Stefan Grönroos

Efficient and Low-Cost Software Defined Radio on Commodity Hardware

Turku Centre *for* Computer Science

TUCS Dissertations
No 218, November 2016

Efficient and Low-Cost Software Defined Radio on Commodity Hardware

Stefan Grönroos

To be presented, with the permission of the Faculty of Science and Engineering of Åbo Akademi University, for public criticism in Auditorium XX in Agora, Vesilinnantie 5 on November 11, 2016, at 12 noon.

Åbo Akademi University
Faculty of Science and Engineering
Agora, Vesilinnantie 3, 20500, Turku, Finland

2016

Supervisor

Docent Jerker Björkqvist
Department of Information Technologies
Åbo Akademi University
Vattenborgsvägen 3, 20500 Åbo,
Finland

Reviewers

Professor Manuel Vélez
Department of Communications Engineering
Faculty of Engineering
University of the Basque Country (UPV/EHU)
Alda. Urkijo S/N., 48013 Bilbao
Spain

Professor Riku Jäntti
Department of Communications and Networking
Aalto University
Otakaari 5, Espoo
Finland

Opponent

Professor Manuel Vélez
Department of Communications Engineering
Faculty of Engineering
University of the Basque Country (UPV/EHU)
Alda. Urkijo S/N., 48013 Bilbao
Spain

ISBN 978-952-12-3456-9
ISSN 1239-1883

Abstract

As general purpose processors become faster, many signal processing functions that used to require special purpose hardware, are now feasible to perform in software. In wireless communications, this means we move towards the "ideal" software-defined radio (SDR), where we have an antenna and analog/digital converters, after which the signal is processed in software.

Defining most of the functionality in software makes software defined radios very flexible, in that they can be reconfigured quickly to perform any task for which they have sufficient computational capacity.

This thesis is composed of a number of studies into high-performance SDR, and very low-cost SDR. The first series of studies explore how well modern programmable general purpose processors are able to run typical signal processing tasks in the physical layer of a modern communications standard: DVB-T2. In the second series of studies, a radio spectrum monitoring system is built using very low-cost components, and some software.

The first series of studies was initiated by an analysis of the DVB-T2 standard for digital television broadcasting. The most computationally complex signal processing functions of a DVB-T2 receiver were implemented on a graphics processing unit (GPU), and various central processing units (CPUs). The objective of these studies was to show that a real-time capable receiver for a modern standard such as DVB-T2 is feasible to implement completely in software on modern off-the-shelf computers.

The second series of studies focuses on distributed radio frequency spectrum monitoring using very low-cost sensors. The sensors are based on cheap off-the-shelf hardware, which was never intended for spectrum monitoring. The nodes were instead adapted to the tasks of spectrum sensing and geolocation of transmitters by running specialized software.

Sammanfattning

De processorer som används för att köra applikationer på datorer och t.ex. mobiltelefoner blir allt snabbare och effektivare. Dessa processorers stora fördel är deras flexibilitet; de kan programmeras för att utföra en mängd olika uppgifter. Processorns uppgift definieras alltså av mjukvaran som körs på den för tillfället. En processortillverkare behöver alltså inte på förhand veta exakt vilka uppgifter processorn kommer att utföra under sin livstid.

Mjukvarudefinierad radio — eller SDR från engelskans Software Defined Radio-– handlar om att låta en programmerbar plattform, t.ex. en generell processor, sköta största delen av uppgifterna i en radiomottagare eller -sändare i mjukvara. Normalt sköts en stor del av den beräkningsintensiva signalbehandlingen i en radio av komponenter specifikt designade för uppgiften. Man använder t.ex. oftast ett speciellt mikrochip för att behandla signalerna från GPS-satelliter och beräkna en position, ett annat chip för att ta emot digital-tv sändningar, och ett eller flera olika chip för att kommunicera med olika typer av trådlösa nätverk. Dessa chip utför sin uppgift effektivt, men är inte flexibla på samma sätt som en processor som kan växla mellan att köra ett datorspel och en ordbehandlare endast genom att ladda in ny mjukvara. Det är också dyrt att designa ny hårdvara för en specifik uppgift.

Målet med SDR är att minska mängden specialiserad hårdvara i en radio för att bygga mer flexibla system. En ideell mjukvaruradio kan t.ex. vara en GPS-navigator, en digital-tv mottagare, eller en mobiltelefon, beroende på vilken mjukvara som körs på den. Detta betyder också att förbättringar på kommunikationsstandarder kan göras genom att leverera ny mjukvara i likhet med hur mjukvaruuppdateringar görs för applikationer på en dator.

I denna avhandling beskrivs ett antal studier, indelade i två grupper, relaterade till SDR. I den första gruppen studier undersöks hur de tunga beräkningsoperationerna som behövs i en DVB-T2 digital-tv mottagare kan utföras på en vanlig konsument-PC utrustad med ett grafikkort. En studie om hur en typisk strömsnål processor för mobiltelefoner kan utföra dessa beräkningar ingår också. Målet med dessa studier är att visa hur moderna datorers beräkningskapacitet kan utnyttjas för att bygga en mjukvaruradio som klarar av en modern trådlös kommunikationsstandard som DVB-T2.

I den andra gruppen studier undersöks hur väldigt billig hårdvara som kostar ca 50€ kan användas för att övervaka radiospektrets användning. Denna hårdvara var inte ursprungligen designad för att övervaka spektrumanvändningen, men genom specialiserad mjukvara blir detta möjligt. En uppdatering till mjukvaran möjliggjorde också användningen av denna billiga hårdvara för att estimera positionen på okända radiosändare i området.

Acknowledgments

I started working on this thesis at the Embedded Systems Laboratory (ESLab) at Åbo Akademi University in 2010, after completing my master's thesis at the same laboratory. There are many people I would like to thank for making these 6+ years a very enjoyable experience.

Jerker Björkqvist has been a great supervisor, for which I am deeply thankful. Jerker has been instrumental in providing guidance, both in the form of technical discussions and in pushing me in the right direction when necessary.

Together with Jerker, Kristian Nybom has been involved as a co-author in most publications and has been great to work with during these years. Kristian's expertise in error-correction coding and signal processing has been key to the success of my own research. I will certainly miss our highly "creative" meetings in the digital television group of the ESLab. I would also like to thank Juhani Hallio, Jani Auranen, and Reijo Ekman at the radio laboratory at Turku University of Applied Sciences for their valuable collaboration.

I am thankful to Professor Johan Lilius for letting me work at the ESLab during all these years. Both Johan and Jerker have given me the opportunity to work in a wide variety of projects, which has given me valuable exposure to different areas of research.

Professor Riku Jäntti provided valuable feedback on my thesis as my reviewer, for which I am grateful. I would also like to express my gratitude towards Professor Manuel Vélez for both acting as a reviewer and agreeing to act as my opponent.

In general, the ESLab is an amazing place to work, and there is not enough space in this section to thank all the great colleagues I have had the opportunity to work with. I would like to especially thank Johan Ersfolk, Wictor Lund, Simon Holmbacka, and Sébastien Lafond for the interesting discussions and collaboration.

I would also like to thank Marat Vagapov and Dag Ågren for sharing their expertise in electronics and embedded systems. The administrative personnel and IT personnel at the department have been important in making sure that practical matters have been taken care of smoothly. These include Christel

Engblom, Karl Rönnholm, Minna Carla, Joakim Storrank, Tomi Suovuo, Susanne Ramstedt, and several others. For funding my research, I would like to thank TUCS and the Nokia Foundation.

I would like to thank my friends for making these years a much more enjoyable experience. Finally, my parents Ingvor and Björn deserve my deepest gratitude for their neverending support.

List of original publications

Reprinted publications

The following publications are reprinted in Part II of this thesis.

Paper I S. Grönroos, K. Nybom, and J. Björkqvist. "Complexity analysis of software defined DVB-T2 physical layer". In: *Analog Integrated Circuits and Signal Processing* 69.2 (2011), pp. 131–142. ISSN: 1573-1979. DOI: 10.1007/s10470-011-9724-4. URL: http://dx.doi.org/10.1007/s10470-011-9724-4

Paper II S. Grönroos, K. Nybom, and J. Björkqvist. "Efficient GPU and CPU-based LDPC decoders for long codewords". In: *Analog Integrated Circuits and Signal Processing* 73.2 (2012), pp. 583–595. ISSN: 1573-1979. DOI: 10.1007/s10470-012-9895-7. URL: http://dx.doi.org/10.1007/s10470-012-9895-7

Paper III S. Grönroos and J. Björkqvist. "Performance evaluation of LDPC decoding on a general purpose mobile CPU". in: *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. Dec. 2013, pp. 1278–1281. DOI: 10.1109/GlobalSIP.2013.6737142. URL: https://dx.doi.org/10.1109/GlobalSIP.2013.6737142

Paper IV S. Grönroos, K. Nybom, and J. Björkqvist. "Implementation and performance analysis of DVB-T2 rotated constellation demappers on a GPU". in: *Analog Integrated Circuits and Signal Processing* 78.3 (2014), pp. 589–598. ISSN: 1573-1979. DOI: 10.1007/s10470-013-0101-3. URL: http://dx.doi.org/10.1007/s10470-013-0101-3

Paper V S. Grönroos et al. "Distributed Spectrum Sensing Using Low Cost Hardware". In: *Journal of Signal Processing Systems* (2015), pp. 1–13. ISSN: 1939-8018. DOI: 10.1007/s11265-015-1033-1. URL: http://dx.doi.org/10.1007/s11265-015-1033-1

Paper VI S. Grönroos, K. Nybom, and J. Björkqvist. "Synchronization of Low-Cost Distributed Spectrum Sensing Nodes for Multilateration-Based

Geolocation". In: *Proceedings of SDR-WinnComm 2015: Wireless Innovation Conference on Wireless Communications Technologies and Software Defined Radio.* Wireless Innovation Forum Conference on Wireless Communications Technologies and Software Defined Radio. The Wireless Innovation Forum, 2015, pp. 223 –229

Related original publications

The following original publications are not reprinted in Part II, as they have been extended by one of the reprinted publications.

