) for channel selection. The output rates of this PDC are derived from this clock, so the sample-rate conversion function is relatively simple. The output signal of the PDC is buffered and transfered to the PC where all demodulation functions are performed.
- **Receiver architecture**: As the receiver complexity is four times the transmitter complexity, we have tried to use for the most demanding mode, OFDM, algorithms which are rather simple but have good performance.
- **Handset production**: Our SDR receiver uses few components because we have developed a CMOS wideband analog SDR front-end and use the notebook's CPU for signal processing purposes. This solution is relatively cheap to produce.

In a commercial product, the built testbed would result in a PCI card equipped with 2 ADCs and 2 DACs plus analog reconstruction filters and digital anti-aliasing filters. This implementation requires no extra DSP hardware which saves costs. However, the PCI interface is fully utilized, disabling the use of other applications. For this reason, we have proposed two other alternatives. The first alternative is to integrate the functionality of the PCI card into the chipset of motherboard and the second alternative uses a separate DSP for the demodulation functions. This alternative is more power efficient, but requires extra hardware. Another application of our testbed, is the testbed itself. The built testbed is very suitable for rapid prototyping. Furthermore, because all demodulation algorithms are in software, they can be altered or replaced very quickly which accelerates the design process.

After building the testbed, we have seen in Chapter 6 that our SDR receiver does not fulfill the requirements set by both standards entirely. This is caused by several reasons. The first reason is that the built analog SDR front-end performs worse than expected. Moreover, the testbed does not contain an analog AGC. This function is essential in a front-end and it is one of the reasons why the required sensitivity is not met in HiperLAN/2 mode. To decrease the phase noise, the used LOs needs to be replaced by more accurate ones. In the digital domain, the implemented receiver in OFDM mode is not complete. Due to time constraints, two functions are missing and should be implemented for a full functional receiver. The first function that is missing, is a DC-offset detection and correction algorithm which is necessary because the architecture of the analog front-end is a zero-IF architecture. The second missing function is I/Q imbalance detection and correction algorithm. Differences in the analog domain between the I and Q path both in the transmitter and receiver introduce this effect [77].

Recent literature such as [84] indicates that for HiperLAN/2, FEC coding and decoding (e.g. Viterbi algorithm), which we did not implement in the project, are the most demanding parts of the physical layer. Additional research has to be carried out whether this also holds for a Pentium 4 implementation and whether this limits a GPP-based software-defined radio. For the analog part of the receiver, most power is consumed by the AD conversion [51]. So, although power consumption of our SDR multi-band analog front-end is larger than for a single standard receiver front-end, its influence on the total power consumption is expected to be small.

In Chapter 1, a number of questions have been posed which have been answered throughout this thesis. For convenience, these answers are summarized here:

- **How to integrate a phase-modulation receiver and an OFDM receiver?**

  In this thesis, we have shown that we can integrate the digital part of a phase-modulation based standard, Bluetooth, and an OFDM-based standard, HiperLAN/2, at a functional level. Because other wireless LAN (and PAN) standards use the same modulation techniques and frequency bands we expect that this SDR receiver is also suited for these standards. Additional research is needed to investigate if spread-spectrum based standards (e.g. Code Division Multiple Access (CDMA)) can be mapped on this functional SDR receiver architecture.

- **To what extent can we use the notebook's CPU for our purposes or do we need other digital signal processing platforms?**

  In our project, we have investigated in which extent we can use the notebook's GPP i.e. x86 processor for digital signal processing purposes. It is possible to map the entire demodulation function of both standards on this processor. This choice saves hardware, but is also power inefficient. However, a critical issue in this solution is the latency of the operating system, because wireless standards require fast response times and the latency of traditional operating systems is too large.

- **Identify the best domain for performing an SDR function (analog/digital)?**

  At the start of the project we have decided to map the channel selection function both on the analog and digital part of the receiver and to perform the demodulation function complete in the digital domain. After building the testbed, it can be concluded that this has been a good choice. Performing more functions in the digital domain will increase power consumption significantly, as faster ADCs and more digital signal processing is needed. Moving, more functions to the analog domain will affect the flexibility of our receiver and is therefore not an option either. Another aspect in the design of an receiver, is how to compensate for distortions of the analog domain. If these distortions can be solved digitally it will enlighten the design of the analog front-end. Due to time constraints, more research is required to investigate this.

- **How to integrate the support of future standards?**

  Our SDR receiver can be extended to other standards, because the only limitations in our testbed are the maximal channel bandwidth of 20 MHz, the dynamic range of the wideband SDR analog front-end and the processing capabilities of the used PC. More research is required to investigate if our SDR testbed can also be used for spread-spectrum based standards (e.g. CDMA) and for example broadcast standards (e.g. TV, radio).