VII S. Grönroos, K. Nybom, and J. Björkqvist. "Complexity Analysis of Software Defined DVB-T2 Physical Layer". In: *Proceedings of the SDR '10 Technical Conference and Product Exposition.* Wireless Innovation Forum, Inc., 2010, pp. 634 –639

VIII S. Grönroos, K. Nybom, and J. Björkqvist. "An Efficient GPU-Based LDPC Decoder for Long Codewords". In: *Proceedings of the SDR 11 Technical Conference and Product Exposition.* Wireless Innovation Forum, Inc., 2011, pp. 272 –279

IX S. Grönroos, K. Nybom, and J. Björkqvist. "DVB-T2 Rotated Constellation Demapping on a GPU". in: *Proceedings of SDR-WinnComm 2013: Wireless Innovation Conference and Product Exposition.* The Wireless Innovation Forum, 2013, pp. 233 –238

X S. Grönroos et al. "Distributed Spectrum Sensing Using Low Cost Hardware". In: *Proceedings of WInnComm-Europe 2014 Wireless Innovation European Conference on Wireless Communications Technologies and Software Defined Radio.* Wireless Innovation Forum, 2014, pp. 1 –8

Contents

List of Acronyms

ADC analog-to-digital converter

API application programming interface

ASIC application-specific integrated circuit

AVX Advanced Vector Extensions

B21C Broadcasting for the 21st Century

BCH Bose-Chaudhuri-Hocquenghem

BER bit error rate

BICM bit interleaved coding & modulation

C-RAN cloud radio access network

CDMA code division multiple access

CGRA coarse-grained reconfigurable array

COTS commercial off-the-shelf

CPU central processing unit

CUDA Compute Unified Device Architecture

D/A digital/analog

DAB Digital Audio Broadcasting

DAC digital-to-analog converter

DARPA Defense Advanced Research Projects Agency

DMA direct memory access

DSA dynamic spectrum access

DSP digital signal processor

DVB-T Digital Video Broadcast - Terrestrial

DVB-T2 Digital Video Broadcast - Terrestrial 2

ESLAB Embedded Systems Laboratory

ETSI European Telecommunications Standards Institute

FEC forward error correction

FFT fast Fourier transform

FIFO first in, first out

FPGA field-programmable gate array

GLONASS Global'naya Navigatsionnaya Sputnikovaya Sistema

GPL General Public License

GPP general purpose processor

GPS Global Positioning System

GPU graphics processing unit

GSE Generic Stream Encapsulation

GSM Global System for Mobile Communications

GuPPI General Purpose PCI Interface

HDTV High Definition Television

I/O input/output

IEEE Institute of Electrical and Electronic Engineers

IF intermediate frequency

JTRS Joint Tactical Radio Systems

LDPC low-density parity-check

LLR log-likelihood ratio

LTE Long-Term Evolution

MISO Multiple Input Single Output

MIT Massachusetts Institute of Technology

MMSE minimum mean square error

MS/s megasamples per second

NFC near-field communication

NLOS non-line-of-sight

OFDM orthogonal frequency-division multiplexing

OpenCL Open Computing Language

OS operating system

OSSIE Open Source SCA Implementation::Embedded

PC personal computer

PCI Peripheral Component Interconnect

PLP physical layer pipe

PPS pulse per second

QAM quadrature amplitude modulation

QPSK quadrature phase shift keying

RBW resolution bandwidth

RF radio frequency

ROM read only memory

SCA Software Communications Architecture

SDR software defined radio

SIMD single instruction, multiple data

SM streaming multiprocessor

SoC system on chip

SSE Streaming SIMD Extensions

TDMP Turbo-Decoding Message-Passing

TDOA time difference of arrival

TI Texas Instruments

TOA time of arrival

USB Universal Serial Bus

USRP Universal Software Radio Peripheral

VLIW very long instruction word

WAM wide area multilateration

Part I

Research Summary

Chapter 1

Introduction

A typical modern mobile phone might need to support 10 or more different wireless communications standards. The phone needs to support four generations of mobile telecommunications networks with their extensions, mid-range data links through various WiFi standards, and short-range communications through several generations of Bluetooth as well as near-field communication (NFC). Satellite navigation using Global Positioning System (GPS), Global'naya Navigatsionnaya Sputnikovaya Sistema (GLONASS), and BeiDou are also often supported in the same device. There might even be an FM-radio receiver inside the handset. The number of wireless communications standards to be supported will only increase, as the fifth generation (5G) of telecommunications networks is being planned.

In the past, most of the signal processing required to support different wireless standards has been handled by application-specific integrated circuits (ASICs) with fixed functionality. As the computational performance of programmable computing platforms continue to increase, however, an increasing number of tasks that have been traditionally handled by dedicated hardware may be replaced by software executing on reconfigurable computing platforms. These reconfigurable platforms include central processing units (CPUs), graphics processing units (GPUs), and digital signal processors (DSPs), as well as reconfigurable hardware such as field-programmable gate arrays (FPGAs). This allows us to move software "closer to the antenna" in radios.

The concept of replacing application-specific circuits with programmable hardware in radio communications is commonly called software defined radio (SDR). In an "ideal" software defined radio, the antenna would be connected almost directly to a programmable processor, with only analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) in between to convert the signal between the analog and digital domains. This concept is illustrated in figure 1.1.

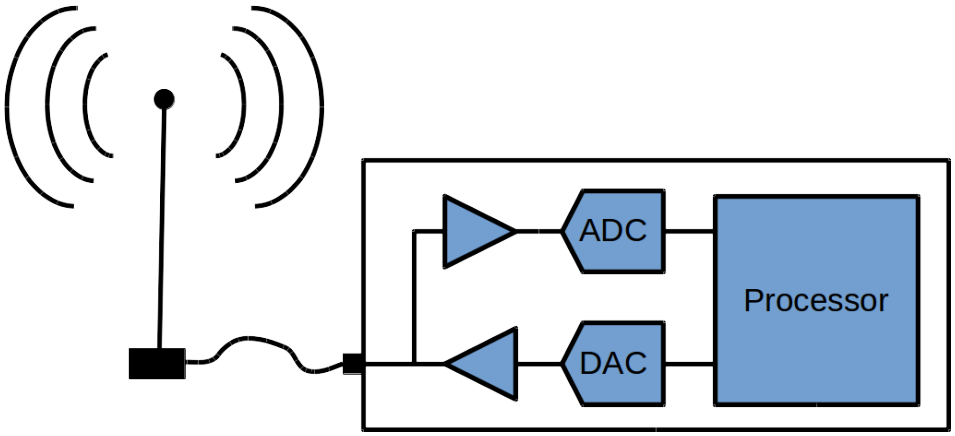

Figure 1.1: An ideal software defined radio, where the antenna is connected almost directly, via an amplification stage, to the analog-to-digital and digital-to-analog converters for reception and transmission, respectively.

There are several significant advantages of defining functionality in software rather than in hardware circuits. Hardware design is generally very expensive, and any significant upgrades generally require a redesign of the hardware. General purpose processors are widely used due to their applicability to a large set of problems, and can therefore enjoy the benefits of mass production, such as lower cost per unit and more advanced production facilities. The general purpose platform can also be reconfigured in a straightforward manner by updating the software being executed, provided that the processing capacity is sufficient for the new functionality. If the new functionality requires more processing capacity, it is often quite easy to adapt the software to a new processor.

Dedicated, single-purpose hardware is typically significantly more energy-efficient when compared to performing the same task in software on a general purpose processor. As the number of features required from the device grows, however, the complexity and size of the ASIC grows.

DSPs are most commonly utilized for high-performance signal processing in software, since their architecture is commonly optimized for processing digitized signals efficiently. General purpose application processors, however, also increasingly contain functionality to enable efficient digital signal and image processing. An example of this is vector processing extensions enabling the application of a single operation on multiple data elements.

GPUs, originally only designed for 2D and 3D graphics acceleration, are increasingly suitable for other tasks that can take advantage of the massively parallel processing capabilities of the GPU. The capabilities of these proces-

sors – not originally intended for signal processing – in an SDR context are explored as part of this thesis.

1.1 Research scope and objectives

The objective of this thesis is to explore the applicability of general purpose processors and low-cost commercial off-the-shelf (COTS) computing platforms to radio applications. The exploration is performed primarily in the form of case studies. Contributions focus on two aspects of SDR using commodity hardware:

1. Feasibility study of implementing components of a modern communications standard in software on commodity hardware.

2. Exploiting COTS hardware to build a low-cost radio spectrum monitoring system.

1.1.1 Efficient SDR algorithms

The first aspect is explored as a series of studies on the Digital Video Broadcast - Terrestrial 2 (DVB-T2) standard for video broadcasting. The DVB-T2 standard is a fairly recent standard with several computationally complex building blocks, making this standard a good test case. First, the complexities of the various building blocks of a DVB-T2 receiver (the transmitter is less interesting due to its lower complexity) are analyzed, and the most computationally complex parts of a receiver are identified.

Based on the initial study, the most computationally intensive building blocks were implemented on both a regular off-the-shelf GPU, and a common CPU, to gain insight into the feasibility of implementing a receiver completely in software. An implementation for CPUs designed for mobile devices was also developed and evaluated.

The goal of this first series of studies is to show that, given efficient algorithms, a completely software-defined implementation of a receiver for a modern telecommunications standard, DVB-T2, is feasible on general purpose hardware.

1.1.2 Low-cost, versatile SDR

One clear advantage of software radios over dedicated hardware is that economies of scale can drive down costs to produce generic radio interfaces and general purpose processors, where the functionality of the system can be customized through software. The second set of studies conducted for this thesis is focused on exploiting low-cost COTS hardware to implement specialized functionality.

This is explored through the construction of a distributed network of radio receivers scanning the radio spectrum for activity, where each node is composed of a very low-cost Raspberry Pi computer and an SDR-capable Universal Serial Bus (USB) dongle intended for the reception of Digital Video Broadcast - Terrestrial (DVB-T) broadcasts. In an extension to this study, it is also demonstrated how – by only replacing the software – this setup can also be used to locate the source of detected radio transmissions geographically.

These studies show how SDR can provide low-cost, versatile alternatives to dedicated systems that are often very expensive due to lower demand and high costs related to developing new hardware.

1.2 Summary of contributions

The contributions of this thesis are discussed in more detail in chapters 4 and 5. A brief summary of the main contributions follow here.

1.2.1 Efficient SDR algorithms

The first series of studies regarding the feasibility of implementing a DVB-T2 receiver in software on commodity hardware encompasses Papers I-IV (see list of original publications). Paper I gives an overview of the complexity of the signal processing blocks in a DVB-T2 transmitter and receiver through analysis of a DVB-T2 simulator. Paper I shows that while the transmitter part has fairly low computational complexity, the receiver contains computationally complex parts. The results show that a DVB-T2 receiver's most computationally complex parts are the low-density parity-check (LDPC) forward error correction (FEC) decoder and quadrature amplitude modulation (QAM) demapper.

Paper II presents a DVB-T2 LDPC decoder implemented on both a GPU and multi-core CPU. The GPU implementation in Paper II shows that real-time LDPC decoding of the codes used in DVB-T2 is feasible on a modern GPU.

Paper III presents an implementation of a similar LDPC decoder on a low-power multi-core ARM CPU. This implementation does not reach real-time performance according the requirements of the DVB-T2 standard. It does, however, provide useful insight into the performance of modern low-power CPUs aimed at the mobile market compared to the more performance oriented desktop computer components.

Furthermore, in Paper IV, the study is completed by presenting a real-time capable DVB-T2 QAM demapper on a GPU. Paper IV also shows that the LDPC decoder and demapper can be executed on the same GPU at a throughput exceeding the DVB-T2 requirements for real-time operation.

Thus, with some reservations further detailed in chapter 4, the studies encompassing Papers I-IV show the feasibility of implementing a receiver for the modern DVB-T2 multimedia broadcasting standard in software on a consumer-grade personal computer (PC).

Author's contributions

The author of this thesis contributed to the first series of studies as follows. The studies in Papers I-IV are based on the analysis of and implementation of optimized signal processing components in a DVB-T2 simulator developed at Åbo Akademi University. The simulator was mainly developed by the co-authors of Papers I-IV. The co-authors assisted with understanding the simulator, as well as writing parts of Papers I-IV. Most parts of Papers I-IV were written by the author of this thesis.

The author performed the performance analysis of signal processing blocks in Paper I independently. The author also implemented and analyzed the optimized LDPC decoders in Papers II-III independently based on less optimized implementations written by co-authors and other contributors to the DVB-T2 simulator. Likewise, the implementation and performance analysis of GPU algorithms in Paper IV was performed by the author.

1.2.2 Low-cost, versatile SDR

The second series of studies exploits mass-produced COTS hardware to build a low-cost radio spectrum monitoring system. This series of studies consists of Papers V-VI. The studies contribute to existing knowledge by demonstrating through field measurements how a very low-cost distributed network of nodes equipped with SDR capabilities can be used for spectrum sensing (Paper V) and also for locating the source of transmissions (Paper VI).

In Paper V the performance of the very low-cost equipment is compared to dedicated spectrum sensing equipment. It is shown that useful measurements can be performed at very low cost, enabling the construction of dense networks of spectrum sensing nodes. Paper VI goes on to show how, through a change in software, the same network of low-cost nodes can also be used to provide coarse location of the source of unknown arbitrary signals.

Author's contributions

For Paper V, the author was responsible for all software implementation and the architecture of the distributed network of spectrum monitoring nodes. The results analysis was also performed by the author. Co-authors wrote much of the background part of the paper, as well as some of the introduction and conclusion. Co-authors also set up and provided measurements from the reference spectrum sensing equipment. Paper VI was mostly written

by the author of this thesis, and all implementation and analysis work was performed by the author.

1.3 Organization of the thesis

The thesis is organized as follows. In chapter 2, we familiarize ourselves with the history of SDR, and the emergence of SDR on commodity consumer hardware. In chapter 3, a brief overview of the features of modern CPUs and GPUs that make these platforms suitable for signal processing tasks is given.

Chapter 4, constains a description of the studies on high-performance SDR algorithms for parts of the DVB-T2 standard. Chapter 5 follows with an overview of low-cost spectrum monitoring and geolocation powered by SDR algorithms. Finally, the thesis is concluded in chapter 6.

Chapter 2

Software Defined Radio

This chapter contains a brief overview of how SDR has evolved over the years. In subsection 2.3, we will especially focus on how SDR platforms have become increasingly available as relatively inexpensive off-the-shelf products that anyone with enough interest can acquire.

2.1 Definition

SDR is defined by the Wireless Innovation Forum and the Institute of Electrical and Electronic Engineers (IEEE) P1900.1 group as "Radio in which some or all of the physical layer functions are software defined" [95, 55]. An "ideal" software defined radio is designed in such a way that the ADC in a receiver or DAC in a transmitter is placed as close to the antenna as possible, after which all further processing is performed digitally in software [78].

In practice, an analog radio frequency (RF) front-end is typically required – or at least desirable – before the DAC/ADC to perform functions such as amplification, filtering, and possibly downconversion of the signal in the analog domain. Since most of the processing is performed in software, the functionality of the radio can be reconfigured simply by running different software.

Often reconfigurable logic devices, such as FPGAs, are also used to realize the reconfigurability, instead of or in addition to software running on a specific processor. In the following section, an overview of the move towards an increased amount of signal processing being performed in software is given. In section 2.3 follows a discussion of the increasing popularity of SDR on commodity hardware such as PCs.

2.2 The early days

SDR is not a new concept, and can be traced back, at least, to the 1980s and mostly to U.S. military projects.

As an example of early designs that would seem to fit the definition of SDR, an implementation of a software modem for transmission and reception of burst messages over VHF voice channels was described in an article published in 1979 by Davies and Davies [19]. The authors implemented most of the signal processing on a Texas Instruments TMS9900 16-bit processor. The advantages of the software approach according to the authors [19] include the use of COTS hardware, and the ability to reconfigure the system simply by switching read only memory (ROM) boards.

In a company newsletter published in May 1985, the E-Systems Garland, Texas division — which was later bought by Raytheon — used the term "Software Radio" to describe their system for analyzing signals captured on tape in software. Their computer system used sixteen Floating Point Systems[1] FPS 5430 Array Processors to perform "full speed" signal processing [80].

2.2.1 SPEAKeasy

The two-phase SPEAKeasy project [66, 17], managed by the Air Force Rome Laboratory in the U.S., and funded by various military branches as well as the Defense Advanced Research Projects Agency (DARPA), was conceived in the late 1980s and officially started in 1992. The project was initiated with the goal of developing an open architecture reprogrammable modem, which would be able to cope with new waveforms as they are developed. The system would also need to be able to handle the myriad of existing waveforms that were in use by both the U.S. military and their allies, and bridge different systems as necessary.

The SPEAKeasy phase 1 radio used four Texas Instruments (TI) TMS320-C40 DSPs as the DSP engine, where various waveforms could be demodulated. The system included a CYPRIS module to handle encryption of information.

In phase 2, which was initiated in 1995, development was continued with especially architectural openness and modularity in mind [17]. The Phase-2 hardware, again, used the same TI DSP chips, however with programmable logic provided by FPGAs. In phase 2, there were also separate cryptographic processors handling security. The phase 2 hardware could handle a frequency range of 2 to 400 MHz, although the original goal was to handle up to 2 GHz frequencies [17].

[1] Floating Point Systems was later acquired by Cray

2.2.2 Joe Mitola

Joe Mitola is regarded as a very influential person in SDR. His initial article on the subject was published in 1993 [78], while Mitola was working at the E-Systems Melpar division in Virginia, USA. In the article, Mitola relates advances in enabling technologies — such as ADCs and DACs, various high performance processor architectures, and interconnects — to the demands of software radios. He also discusses computational models and architectures for software radios.