7.4 Suggestions for future research

Building a receiver front-end for wireless standards standards requires both theoretical and practical knowledge. For OFDM standards, the design of the baseband functionality of the analog front-end for both the receiver and transmitter is important as this part has most influence on the received signal. In this project, our main focus was on the analog RF front-end and digital baseband signal processing.

Further research should focus on the analog baseband functions. What are critical parameters in this part of the front-end? Can analog distortions be corrected in the digital domain, thereby relaxing the analog front-end design? What is the best domain for implementing a radio function (analog/digital)? How does phase noise of the LOs influence OFDM symbols? We have noticed in baseband experiments that the phase noise modeled as a Gaussian random process is not entirely correct.

At this moment, our SDR is not complete because there are some functions missing. The first function is the analog Automatic Gain Control (AGC). Without this function, the receiver cannot meet the sensitivity requirements of both standards. In addition, in OFDM mode, the receiver does not contain a DC offset and I/Q mismatch detection and correction algorithms. Further research should focus on extending the receiver with these functions. In addition, simulations with multi-path channels could be performed to investigate the effect of such channels on both receivers.

Moreover, we have implemented in the phase-modulation mode of the SDR receiver, the Park's demodulator and not the MAP receiver. The latter has much better performance, but this receiver needs a very accurate modulation-index estimation algorithm. At this moment, we do not have an suitable estimation algorithm [34]. Also, in OFDM mode we have chosen a particular architecture and algorithms. The receiver works in this mode, but other algorithms or architecture could improve performance or lower the computational complexity. An good example is the FFT. At this moment we use a 64-point FFT. Another option is to use a 128 or 256-point FFT. In this case, adjacent channels will appear in unused bins of the FFT. So, this solution can reduce the requirements of the channel selection function. However, reducing the requirements of the filters will also affect the timing and parameter estimation function in a negative manner. In addition, further research could investigate to what extent it is possible to alter the SDR receiver in such a way that it can receive multiple standards simultaneously.

Due to the limited time and manpower we have aimed in this project on the physical layer of only two standards, Bluetooth and HiperLAN/2. Future research could aim on extending the built SDR receiver with other standards. Because the *Laurent* transformation can be used for all phase-modulation standards, we believe that our receiver can be used both for OFDM standards and phase modulation standards. However, additional research has to verify this.

Besides single-carrier and multi-carrier transceivers (e.g. OFDM), there are spread-spectrum transceivers. Spread-spectrum based systems, such as CDMA, are mainly used in outdoor mobile communication standards, such as UMTS. More research is required to investigate if this type of receiver can be mapped on the functional SDR receiver architecture. As the only limitation in our testbed is the maximal channel bandwidth of 20 MHz, the SDR testbed can also be used for other types of communications, such as broadcast standards (e.g. TV, radio).

For a HiperLAN/2 transceiver, the most dominant part in the physical layer is FEC coding and decoding [84] which is not investigated in this project.

Further research could extend the receiver with this functionality and verify if this limits a GPP-based software-defined radio. Moreover, we have used the Pentium 4 as implementation platform, but the same C-code can also be mapped on other processors, such as the low-power Pentium-M or a DSP. Additional research could also focus on realizing the functional SDR receiver in hardware.

Another interesting topic is the operating system because our application requires a real-time (low-latency) operating system. For the analog front-end, the AD conversion requires most power. In our project we used Common Of The Shelf (COTS) ADCs and further research could aim at low-power ADCs for wireless communication applications.

The last suggestion for further research is to investigate a SDR *transmitter*, as we aimed in this research only at the receiver. Can the functional transmitter architecture of different types of standards be combined? (Literature, such as [85] indicates that this is true for certain types of modulation.) What are critical parameter in the transmitter designs? Is our transmitter architecture which is used in the testbed suitable for this purpose?

References

[1] J. Mitola. *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering*. Wiley, 2000.

[2] R.H. Walden. *Performance trends for analog-to-digital converters*. IEEE Communications Magazine, pages 96–101, February 1999.

[3] V.G. Bose. *Design and Implementation of Software Radios Using a General Purpose Processor*. PhD thesis, Massachusetts Institute of Technology, 1999.

[4] G.E. Moore. *Cramming more components onto integrated circuits*. Electronics magazine, 1965.

[5] E. Buracchini. *The Software Radio Concept*. IEEE Communciations Magazine, September 2000.

[6] *http://www.sdrforum.org*.

[7] P.G. Cook. *Software Defined Radio Semantics*. SDR Forum, 2000.