Mitola mentions [78] that the ideal software radio would be able to communicate using any signal format by simply changing software algorithms, as long as the processing capabilities and ADC/DAC bandwidth requirements are met, and the communication is taking place within a supported frequency range. Mitola also mentions that future software radios could automatically select the optimal transmission parameters for a radio link, given requirements on cost, availability, and signal quality. This concept of a context-aware, reconfigurable radio would later be called *cognitive radio*, and was further explored by J. Mitola and G.Q. Maguire in an article published in 1999 [76].

In 1995, Mitola published an article [77] where he discussed architectural issues in software radio systems. He splits channel processing in a software radio into antenna, RF, intermediate frequency (IF), baseband, and bitstream segments. Based on this segmentation, Mitola discusses the estimation of demand on hardware resources in a software radio in order to meet the often strict timing budgets in such a system.

Recognizing that, especially at the time, a single processor does not have the computational capacity to handle all stages of signal processing on its own, Mitola also discusses the mapping of the various segments onto heterogeneous multiprocessor hardware architectures. He also mentions other tradeoffs that might need to be made in order to overcome hardware limitations or to lower cost, such as offloading some processing to dedicated single-purpose hardware chips.

2.3 SDR on consumer hardware

During the 21th century, powerful platforms for SDR have become more cost effective, which has lead to increased interest in SDR even among hobbyists. Especially SDR systems based around ordinary consumer PCs have become more prevalent, as general purpose CPUs have followed Moore's law and roughly doubled in performance every two years [94], thus making the heavy number crunching involved in processing many waveforms feasible using these processors.

2.3.1 MIT SpectrumWare

In a paper presented at the USENIX '98 conference, Ismert[57] identified two weak points when using a PC for SDR. One was the lack of proper input/output (I/O) devices capable of transferring a high-speed stream of digital samples between the ADC/DAC and the main memory of the PC. The other weakness identified was the lack of operating system support for transporting these continuous streams of data with adequately low jitter through the operating system (OS) kernel to the user-space SDR application for processing.

According to Ismert, jitter in the rate at which an application can process incoming data is caused by features such as virtual memory, cache memories, varying I/O bus utilization levels, and multi-tasking in the OS. The jitter issues are mostly not present in traditional radios based on special purpose circuits, or dedicated DSPs, since in these systems, signal processing often happens at the same rate as the I/O, and no other tasks are allowed to interfere with the processing.

Ismert presented the General Purpose PCI Interface (GuPPI) hardware, which was a device connected to the Peripheral Component Interconnect (PCI) bus of a PC, and which consisted of two boards: one containing the ADC/DAC hardware for digitizing an analog signal, and the other for streaming the samples to the main memory using direct memory access (DMA) transfers. First in, first out (FIFO) buffers on the device were used to absorb jitter due to the bursty nature of PCI bus access. Extensions to the Linux kernel were also implemented to avoid copying of data buffers when moving data between kernel and user space.

The GuPPI hardware was used within the SpectrumWare project at Massachusetts Institute of Technology (MIT). Within this project, a software framework called SPECtRA was developed, providing a number of signal processing modules, a way to connect those modules, as well as a scripting language to define the topology of the radio system [11].

A company, Vanu Inc., that was founded based on research within the MIT SpectrumWare project, commercialized a Global System for Mobile Communications (GSM) base station using ordinary HP Proliant servers based on Intel Pentium processors to perform all of their baseband processing. The base stations could also be updated to the code division multiple access (CDMA) standard after deployment through updated software [65].

2.3.2 GNU Radio and the USRP

Initially based on code from the SpectrumWare project — but completely rewritten in 2004 — GNU Radio [37, 9] was initiated by Eric Blossom and John Gilmore in 1998 [79, 10, 36]. This software has been actively devel-

oped since, and has become a popular framework for various SDR projects, especially among hobbyists and researchers. The GNU Radio framework contains a large amount of pre-written signal processing blocks and handles the interconnection of these blocks, as well as performing the passing of data between blocks and scheduling of the signal processing operations.

The high level structure of a GNU Radio-based application is commonly written in the Python language, while performance-critical signal processing code is written in C++. The project is developed under an open source GNU General Public License (GPL) license. Typically, at first, GNU Radio would be used together with the computer's sound card acting as the digital/analog (D/A) interface to an external front-end, or with rather costly combinations of front-ends, such as cable TV tuner development boards and dedicated D/A boards [27].

Matt Ettus also became an active developer in the GNU Radio project in the early years. Due to the lack of affordable data acquisition hardware suitable for acting as an interface between the antenna and the computer, Ettus designed and sold the Universal Software Radio Peripheral (USRP) [101] hardware. The USRP was based around an Altera Cyclone FPGA and four 12-bit 64 megasamples per second (MS/s) ADCs for reception as well as four 14-bit 128 MS/s DACs for transmission [28]. The FPGA was mostly used for downconversion on the receiver side, as well as interpolation on the transmitter side.

The USRP was designed to hold up to two transmitter and two receiver daughterboards, where the daughterboards contain the analog front-ends required for the frequency range of interest. The USRP connected to a host PC through a USB 2.0 bus. The first USRPs were being shipped by early 2005 for less than USD 500, without daughterboards [26]. Since then, the USRP product range has evolved and expanded, and there are now both USRPs with very high-bandwidth interfaces such as 10 Gigabit Ethernet and PCI Express, as well as battery powered stand-alone devices containing on-board processors [29].

2.3.3 Other frameworks for SDR development

In addition to GNU Radio and other frameworks discussed in earlier sections, several other platforms for the development of SDRs and cognitive radios have been developed.

Iris [99] is a framework for the design of cognitive radio testbeds. To meet this goal, Iris emphasizes runtime reconfigurability in order to enable reacting to changes in the environment [99]. Iris also supports processing platforms such as FPGAs in addition to general purpose CPUs. Created within the EU FP7 SAMURAI project, ASGARD[7] is another platform aimed at providing a testbed for the development of cognitive radios. AS-

GARD is mainly targeting general purpose processors and runs on the Linux operating system.

The Open Source SCA Implementation::Embedded (OSSIE) project[38] is an open source framework for the development of SDRs compliant with the Software Communications Architecture (SCA). The SCA is an architecture for building interoperable communications systems, and was developed within the U.S. military's Joint Tactical Radio Systems (JTRS) project.

2.3.4 The rise of inexpensive SDR equipment

With the gain in popularity of frameworks such as the open-source GNU Radio software framework, a wave of more or less open hardware suitable for use with GNU Radio and similar software has also followed. Many of these initiatives aim to provide low-cost, high-bandwidth platforms featuring RF front-ends and D/A hardware to connect to host computers.

Many of the products in the USRP family have board schematics available, and software drivers are open source. Later projects include the HackRF [39], which in addition to schematics, also releases PCB layout files under open licenses. Several of the more recent SDR hardware interfaces have been funded through "crowdfunding", i.e. through monetary contributions by a large number of people.

Around the year 2010, it was discovered that the demodulation chip of certain DVB-T (Digital Video Broadcast - Terrestrial) USB-based receivers is capable of producing raw 8-bit I/Q samples over the USB in addition to decoding DVB-T on-chip. This capability was originally intended for decoding Digital Audio Broadcasting (DAB) and FM radio. At a price point of roughly USD 20, however, this capability made these dongles highly popular as very low-cost, relatively high-bandwidth SDR receivers [87]. The dongles are capable of reliably streaming samples at a rate of approximately 2.5 MS/s to a host computer in the 20 MHz to 2 GHz range depending on the exact tuner chip used.

Chapter 3

General purpose processors and signal processing

This chapter contains an overview of various general purpose processing architectures, and how they are suitable for digital signal processing. We will mostly focus on the types of processors commonly found in general purpose computing platforms, such as PCs and lower power devices such as mobile phones. These architectures are mostly various CPUs and GPUs.

While DSPs were designed with signal processing in mind, regular CPUs have also become increasingly capable signal processors due to the data parallelism provided by various multimedia extensions to the architecture. GPUs were originally designed for graphics acceleration alone, but have since evolved into massively parallel general purpose processors.

The contributions of this thesis are mostly focused on general purpose CPU and GPU architectures, and thus we will mostly ignore the traditional choices of architecture for SDR: DSPs and FPGAs.

3.1 Central processing units (CPUs)

Popular CPU architectures such as the IA-32 (32-bit) and x86-64 (64-bit) instruction set architectures used in Intel and AMD processors were not designed with signal processing in mind. As these architectures evolved, however, various extensions to the instruction sets have introduced instructions that can accelerate common signal processing tasks.

3.1.1 Vector instructions

MMX and SSE

Acceleration of digital signal processing on general purpose CPUs is often achieved using vector instructions, providing single instruction, multiple data

(SIMD) processing which allows the execution of a single operation on multiple pieces of data in parallel. SIMD was introduced in later revisions of the Intel Pentium processor with the MMX extension [56]. MMX introduced eight 64-bit registers, MM0 to MM7 to the IA-32 architecture. These registers were actually aliased to the lower 64 bits of the 80-bit floating point registers. The 64-bit registers support operations on packed integer data, i.e. eight single byte integers, four 16-bit integers, or two 32-bit integers.

An illustration of the PADDB instruction for the addition of two registers containing 8 packed single-byte integers is shown in figure 3.1.

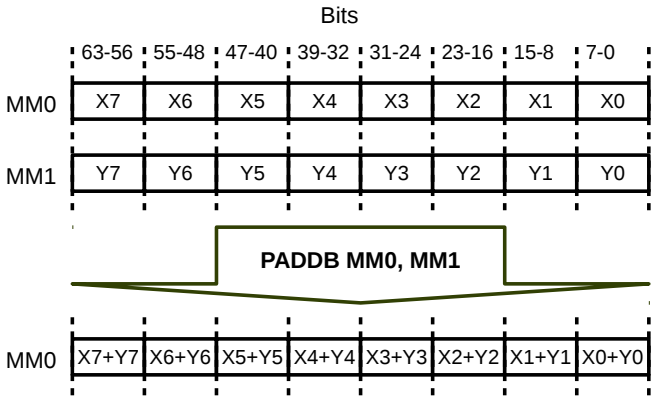

Figure 3.1: An illustration of register contents and the effects of the MMX instruction PADDB for addition of 8 packed byte values. The MM0 and MM1 registers contain the operands. The result is stored in the first operand register.

MMX instructions support addition, subtraction and multiplication arithmetic instructions, where addition and subtraction can also be saturating in case of overflow. There are also comparison, logical, and shift operations, in addition to instructions for packing and unpacking data, as well as loading and storing data from/to memory and other registers.

With the Pentium 3 processor family, Intel introduced Streaming SIMD Extensions (SSE), which provide instructions for computation on packed floating point data, in addition to further integer instructions. SSE provides 128-bit XMM registers for single-precision floating point operations, i.e. operations on four packed floating point values in parallel. Some additional integer operations were also added in SSE, still using the MMX registers. SSE2 also allowed integer operations to use 128-bit registers. SSE3, SSSE3 (Supplemental SSE 3), and SSE4 also added further SIMD operations.

The second generation Intel Core processor family, first released in 2011, introduced the Advanced Vector Extensions (AVX) instruction set, which

expanded SIMD registers to 256 bits for floating point operations. With AVX2, 256-bit registers could also be used for integer operations. With AVX2 we can thus, in ideal conditions, achieve speedups approaching 32 times with 8-bit integer arithmetic, or up to 8 times with single-precision floating point arithmetic compared to using non-SIMD instructions.

ARM NEON

The ARM Cortex-A family of processors may include one or more NEON SIMD (also known as Advanced SIMD) units, which give these ARM processors similar SIMD capabilities as processors equipped with the SSE family of SIMD units. The NEON register set consists of 32 64-bit *doubleword* registers, which can also be viewed as a set of 16 128-bit *quadword* registers. Registers may contain packed integer data of size 8, 16, 32, or 64 bits, or single precision floating-point values. Polynomial types are also supported[6].

The NEON instructions may use quadword registers as operands for 128-bit wide SIMD. Whether the full 128 bits can be processed in parallel or not depends on the instruction, architecture and implementation of the processor.

3.2 Graphics processing units (GPUs)

Graphics accelerators became commonplace in home computers in the late 1990s due to the ever increasing complexity of 3D graphics in computer games. In the early accelerators, the graphics pipeline consisted of fixed-function blocks for processing geometry and pixel data, such as perspective transformations, texture mapping, and the application of various visual effects such as fog.

In the early 2000s, however, some parts of the graphics pipeline became programmable to enable more customized visuals. These graphics accelerators would typically have programmable vertex (coordinate data) and fragment (pixel data) processors that would run programs, called shaders, to be executed on each incoming vertex and fragment separately. For example, a C-like language called the GL Shading Language (GLSL) was used to program these processors in applications using the OpenGL application programming interface (API).

These shader programs, together with techniques for reading back results from the graphics card, for example in the form of a texture, enabled an early opportunity for general purpose programming on a graphics accelerator. Input data could be supplied as "vertices" and texture data, and output data would be read from the graphics card memory as texture data, for example. Algorithms had to be suitable for running as operations on vertices and fragments, and had to be expressed in a limited shader language.

In the mid-2000s, the programmable processors on the graphics card moved towards a unified shading architecture, where vertex and fragment processing was handled by the same processor, leading to even more general GPU cores [93, 20]. As the processing units on the GPU became more applicable to general purpose computing, programming environments that facilitate general purpose programming followed. NVIDIA's proprietary Compute Unified Device Architecture (CUDA) API simplifies the writing of GPU kernels, i.e. functions that are executed on the GPU to process a piece of input data. The same kernel is typically executed in parallel on a large set of input data in parallel. This is very similar to how a fragment shader is applied to many pixels of the output image in parallel, or how a vertex shader is applied to many 3D points in parallel.

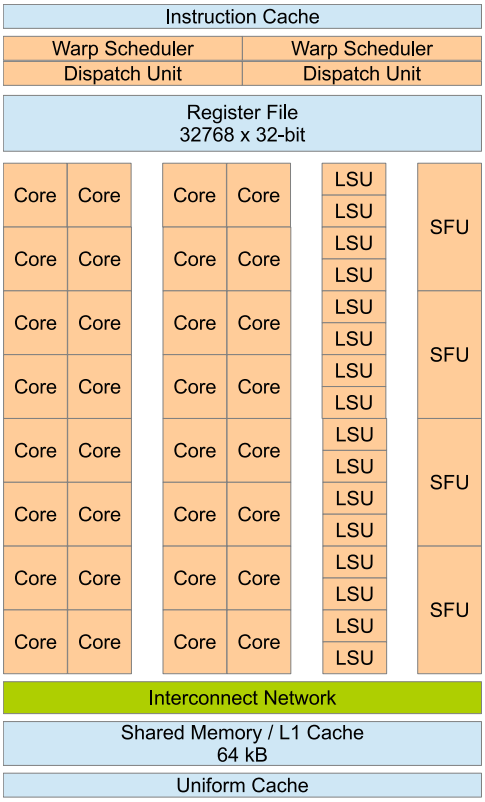

Figure 3.2: A streaming multiprocessor (SM) in the Fermi architecture, as depicted in [84]. There are two warp schedulers per SM, which may schedule a group of 16 threads onto one of two groups of 16 cores. There are also 16 load-store units (LSU), and four special function units (SFU).

Figure 3.2 depicts one streaming multiprocessor (SM) of a fairly modern CUDA-enabled GPU with the Fermi architecture. A typical Fermi GPU might have 1-32 of these multiprocessors. The SM consists of two groups of 16 simple cores each. One such group of cores will execute the same instruction of a kernel at the same time in a SIMD-like fashion. Contrary to a modern CPU, which dedicates large amounts of chip area to cache memories and control logic, the GPU is optimized to process large streams of data in similar ways very quickly. The programming model of GPUs is often called *stream processing*, as they operate on streams of data.

Besides CUDA, there are other frameworks for general purpose GPU programming. The Open Computing Language (OpenCL)[1] is a cross-platform standard for parallel programming on many different architectures: GPUs, CPUs, and beyond. The more recent Vulkan API[2] is aimed at providing a low-level cross-platform API for both general computation and graphics.

3.3 Digital signal processors (DSPs)

While the work described in this thesis is focused on signal processing using CPUs and GPUs, the DSP deserves to be mentioned in this context, since these processors are designed specifically with signal processing tasks in mind. High-performance DSPs are not typically available in general purpose desktop computers or workstations. DSPs are however quite common in mobile devices as part of the main system on chip (SoC), where low power consumption is critical.

Modern DSPs come in a variety of forms [16, 100]. DSPs, such as TI's C6000 series and Qualcomm's Hexagon, are often based on a very long instruction word (VLIW) architecture for executing multiple different operations per clock cycle [100, 16]. While GPUs achieve high performance through massive parallelism, DSPs tend to focus more on single-core performance [3]. Some DSP are also tailored to some specific domain of signal processing, and may come equipped with accelerators for typical signal processing operations such as fast Fourier transforms (FFTs), FEC, etc. Recent DSPs also exploit data parallelism in the form of vector instructions, i.e. SIMD, and may also be multi-core similarly to CPUs.

DSPs can be highly competitive with CPUs and GPUs — especially in terms of energy efficiency — in signal processing tasks. The decision to ignore DSPs in this work was mostly due to the following factors:

- A lack of user-programmable DSPs in most general purpose computers, such as PCs (both consumer and server hardware) and also on several low-power SoCs, such as the popular Samsung Exynos 4 and 5 families

[1] `https://www.khronos.org/opencl/`
[2] `https://www.khronos.org/vulkan/`

and the Broadcom SoC used in the low-cost Raspberry Pi miniature computer. These are widely available platforms that were used for the research in this thesis.

- DSPs are established in the SDR field. We were interested in evaluating alternative, widely available, hardware platforms, i.e. modern multi-core SIMD-capable CPUs and GPUs.

FPGAs are also often used in SDR systems. These platforms, being reconfigurable hardware rather than general purpose processors, do however fall outside the scope of this thesis. In addition, there are various other programmable architectures suitable for SDR applications [4]. These tend to not be widely available in compute platforms, and are thus not covered here.

Chapter 4

Case I: DVB-T2 on general purpose hardware

The first series of studies, encompassing original papers I-IV [43, 45, 40, 46] of this thesis, was aimed at gaining insight into how well the signal processing components of a modern telecommunications standard may be executed on a general purpose computing platform such as a desktop PC.

The Embedded Systems Laboratory (ESLAB) at Åbo Akademi University has been involved in standardization efforts for the DVB-T2 digital television broadcasting standard within the Celtic project Broadcasting for the 21st Century (B21C), and later also for the lightweight profile, DVB-T2 Lite. During these projects, a software DVB-T2 physical layer simulator had been developed [85]. The familiarity with the standard, combined with the existing simulation framework made the DVB-T2 standard a natural choice for further examination.