[8] V.J. Arkesteijn, E.A.M. Klumperink, and B. Nauta. *An Analogue Front-End Architecture for Software Defined Radio*. 13^{th} proRISC workshop, November 2002.

[9] V.J. Arkesteijn, E.A.M. Klumperink, and B. Nauta. *An Analogue Front-End Architecture for Software Defined Radio*. MMSA2002, December 2002.

[10] *http://www.intel.com/research/silicon/mooreslaw.htm*

[11] J. Hoffmeyer, I. Park, M. Majmundar, and S. Blust. *Radio Software Download for Commercial Wireless Reconfigurable Devices*. IEEE Communciations Magazine, March 2004.

[12] N. Georganopoulos, T. Farnham, R. Burgess, T. Schöler, J. Sessler, P. Warr, Z. Golubicic, F. Platbrood, B. Souvill, and S. Buljore. *Terminal-Centric View of Software Reconfigureable System Architecture and Enabling Components and Technologies*. IEEE Communciations Magazine, May 2004.

[13] `http://www.semiconductors.philips.com/comms`.

[14] T.A.C.M. Claasen. *High speed: the only way to exploit the intrinsic computational power of silicon*. ISSCC'99, 1999.

[15] J. Mitola. *Technical challenges in the Globalization of Software Radio*. IEEE Communications Magazine, pages 84–89, February 1999.

[16] *The Bluetooth-HiperLAN/2 SDR receiver project website*. `http://www.sas.el.utwente.nl/home/SDR/`.

[17] ETSI. *Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer*. Technical Specification ETSI TS 101 475 V1.2.2 (2001-02), ETSI, February 2001.

[18] BluetoothSIG. *Specification of the Bluetooth System - Core*. Technical Specification Version 1.1, 'Bluetooth SIG', February 2001.

[19] `http://www.eetimes.com/showArticle.jhtml?articleID=19200138`.

[20] W.H.W. Tuttlebee. *Software Defined Radio: Origins, Drivers and International Perspectives*. 2002.

[21] `http://www.sas.el.utwente.nl/home/SDR/docs/SDRprojects.pdf`.

[22] F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Functional Analysis of a SDR Based Bluetooth/HiperLAN Terminal Demonstrator*. 2^{nd} PROGRESS workshop on Embedded Systems and Software, 2001.

[23] V.J. Arkesteijn, R. Schiphorst, F.W. Hoeksema, E.A.M. Klumperink, B. Nauta, and C.H. Slump. *A Software-Defined Radio Test-bed for WLAN Front Ends*. 3^{nd} PROGRESS workshop on Embedded Systems and Software, 2002.

[24] T.S. Rappaport. *Wireless Communications*. Prentice Hall, Inc., 1996.

[25] R. de Buda. *Coherent Demodulation of Frequency-Shift Keying With Low Deviation Ratio*. IEEE transactions on communications, 20:466–470, June 1972.

[26] M.H.L. Kouwenhoven. *High-Performance Frequency-Demodulation Systems*. PhD thesis, Delft University of Technology, 1998.

[27] A.B. Carlson. *Communication Systems*. McGraw-Hill, Inc., 1986.

[28] J.C. Haartsen. *The Bluetooth Radio System*. IEEE Personal Communications, 7(1):28–36, February 2000.

[29] L.C. van Mourik. *Design & Implementation of Digital Channel Selection Filters for a Combined Bluetooth and Hiperlan/2 Receiver*. Master's thesis, University of Twente, August 2002.

[30] J.H. Park. *An FM Detector for Low S/N*. IEEE Transactions on Communications Technology, COM-18(2):110–118, April 1970.

[31] P.A. Laurent. *Exact and appropiate construction of digital phase modulateds by superposition of amplitude modulated pulses (AMP)*. IEEE transactions on communications, 34(2):150–160, February 1986.

[32] U. Mengali. *Decomposition of M-ary CPM Signals into PAM Waveforms*. IEEE transactions on information theory, 41(5):1265–1275, September 1995.

[33] G. Kawas Kaleh. *Simple Coherent Receivers for Partial Response Continous Phase Modulation*. IEEE Journal on Selected Areas in Communications, 7(9):1427–1436, December 1989.

[34] F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Modulation-index estimation in a combined CPM/OFDM receiver*. Proceedings of the 4th SPS2004 Symposium, April 2004.

[35] ETSI. *Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; System Overview*. Technical Report ETSI TR 101 683 V1.1.1 (2000-02), ETSI, February 2000.

[36] M. Johnsson. *HiperLAN/2 - The Broadband Radio Transmission Technology Operating System in the 5 GHz Frequency Band*. HiperLAN/2 Global Forum, 1999. `http://www.hiperlan2.com`.

[37] L.F.W. van Hoesel. *Design and implementation of HiperLAN/2 physical layer models for simulation purposes*. Master's thesis, University of Twente, August 2002.

[38] R. van Nee and R. Prasad. *OFDM for wireless multimedia communications*. Artech House, 2000.

[39] *`http://itpp.sourceforge.net`*.

[40] R.L. Peterson and R.E. Ziemer. *Introduction to digital communication*. Prentic Hall, second edition edition, 2001.

[41] F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Spectral Weighting Functions for Single-symbol Phase-noise Specifications in OFDM Systems*. Proceedings of the 8th International OFDM-Workshop, September 2003.

[42] P. Mandari, T. Keller, and L. Hanzo. *Orthogonal Frequency Division Multiplexing Techniques for Frequency-Selective Fading Channels*. IEEE Journal on Selected Areas in Communications, 19(6), June 2001.

[43] T.M. Schmidl and D.C. Cox. *Robust Frequency and Timing Synchronisation for OFDM*. IEEE Journal on Selected Areas in Communications, 45(12), December 1997.

[44] M. Sandell, J.J. van der Beek, and P.O. Borjesson. *ML estimation of time and frequency offset in OFDM systems*. IEEE Journal Transactions on Signal Processing, 45(7), July 1997.

[45] S. Srinivasan. *Comparison of Combined Synchronisation and Frequency Offset Estimation Algorithms for HIPERLAN/2*. Bachelor's thesis, University of Twente, February 2004.

[46] A. Berno. *Time and Frequency Synchronization Algorithms for HiperLAN/2*. Master's thesis, University of Padova, October 2001.

[47] V.J. Arkesteijn. *To appear: A wide-band analog front-end for Software Defined Radio*. PhD thesis, Univerisity of Twente, 2005.

[48] B. Razavi. *RF Microelectronics*. 1998.

[49] K. Baldwin, K. Halford, and S. Halford. *Secrets of OFDM Systems Engineering (Sheets)*. OFDM forum - OFDM workshop for WLAN applications, 2001.

[50] M. Banu, V. Prodanov, P. Kiss, and D. Manstretta. *The Challenges of Fully Integrated OFDM Transceivers for 802.11 Wireless-LAN Systems (Sheets)*. ISSCC'03 GIRAFFE workshop, 2003.

[51] P. Wambacq, G. Gielen, E. Janssens, G. Vandersteen, and P. Vanassche E. Lauwers. *SALOMON project*. October 2000.

[52] F.W. Hoeksema. *Three Filters for HiperLAN2 Channel Selection*. Internal report (R006), July 2002.

[53] A.V. Oppenheim and R.W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, Inc., 2002.

[54] J.G. Proakis. *Digital Communication*. McGraw Hill, fourth edition, 2001.

[55] J.E. Gunn. *A Low-Power DSP Core-Based Software Radio Architecture*. IEEE Journal on Selected Areas in Communications, April 1999.

[56] *http://chameleon.ctit.utwente.nl/*.

[57] I. Ripoll, P. Pisa, L. Abeni, P. Gai, A. Lanusse, S. Saez, and B. Privat. *Deliverable D1.1 - RTOS Analysis*. 2002. http://www.mnis.fr/opensource/ocera/rtos/.

[58] *http://www.intel.com/products/mobiletechnology/prowireless.htm*

[59] *http://www.adlinktech.com*.

[60] *http://www.intersil.com*.

[61] *http://www.fftw.org*.

[62] *http://www.intel.com*.

[63] *http://www.ti.com*.

[64] J. Edward (Blue Wave Systems). *Performance Evaluation of the TMS320C64x Architecture and C Compiler*. 2000. http://mcg.motorola.com/us/general/bwswp.pdf

[65] Texas Instruments. *TMS320C6000 CPU and Reference Guide (SPRU189F)*. 2000.

[66] *http://www.agilent.com*.

[67] V.J. Arkesteijn, R. Schiphorst, F.W. Hoeksema, E.A.M. Klumperink, B. Nauta, and C.H. Slump. *A combined receiver front-end for Bluetooth and HiperLAN/2*. 4^{nd} PROGRESS workshop on Embedded Systems and Software, 2003.

[68] F. Bruccoleri, E.A.M. Klumperink, and B. Nauta. *Wide-Band CMOS Low-Noise Amplifier Exploiting Thermal Noise Canceling*. JSSC, 39(2):275–282, February 2004.

[69] Ericsson. *Ericsson Bluetooth Development Kit (EBDK) documentation*. October 1999.