The physical layer simulator framework was used in papers I-IV, first in order to estimate the complexity of the various parts of the DVB-T2 physical layer, and later as a framework in which optimized algorithms were validated and benchmarked for performance.

4.1 DVB-T2

DVB-T2 is the successor to the DVB-T standard, a standard that is widely used for digital television broadcasting. DVB-T2 was developed to increase efficiency in terms of bit-rate per herz of bandwidth, which would limit the impact of an increasing number of High Definition Television (HDTV) broadcasts on RF spectrum usage.

Some features of DVB-T2 that provide significant improvements in performance over its predecessor include [25, 103]:

- LDPC FEC code for increased error correction performance

- QAM with up to 256-QAM constellations for increased capacity, in addition to quadrature phase shift keying (QPSK), 16-QAM and 64-QAM. Optional rotation of the QAM constellation for increased robustness in some scenarios.

- Increased FFT sizes used by the orthogonal frequency-division multiplexing (OFDM) modulation

- Multiple Input Single Output (MISO)

The work performed with respect to analyzing and optimizing algorithms for the physical layer of DVB-T2 mostly involve the so-called bit interleaved coding & modulation (BICM) module. A brief overview of the architecture of a DVB-T2 modulator is provided here, in order to aid the understanding of which stages are involved in the system.

Data fed into a DVB-T2 modulator are encapsulated as MPEG-2 Transport Streams or Generic Stream Encapsulation (GSE) streams. These streams are first split into one or more physical layer pipes (PLPs), where each PLP may use different coding and modulation. The data is then passed through an input processing module, which converts the input data streams into DVB-T2 baseband frames. After passing through the input processing module, each baseband frame is processed by the Bit Interleaved Coding & Modulation (BICM) module, which contains the following stages (in order):

- FEC coding. DVB-T2 uses an outer Bose-Chaudhuri-Hocquenghem (BCH) code, as well as an inner LDPC code. The resulting FEC blocks can be either 16200 (short code) or 64800 bits (long code) long. 6 different LDPC code rates are available.

- Bit Interleaver (not used for QPSK modulation). Consists of parity bit interleaving, followed by column twist interleaving.

- Mapper, which maps bits onto constellations. Produces cell words.

- Constellation Rotation, if rotated constellations are used. The cell values produced by the mapper are rotated in the complex plane (the angle depends on the modulation used), and the imaginary part is cyclically delayed by one cell.

- Cell Interleaver. Used to uniformly spread the cell words of a FEC block.

- Time Interleaver. In this block, cells of groups of FEC blocks, making up TI-blocks – which in turn make up Interleaving Frames – are interleaved.

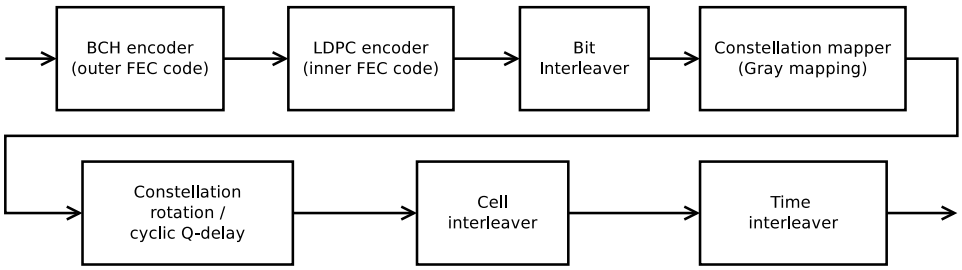

Figure 4.1: Block diagram of the BICM module of a DVB-T2 modulator [25].

The BICM module is also depicted in figure 4.1. The BICM module is followed by the Frame Builder, the task of which is to assemble *T2 frames* from the Interleaving Frames of each PLP, as well as various signaling data. Cells that are going to be included in one OFDM (Orthogonal Frequency-Division Multiplexing) symbol are grouped together in this module. The Frame Builder module also includes frequency interleaving, where the cells belonging to an OFDM symbol are interleaved, providing interleaving in the frequency domain.

The frames produced by the Frame Builder are sent to the OFDM Generation module for further processing. The OFDM Generation module includes the following parts:

- MISO processing. This is optional, and allows for the generation of two slightly different output signals for transmission from two groups of transmitters.
- Pilot Insertion. Cells containing reference information are inserted at to the receiver known points in the transmitted signal. Pilots can be, among other uses, used to aid in synchronization and channel estimation at the receiver.
- IFFT (Inverse Fast Fourier Transform). The OFDM symbols are modulated here. FFT sizes of 1K, 2K, 4K, 8K, 16K, and 32K are supported by the standard.
- PAPR reduction. This optional part allows us to reduce the Peak-to-Average Power Ratio (PAPR) of the transmitted signal.
- Guard interval insertion. This is where we insert guard intervals, which are a cyclic continuation of the useful part of an OFDM symbol.
- P1 symbol insertion. The P1 symbols are special 1K OFDM symbols that are used mainly to aid the receiver in recognizing and tuning in to the DVB-T2 signal.

The output of the OFDM Generation module is a signal ready for transmission. In Paper I, the execution speed of blocks in the BICM, frame builder, as well as OFDM generation stages are analyzed. Results from the analysis drive optimization efforts detailed in Papers II-IV.

4.2 Paper I: Performance analysis of physical layer

In the first paper [43] — an extended conference publication [42] — we profile the software DVB-T2 simulator in order to gain insight into which building blocks of the DVB-T2 standard appear to be the most computationally complex, and whether they meet the throughput requirements for DVB-T2 transmitters and receivers. The DVB-T2 simulator was mostly developed with correctness in mind, and did not exploit features such as multi-threading and vector instructions to increase performance, however the study does still show the relative complexity of the involved signal processing tasks, and provides guidance for further optimization efforts.

Some signal processing blocks that would be necessary in an actual receiver were not available in the simulator. A simulator often assumes perfect receiver-transmitter synchronization, for example, while an actual receiver needs to perform signal processing to detect and synchronize to the transmitted signal. These non-implemented blocks were thus left out of this study.

The study in Paper I indicated that most parts of the modulator (transmitter) executed well above the required throughputs in a DVB-T2 system. The BCH FEC encoder was the only block that did not reach real-time throughput. Other blocks performed at roughly 1.3 to 160 times the real-time requirements. With modest optimization efforts, the transmitter could likely be executed at real-time performance on a regular multi-core CPU.

The demodulator (receiver) side contained several bottlenecks on performance. The LDPC decoder performed at only 2% of real-time requirements in some configurations, and the QAM constellation demapper performed at 8% of real-time requirements in the demanding 256-QAM configuration. These blocks were thus very computationally expensive, and would require further optimization, acceleration, and/or faster algorithms to meet real-time requirements. Other blocks were quite fast at roughly double the required throughput or more. As the demodulator contained the most computationally complex parts, optimization efforts in Papers II-IV were concentrated there.

4.2.1 Author's contributions to Paper I

The idea for the article was conceived by me and the co-authors of the paper. The article is based on analyzing the DVB-T2 physical layer simulator, which was to a large part developed by my co-authors Kristian Nybom and Jerker

Björkqvist. Some simulator blocks were also written by graduate students as part of their theses. I was responsible for performing the analysis and writing the majority of the paper.

4.3 Papers II-III: LDPC decoder

In Paper I, it was found that the LDPC FEC decoder was able to perform at below 10% of the real-time requirements on a desktop PC within the existing software simulator. This finding indicated that the LDPC decoder requires heavy optimization and perhaps acceleration on alternative computing platforms to reach real-time throughput figures while maintaining sufficiently good error correction capabilities.

A GPU-accelerated LDPC-decoder was first implemented for a conference publication [41]. Paper II [45] extends that conference publication with an implementation targeting SSE SIMD-enabled CPUs. For Paper III [40], we created a version of the decoder supporting the NEON SIMD engine often present in high-end ARM processors.

4.3.1 LDPC codes and decoder algorithms

A brief introduction to LDPC codes and the decoding process follows. A binary LDPC code with a codeword length of n, with k information bits and $n-k$ parity bits is defined by a sparse binary $(n-k) \times n$ matrix $\mathbf{H}$. The $\mathbf{H}$ matrix also describes a bipartite graph with n variable nodes and $n-k$ check nodes.

Each "1" in the $\mathbf{H}$ matrix then denotes an edge between a variable node and check node. Thus, column i of $\mathbf{H}$ describes to which check nodes variable node i is connected, and row i describes to which variable nodes check node i is connected. The variable nodes then correspond to the codeword bits, and the check nodes define dependencies between variable nodes (parity). A codeword $\mathbf{x}$ is valid if $\mathbf{H}\hat{\mathbf{x}}^T = \mathbf{0}$ holds.

LDPC decoding using the optimal maximum likelihood decoder is not feasible for useful code sizes [97]. Typically, these codes are decoded using iterative belief propagation algorithms. In these algorithms, and using the so-called flooding message update schedule [64], we have a two-stage procedure. In the first stage, each variable node sends a message to all connected check nodes on its belief of the bit value corresponding to the variable node in question. In the second stage, the check nodes send messages to the variable nodes.