[70] C. Sundberg J.B. Anderson, T. Aulin. *Digital Phase Modulation*. 1986.

[71] A. Soltanian and R.E. Van Dyck. *Performance of the Bluetooth System in Fading Dispersive Channels and Interference*. Proceedings of IEEE Globecom, November 2001.

[72] J. Walrand. *Communication Networks*. McGraw-Hill, U.S.A., second edition, 1998.

[73] J. Medbo and P. Schramm. *Channel Models for Hiperlan/2 in Different Indoor Scenarios*. Technical Specification ETSI EP BRAN 3ERIO85B, ETSI, March 1998.

[74] J. Khun-Jush, P. Schramm, U. Wachsmann, and F. Wenger. *Structure and Performance of the HiperLAN/2 Physical layer*. IEEE VTC conference, 1999.

[75] *http://www.analog.com*.

[76] L. Hanzo T. Keller W. Web. *Single- and Multi-carrier Quadrature Amplitude Modulation; Principles and Applications for Personal Communications, WLANs and Broadcasting*. Wiley, U.S.A., 2000.

[77] S. Lang, R. Mysore Rao, and B. Daneshrad. *Design and Implementation of a 5 GHz Software Defined Wireless OFDM Communication Platform*. IEEE Communciations Magazine, June 2004.

[78] J. Tubbax, B. Côme, L. van der Perre, L. Deneire, S. Donny, and M. Engels. *Compensation of IQ imblance in OFDM systems*. ICC, 2003.

[79] B. Côme, R. Ness, S. Donnay, L. Van der Perre, W. Eberle, P. Wambacq, M. Engels, and I. Bolsens. *Impact of Front-End Non-Idealities on Bit Error Rate Performances of WLAN-OFDM Transceivers*. RAWCON2000, 2000.

[80] *http://www.pcisig.com/home*.

[81] *http://www.infoworld.com/article/03/12/01/HNpcap_1.html*.

[82] R. Raaijmakers. *Agere zet mes in WLAN-design*. Bits & Chips, (6^e jaargang, nummer 9), April 2004. http://www.bits-chips.nl/info.asp?mode=&ac=view&inf_id=3603

[83] *http://www.sandpile.org*.

[84] K. Masselos, S. Blionas, and T. Rautio. *Reconfigurability requirements of wireless communication systems*. IEEE hetrogeneous reconfigurable systems-on-chip workshop, 2004. http://www.imec.be/adriatic/pubs/paper_200204_kmas.zip

[85] A. Wiesler and F.K. Jondral. *A Software Radio for Second- and Third-Generation Mobile Systems*. IEEE transactions on vehicular technology, 51(4):738–748, July 2002.

List of publications

[1] F.W. Hoeksema, S. Srinivasan, R. Schiphorst, and C.H. Slump. *Timing Metrics of Joint Timing and Carrier-Frequency Offset Estimation Algorithms for TDD-based OFDM Systems*. Proceedings of the 9^{th} International OFDM-Workshop, September 2004.

[2] R. Schiphorst, F.W. Hoeksema, and C.H. Slump. *Implementierungsalternative für einen flexiblen drahtlosen wireless LAN transceiver*. Frequenz journal, June 2004.

[3] F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Modulation-index estimation in a combined CPM/OFDM receiver*. Proceedings of the $4^{t}h$ SPS2004 Symposium, April 2004.

[4] R. Schiphorst, F.W. Hoeksema, V.J. Arkesteijn, C.H. Slump, E.A.M. Klumperink, and B. Nauta. *Implementation Alternatives for a Flexible Wireless LAN Transceiver*. Proceedings of the $4^{t}h$ SPS2004 Symposium, April 2004.

[5] R. Schiphorst, F.W. Hoeksema, and C.H. Slump. *Implementation Alternatives for a Flexible Wireless LAN Transceiver*. Workshop on Software Radios (WSR2004), Karlsruhe (G), March 2004.

[6] F.W. Hoeksema, K.L. Hofstra, J. Potman, and R. Schiphorst. *The name is Link Radio Link*. Jaarboek Scintilla, 2002-2003, pages 137–148, 2003.

[7] R. Schiphorst, F.W. Hoeksema, and C.H. Slump. *A flexible WLAN receiver*. 14^{nd} proRISC workshop on Circuits, Systems and Signal Processing, November 2003.

[8] V.J. Arkesteijn, R. Schiphorst, F.W. Hoeksema, E.A.M. Klumperink, B. Nauta, and C.H. Slump. *A combined receiver front-end for Bluetooth and HiperLAN/2*. 4^{nd} PROGRESS workshop on Embedded Systems and Software, 2003.

[9] R. Schiphorst, F.W Hoeksema, and C.H. Slump. *A Bluetooth-enabled HiperLAN/2 receiver*. Proceedings of the VTC Fall 2003, October 2003.

[10] F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Spectral Weighting Functions for Single-symbol Phase-noise Specifications in OFDM Systems*. Proceedings of the 8^{th} International OFDM-Workshop, September 2003.

[11] R. Schiphorst, F.W Hoeksema, and C.H. Slump. *A (simplified) Bluetooth Maximum A posteriori Probability (MAP) receiver*. Proceedings of IEEE SPAWC2003, June 2003.

[12] L.F.W. van Hoesel, F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Frequency Offset Correction in a Software Defined HiperLAN/2 Demodulator Using Preamble Section A*. Proceedings of the MMSA2002 Conference, December 2002.

[13] L.C. van Mourik, R. Schiphorst, F.W. Hoeksema, and C.H. Slump. *Performance Evaluation of a Combined HiperLAN/2-Bluetooth Digital Front-end*. 13^{nd} proRISC workshop on Circuits, Systems and Signal Processing, November 2002.

[14] V.J. Arkesteijn, R. Schiphorst, F.W. Hoeksema, E.A.M. Klumperink, B. Nauta, and C.H. Slump. *A Software-Defined Radio Test-bed for WLAN Front Ends*. 3^{nd} PROGRESS workshop on Embedded Systems and Software, 2002.

[15] C.H. Slump, R. Schiphorst, and F.W. Hoeksema. *On AD Conversion for Telecommunications*. Proceedings of the SPS-2002 Conference, March 2002.

[16] F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Specification for Digital Channel Selection Filters in a Bluetooth Capable HiperLAN/2 Receiver*. Workshop on Software Radios (WSR2002), Karlsruhe (G), March 2002.

[17] R. Schiphorst, F.W. Hoeksema, and C.H. Slump. *Bluetooth demodulation algorithms and their performance*. Workshop on Software Radios (WSR2002), Karlsruhe (G), March 2002.

[18] F.W. Hoeksema, R. Schiphorst, and C.H. Slump. *Functional Analysis of a SDR Based Bluetooth/HiperLAN Terminal Demonstrator*. 2^{nd} PROGRESS workshop on Embedded Systems and Software, 2001.

[19] R. Schiphorst, F.W. Hoeksema, and C.H. Slump. *Channel selection requirements for Bluetooth receivers using a simple demodulation algorithm*. 12^{nd} proRISC workshop on Circuits, Systems and Signal Processing, November 2001.

[20] R. Schiphorst, F.W. Hoeksema, and C.H. Slump. *Front-end architectures of Software-Defined Radio, an exploration*. Proceedings of the ICASSP2001 Conference, May 2001.

[21] V.J. Arkesteijn, R. Schiphorst, F.W. Hoeksema, E.A.M. Klumperink, B. Nauta, and C.H. Slump. *Software Defined Radio*. 11nd proRISC workshop on Circuits, Systems and Signal Processing, November 2001.

Nomenclature

α	Wanted component of I/Q imbalance
α_n	The n^{th} symbol
β	Crosstalk component of I/Q imbalance
$\beta_{k,i}$	Radix-2 representation of k
$\boldsymbol{\alpha}$	Symbol sequence
$\Delta\phi$	I/Q phase imbalance
Δ_f	Subcarrier spacing
ϕ	Phase signal
$\phi(t)$	Time varying phase noise
ψ	Output signal of the Park's demodulator
θ_0	Phase offset
$\tilde{s}$	Transmitted complex-envelope signal
ε	I/Q amplitude imbalance
$\widehat{\delta f}$	Estimated frequency offset
$\widehat{\phi}_{common}$	Common phase offset
$\widehat{C}_{l,preambleC}$	The l^{th} received carrier value of the preamble C
$\widehat{C}_{m,n}$	The m^{th} subcarrier value of the n^{th} OFDM symbol
$\widehat{C}_{m,n}$	The value of subcarrier m of the n^{th} received OFDM symbol.
$\widehat{C}^{a}_{m,n}$	Complex value of the received carrier m of the n^{th} OFDM after channel equalization