In each stage, we calculate a new message for each edge in the bipartite graph, i.e. for each 1 in $\mathbf{H}$. A hard decision may be performed in the variable node update, where we obtain the current best guess, $\hat{\mathbf{x}}$, of the transmitted codeword. The guess is valid if the equation $\mathbf{H}\hat{\mathbf{x}}^T = \mathbf{0}$ is fulfilled.

If the equation is not fulfilled, we continue iterating the two stages until it is fulfilled, or until we reach a predefined maximum number of iterations.

The LDPC codes specified in the DVB-T2 standard use very long codeword lengths of 16200 or 64800 bits. The best iterative belief propagation decoder algorithm is the *sum-product* decoder [73], which is also, however, quite complex in that it uses computationally costly transcendental functions. Thus, the sum-product decoder is seldom used in decoder implementations. The *min-sum* [108, 15] decoder trades some error correction performance for speed by approximating the complex computations of outgoing messages from the check nodes. The min-sum algorithm was used in all of our optimized LDPC decoder implementations.

The flooding schedule — where all check nodes are updated at once, followed by the variable node update — was chosen due to its inherent parallelism. There are, however, also alternative ways to schedule the message passing. These approaches (such as ones proposed by Mansour and Shanbhag [74]; and Sharon, Litsyn, and Goldberger [96]) tend to serialize the decoding algorithm, and are therefore more difficult to parallelize. They do however tend to require significantly fewer iterations than the flooding schedule to recover a codeword.

4.3.2 Designing a parallel decoder architecture

There is a large amount of inherent parallelism in each stage of the decoder algorithm described in the last section. In the first stage, all n variable nodes can generate messages for their connected check nodes in parallel, while in the second stage, each of the $n - k$ check nodes can generate messages in parallel. Thus, since n and $n - k$ are both in the thousands or tens of thousands, parallelism would not seem to be an issue, even with the hundreds or even thousands of cores found in modern GPUs.

The issue, instead, in LDPC decoding is memory access patterns. Messages between the two stages of the algorithm need to be stored for each edge in the graph. Depending on the order in which messages are stored in memory, at least one stage will require very irregular memory accesses to update messages connected to a single variable or check node, due to the pseudorandom layout of $\mathbf{H}$. This is problematic both on the CPU and GPU platforms, since cache memories can not be used efficiently, and SIMD-like operations become difficult when input and output data is scattered in memory.

Our solution to this issue was to decode up to 128 codewords in one batch, slightly increasing decoding latency while greatly improving memory usage patterns. When decoding several codewords in parallel, for each one in $\mathbf{H}$, we can store the corresponding message for all codewords consecutively in memory. When decoding, we can then perform the exact same operations for each codeword corresponding to a single variable or check node update

in parallel. Thus, we can read and write messages from/to memory for all or part of the codewords in parallel, greatly increasing performance.

Given that we use 8 bits per message for our LDPC decoder, we need to decode at least 16 codewords in parallel on a CPU using SSE to utilize the 128-bit SIMD registers fully. On the GeForce GTX 570 GPU, we have groups of 16 cores executing the same code in lockstep, i.e also here we should decode a minimum of 16 codewords in parallel. The reason for us decoding up to 128 codewords is due to cache line sizes of the various architectures. The GeForce GPU fetches 128 bytes from RAM memory at a time, when the L1 cache is enabled, while the Intel Core i7 CPU used has a cache line size of 64 bytes. In addition, different variable and check node updates may be executed on different cores on the CPU, or on different processors on the GPU (the GeForce GTX 570 has 15 processors containing 32 cores each).

4.3.3 GPU implementation

We targeted the GPU platform for our first optimized LDPC decoder, due to promising results in earlier works [30]. The target hardware was a fairly capable NVIDIA GeForce GTX 570, with 480 simple CUDA cores, and the implementation was written using the CUDA API in the C language. With this implementation we were able to slightly exceed real-time decoding throughput even when running 50 iterations of the min-sum algorithm with a codeword length of 64800 bits.

4.3.4 Intel CPU implementation

The CPU implementation was benchmarked on an Intel Core i7-950 CPU with 4 physical cores, which consist of two logical cores each due to Intel Hyperthreading technology. This implementation used SSE SIMD for parallelism on a single core, i.e. 16 codewords could be decoded in parallel using SIMD instructions. The decoder was also multi-threaded in order to scale to several processor cores. When using all cores, the implementation reached throughputs that were roughly 25% of the obtained GPU throughputs.

4.3.5 ARM CPU implementation

For Paper III [40], we implemented the same LDPC decoder on a Hardkernel ODROID-X development board, which features a Samsung Exynos 4 SoC containing four ARM Cortex-A9 cores. This SoC is found in several mobile phones as well, and served as an indicator of how well software decoding of LDPC would work on a mobile platform compared to a desktop computer system.

Porting the code for the Intel SSE implementation of the decoder was straightforward, as many instructions found in SSE have counterparts in

ARM's NEON SIMD instructions. The performance of the ARM version was inferior to both the GPU and Intel CPU versions, and the throughput on four cores was roughly an order of magnitude worse than the Intel CPU version, and roughly 1/50th of the GPU implementation.

The mobile platform was found to scale significantly worse with an increased number of cores available to the decoder than the Intel CPU version. One experiment showed only a 1.8 times speedup with 4 cores compared to one core on the ARM platform, while performance on the Intel CPU scaled with a multiple greater than 3 when 4 physical cores were available. This is likely due to the limited memory bandwidth and/or cache sizes available on the SoC.

While not in Paper III, the same ARM implementation was later benchmarked on a more recent Hardkernel ODROID-XU platform with the Exynos 5 "octa core" Big.LITTLE architecture consisting of four ARM Cortex-A15 cores and four Cortex-A7 cores. Only one cluster of either four Cortex-A15 cores or four Cortex-A7 cores is active at any one time. This architecture not only showed performance scaling more in line with the Intel processor when adding cores, but also a roughly 3 times increase in throughput performance.

4.3.6 Performance summary and energy efficiency

For this summary, the SSE implementation was also benchmarked on a laptop equipped with an Intel i5-3320M CPU (dual-core, with 4 virtual cores), in order to provide data on performance using a mid-range processor with a stricter power budget compared to the i7-950 performance oriented desktop CPU used in Paper II.

The graph in figure 4.2 summarizes throughputs achieved for decoding 20 iterations of a DVB-T2 LDPC code with code rate 1/2 on all five platforms: GeForce GTX-570 GPU, Intel Core-i7 950 desktop CPU, Intel Core-i5 laptop CPU, Exynos 4 with ARM Cortex-A9, and Exynos 5 with Cortex-A15 and Cortex-A7 cores. For the Exynos 5, performance for the A15 and A7 clusters (all four cores of a cluster were used) are shown separately.

Energy efficiency was not discussed extensively in Papers I-III. Energy efficiency is however interesting in this case, since we have three implementations of an LDPC decoder with approximately the same software architecture across all platforms. Although it is difficult to measure the power consumption of an implementation fairly on such a diverse set of platforms, I will attempt to give some insight into the energy efficiency of the various implementation/platform combinations here.

The power consumption of the desktop computer CPU and GPU implementations was measured by measuring the voltage and current on the voltage rails of the computer's power supply. The laptop power consumption was measured using built-in circuitry for measuring and reporting battery

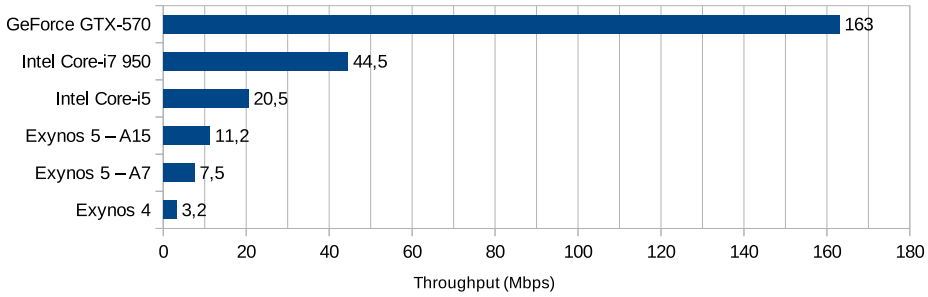

Figure 4.2: Decoder average throughput in Mbps (Megabits per second), long code ($n = 64800$).

discharge rate. It should be noted that the power consumption was measured for the whole laptop, including the screen. The ODROID-X power consumption was measured on the 5 volt input to the board.

Facilities for measuring power of the Cortex-A15, Cortex-A7, memory, and GPU subsystems separately are included in the ODROID-XU board. Here these measurements were added together to measure the full system consumption. The graph in figure 4.3 shows the throughputs divided by peak power consumption during decoding, i.e. Mbps/W for the same platforms listed in figure 4.2.

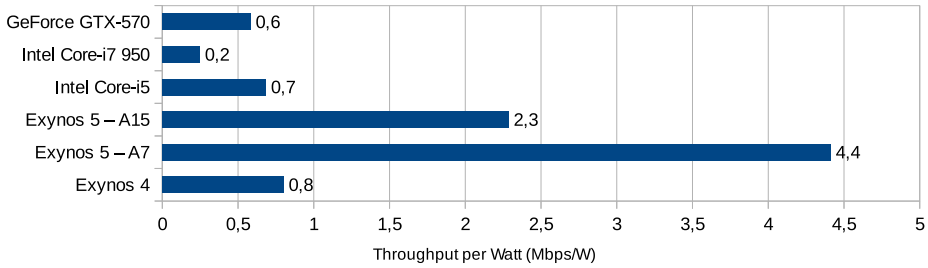

Figure 4.3: Decoder energy efficiency, measured as throughput per watt. Power consumption is the peak consumption during decoding.