$\widehat{C}^{b}_{m,n}$	Complex value of the received carrier m of the n^{th} OFDM after channel equalization and common phase offset correction
$\widehat{H}_{f,l}$	Channel estimation in the frequency domain of the l^{th} carrier value
$\widetilde{r}$	Received baseband signal
$\widetilde{r}_c$	Received baseband signal after frequency-offset correction
$\widetilde{s_n}$	Baseband signal of the transmitted n^{th} OFDM symbol
A_{DC}	DC offset
B	Bandwidth [Hz]
b_n	Bit stream after FEC coding
$b_{k,n}$	Pseudo symbol
C	The correlation length
C/I_{DC}	Carrier-to-DC-offset-Interference ratio
C/I_{IQ}	Carrier-to-IQ-imbalance-Interference ratio
c_k	PAM waveform
$C_{l,preambleC}$	Transmitted carrier value of the preamble C section
$C_{l,n}$	The n^{th} transmitted OFDM symbol in the frequency domain
d^2	Squared Euclidean distance between two symbol sequences
d_n	Bit stream before FEC coding
E_b	Bit energy
f_c	Carrier frequency
f_d	Frequency deviation
f_{sample}	Sample frequency
g	Frequency response function
h	Modulation index
H_0	Matched filter for the first Laurent waveform
I	In-phase signal

K_{mod}	Modulation type dependent normalization factor
L	Duration of the phase response function
L	The distance between the correlation segments
M	Timing metric
N	Observation period
N_0	Single-sided spectral density of the AWGN noise
N_{BPSC}	Coded bits per subcarrier
N_{CBPS}	Coded bits per OFDM symbol
N_{DBPS}	Data bits per OFDM symbol
N_{SD}	Number of data carriers
N_{SP}	Number of pilot carriers
N_{ST}	Total number of carriers
P_b	Bit-Error Rate (BER)
p_n	Pilot value n
P_s	Signal power
P_{b_O16QAM}	Bitrate of an OFDM signal with 16-QAM modulation
P_{b_O64QAM}	Bitrate of an OFDM signal with 64-QAM modulation
P_{b_OBPSK}	Bitrate of an OFDM signal with BPSK modulation
P_{b_OQPSK}	Bitrate of an OFDM signal with QPSK modulation
P_{DC}	DC-offset power
P_e	Packet-error rate
P_{s_BPSK}	Signal power of a BPSK signal
P_{s_OBPSK}	Signal power of an OFDM signal with BPSK modulation
Q	Q-function
Q	Quadrature signal
q	Phase response function

R	Bit-rate
r	Input signal of the MAP receiver
R_c	Code rate of a Convolutional encoder
R_{OBPSK}	Bitrate of an OFDM signal with BPSK modulation
T	Symbol time
t	Time
T_{CP}	Cyclic prefix duration
T_S	Symbol time
T_U	Useful symbol part duration
X_n	State n of the scrambler

Acronyms

2G	2^{nd} Generation
3G	3^{rd} Generation
ACL	Asynchronous Connection-Less
AD	Analog-to-Digital
AFC	Access Feedback Channel
AGC	Automatic Gain Control
ASIC	Applications-Specific IC
ADC	Analog-to-Digital Converter
AM	Amplitude Modulation
ARP	Antenna Reference Point
AWGN	Additive White Gaussian Noise
BCH	Broadcast CHannel
BER	Bit-Error Rate
Bluetooth	low-cost wireless standard named after a Danish king
BM	Branch Metric
BPF	Band-Pass Filter
BPSK	Binary Phase-Shift Keying
BT	Bandwidth Time product
CDMA	Code Division Multiple Access
CMOS	Complementary Metal Oxide Semiconductor

C/I	Carrier to Interference
COTS	Common Of The Shelf
CP	Cyclic Prefix
CPM	Continuous-Phase Modulation
CPU	Central Processing Unit
CRP	Channel Reference Point
CT2	Cordless Telephone 2
CT2+	Cordless Telephone 2+
DA	Digital-to-Analog
DAC	Digital-to-Analog Converter
DBPSK	Differential Binary Phase-Shift Keying
DC	Direct Current
DECT	Digital Enhanced Cordless Telecommunications
DLC	Data Link Control
DQPSK	Differential Quadrature Phase-Shift Keying
DRP	Demodulation Reference Point
DSP	Digital Signal Processor
ETSI	European Telecommunications Standard Institute
FCH	Frame control CHannel
FEC	Forward Error Correction
FFT	Fast Fourier Transform
FIFO	First In, First Out
FIR	Finite Impulse Response
FIRST	Flexible Integrated Radio Systems Technology
FH	Frequency Hopping
FM	Frequency Modulation

FPGA	Field Programmable Gate Array
FSK	Frequency Shift Keying
GloMo	Global Mobile
GPP	General Purpose Processor
GFSK	Gaussian Frequency Shift Keying
GMSK	Gaussian Minimum Shift Keying
GSM	Global System for Mobile communication
HiperLAN/2	HIgh-PErformance Radio Local-Area Network/2
HomeRF	Home Radio Frequency
HR	Hardware Radio
HW/SW	HardWare/SoftWare
I	In-phase component
IC	Integrated Circuit
IDFT	Inverse Discrete Fourier Transformation
IF	Intermediate Frequency
IFFT	Inverse Fast Fourier Transform
IMT-2000	International Mobile Telecommunication 2000
I/O	Input/Output
IIP$_3$	Third-Order Intermodulation Intercept Point
IS-136	EAI Interim Standard 136 - U.S. Digital Cellular with Digital Control Channels
IS-54	EAI Interim Standard for U.S. Digital Cellular
IS-95	EAI Interim Standard for U.S. Code Divsion Multple Access
ISI	Inter-Symbol Interference
ISM	Industrial, Scientific and Medical
IEEE	Institute of Electrical and Electronics Engineers

ISDR	Ideal Software-Defined Radio
L2CAP	Logical Link Control and Adaptation Protocol
LAN	Local-Area Network
LLC	Logical Link Control
LNA	Low-Noise Amplifier
LO	Local Oscillator
LPF	Low-Pass Filter
MAC	Medium Access Control
MAP	Maximum A posteriori Probability
MMSE	Minimum Mean Square Error
Modem	Modulator/Demodulator
MOPS	Million Operations Per Second
MRP	MAC Reference Point
MSPS	Million Samples Per Second
NF	Noise Figure
NWO	Nederlandse Organisatie voor Wetenschappelijk Onderzoek
OFDM	Orthogonal Frequency Division Multiplexing
OQPSK	Offset Quadrature Phase-Shift Keying
OSI	Open System Interconnection protocol model
PAM	Pulse Amplitude-Modulated
PAN	Personal Area Network
PER	Packet-Error Rate
PASTORAL	Platform And Software for Terminals: Operationally ReconfigurAble
PC	Personal Computer
PCB	Printed Circuit Board

PCI	Peripheral Component Interconnect
PDC	Programmable Down Converter
PER	Packet-Error Rate
PHS	Personal Handyphone System
PHY	Physical layer
PRP	Physical layer Reference Point
PROGRESS	PROGram for Research on Embedded Systems & Software
PROMURA	PROgrammable Multimode Radio for Multimedia Wireless Terminals
Q	Quadrature-phase component
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
RCH	Random access CHannel
RF	Radio Frequency
RMS	Root-Mean Square
RP	Reference Point
SaS	Signals and Systems
SAW	Surface Acoustic Wave
SCO	Synchronous Connection-Oriented
SCR	Software-Controlled Radio
SDR	Software-Defined Radio
SDRF	SDR Forum
SDU	Service Data Unit
SIMD	Single Input Multiple Data
SLATS	Software Libraries for Advanced Terminal Solutions
SNR	Signal-to-Noise Ratio

SODERA	re-configurable radio for SOftware DEfined RAdio for 3^{rd} generation mobile terminals
SORT	SOftware Radio Technology
SR	Software Radio
STW	Stichting voor de Technische Wetenschappen
SUNBEAM	Smart Universal BEAMforming
TDMA	Time Division Multiple Access
TM	Timing Metric
UMTS	Universal Mobile Telecommunication System
USR	Ultimate Software Radio
VCO	Voltage-Controlled Oscillator
WLAN	Wireless LAN
x86	Intel and compatible computer processors

Acknowledgments

Many people have contributed in some way to the realization of this thesis. First of all I want to thank (in alphabetic order): Bram Nauta, Erik Klumperink, Fokke Hoeksema, Jordy Potman, Harm Cronie, Kees Slump, Klaas Hofstra and Vincent Arkesteijn for the fruitful discussions, comments and advices. Furthermore, I am grateful to members of my promotion commitee Prof.dr.ir. J.C. Haartsen, Prof.dr.ir. J.W.M. Bergmans, Prof.dr.ir. A.-J. van der Veen, Prof.dr.ir. Th. Krol and Prof.dr. W.H.M. Zijm for reading this dissertation and to evaluate my research. I also want to thank Geert-Jan Laanstra and Henny Kuipers for building and testing of the SDR testbed. Others have contributed in a less explicit but also valuable way by creating and maintaining a pleasant working atmosphere.

Curriculum Vitae

Roel Schiphorst received his M.Sc. degree in Electrical Engineering from the University of Twente in 2000 for his research on software-defined radio. In September 2000 he joined the Signal and Systems group of the University of Twente as a Ph.D. student. In the summer of 2004, he finished his Ph.D. thesis on software-defined radio which describes a flexible multi-standard radio for wireless LAN standards. Since September 2004, he is working as a research scientist at the same chair.