We can see that the laptop, the GPU, and the older ODROID-X platform seem to be roughly on par in terms of energy efficiency. The desktop Core-i7 950 processor consumes quite much power in itself, and the surrounding peripherals (motherboard, RAM, disk drives) also count towards the power consumption, making this platform quite inefficient as a whole. While the GPU has the same disadvantages, it delivers enough throughput to rival

the laptop in energy efficiency. The Exynos 5, with higher throughput and similar power consumption to the older Exynos 4, delivers superior energy efficiency at lower throughputs than the desktop and laptop platforms. These results highlight the rapid evolution of processors made for mobile platforms, where performance has increased significantly while retaining similar power consumption.

It is important to stress that an "apples to apples" comparison in terms of energy efficiency is difficult to make across this diverse set of platforms, since throughputs differ by almost two orders of magnitude between the slowest and fastest platforms. The most efficient mobile platforms can not be used in case we require throughputs much above 10 Mbps for example. Conversely, if our requirements are lower than the maximum throughput of the platform, we can often use lower clock frequencies to further increase energy efficiency.

There are several articles on hardware implementation of DVB-S2/T2 LDPC decoders in the literature, however many do not measure power consumption of the design. Urard et al. presented a DVB-S2 LDPC and BCH decoder together with power consumption figures [102]. This enables a rough comparison of energy efficiency. On a chip using 90 nm technology, Urard et al. achieve 135 Mbps throughput at a 300 MHz clock frequency while dissipating 720 to 980 mW depending on which code is used. The authors also quote a best-case energy efficiency of 67 pJ/iteration/Hz. Given the relationship of 135 Mbps/300 MHz, this requires 149 pJ/iteration/bps, which gives a throughput per watt — assuming 20 iterations — of 336 Mbps/W. This is an advantage of roughly two orders of magnitude over the general purpose processor (GPP) implementations presented here, i.e. the reconfigurability of the GPP platforms still comes at quite a high cost.

With mobile GPUs evolving at a rapid pace, evaluating the LDPC decoder on such hardware would also be interesting from an energy efficiency point of view. An implementation suitable for mobile GPUs has not yet been written, however. The current CUDA implementation might be suitable as-is for execution on low-power platforms with a CUDA capable GPU, such as platforms equipped with the NVIDIA Tegra K1 SoC.

4.3.7 Related works

A fairly recent article by Le Gal and Jego contains an overview of various GPU and CPU implementations and their performance as compared to the authors' own implementation [68]. There have been numerous implementations of LDPC decoders on various GPUs over the years [98, 30, 53, 35, 14, 1, 33, 75, 31, 105, 13, 104, 58, 59, 32, 106, 71, 72, 69]. A smaller amount of works have focused on general purpose CPUs, however publications have covered both x86-based CPUs [33, 88, 51, 68], and less common architectures such as the Cell Broadband Engine architecture found in for example

Playstation 3 game consoles [34, 112]. Le Gal and Jego also recently reported on an implementation of their LDPC decoder for ARM processors [67].

LDPC decoder implementations in the literature are not straightforward to compare performance-wise, as the platforms on which they were tested differ widely between publications. The LDPC code types and sizes also differ, with a large part of implementations targeting codes with $n < 5000$ compared to the long codewords used in the DVB-T2 standard. Falcão et al. also published [31] a decoder for DVB-S2 codes — mostly identical to codes in T2 — in 2011 which would appear to perform at roughly the same level of performance as the proposed GPU decoder of Paper II (see Paper II for a comparison). To the author's best knowledge, these implementations belonged to the state of the art in GPU-accelerated LDPC decoders at the time.

The Intel CPU implementation was also, to the author's best knowledge, state-of-the-art at the time of publishing. Performance-oriented implementations for mobile CPUs would not seem to have been reported until recently.

Recently Le Gal and Jego have published a series of articles detailing LDPC decoders for GPUs, x86 CPUs, as well as low-power ARM CPUs [69, 68, 67]. These differ from the norm in that they use a layered or Turbo-Decoding Message-Passing (TDMP) LDPC decoder architecture instead of the more common flooding message update schemes. TDMP-based decoders are more difficult to parallelize than decoders based on flooding message update schemes, such as the implementations described in Papers II and III. Indeed, Le Gal and Jego have opted for the solution where intra-codeword parallelism is not exploited at all. Instead, a larger amount of codewords are decoded in parallel.

For the GPU implementation, this means processing thousands of codewords in parallel to keep the GPU fully utilized, which has the implication that the system needs to buffer a large amount of codewords to send to the GPU. This also leads to fairly high latency in the decoder. On current CPUs, parallelism is much lower than on a GPU, and thus the approach taken by Le Gal and Jego has fewer disadvantages. Indeed, speedups of roughly 10 times compared to the CPU decoder in Paper II is reported [68]. A speedup of two is attributed to the use of the TDMP algorithm which is commonly assumed to require only half the number of iterations for the same decoding performance. The remaining 5x speedup is due to the different approach to parallelism as well as other optimizations. Le Gal and Jego also report a good speedup compared to Paper III for their mobile CPU implementation [67]. Similarly to the findings detailed in section 4.3.6, Le Gal and Jego also reported a large increase in performance between an older ARM Cortex-A9 processor and a more recent A15 processor.

4.3.8 Author's contributions to Papers II-III

The LDPC decoders were implemented within the DVB-T2 simulator, which was developed mostly by Kristian Nybom and Jerker Björkqvist. The existing reference LDPC decoder implementations for the simulator were developed by Kristian Nybom. I was responsible for developing the optimized decoder for both the GPU and Intel CPU in Paper II, as well as the ARM CPU implementation in Paper III. I wrote the majority of papers II and III, and performed the experiments described in these. My co-authors also wrote some of the introduction, background and concluding parts of the papers.

4.4 Paper IV: Constellation demapper

In addition to the channel coding blocks, the QAM constellation demapper of a DVB-T2 receiver is quite computationally complex. The existing simulator block performed at under 10% of real-time requirements when measured in Paper I. The BICM module of a DVB-T2 transmitter is depicted in figure 4.1. After channel coding, i.e. BCH and LDPC coding, and a bit interleaver, a constellation mapper maps bits onto QAM-constellations. The mapper outputs cells, which are coordinates on the complex I-Q plane, where each cell encodes 2 (QPSK) to 8 (256-QAM) bits. The following block, constellation rotation, is optional, and allows the rotation of the constellation diagram in the I-Q plane. Figure 4.4 illustrates a non-rotated and rotated 64-QAM constellation.

If rotation is enabled, the Q component is also cyclically delayed in order to separate it from the I component by one OFDM cell. In the following block, the cell interleaver, the I and Q components are further separated from each other. Since the constellation diagram was rotated by a small angle, each bit sequence will have a unique position in both the I and Q dimensions. Since the I and Q components are subsequently separated, this provides robustness against the loss of one component.

The demapper in the receiver needs to process the incoming cells, corrupted by noise, and produce soft bits, i.e. likelihood values for each bit to be processed by the LDPC decoder. With non-rotated constellations and gray mapping of bits to constellation points, we can treat the I and Q components independently, since each mapped bit position is conveyed by one of the axes. With rotation, this property does not hold, as the components are now correlated, which increases demapper complexity significantly.

As the choice of an efficient algorithm often has a larger impact on performance than code optimization, we first considered which demapping algorithm to use. The optimal, although highly complex, maximum likelihood algorithm for calculating the log-likelihood ratio (LLR) value for bit b_i

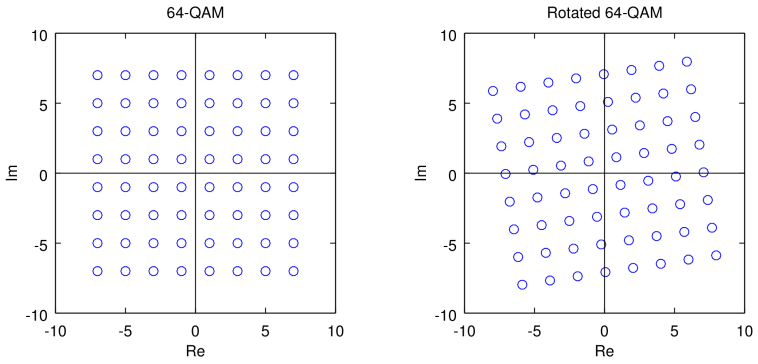

Figure 4.4: DVB-T2 broadcasts may optionally use rotated constellation diagrams together with a cyclic delay of the Q component for extra robustness. A non-rotated and rotated 64-QAM diagram is shown in the figure. Rotation angles depend on QAM order.

($i \in [1, m]$ if we use 2^m-QAM) can be expressed as follows [21]:

$$\begin{aligned} \mathbf{LLR}(b_i) &= \ln\left(\frac{Pr(b_i = 1|\mathbf{r})}{Pr(b_i = 0|\mathbf{r})}\right) \\ &= \ln\left(\frac{\sum_{\mathbf{x}\in C_i^1}(e^{-\frac{D(\mathbf{x})}{2\sigma^2}})}{\sum_{\mathbf{x}\in C_i^0}(e^{-\frac{D(\mathbf{x})}{2\sigma^2}})}\right), \end{aligned} \tag{4.1}$$

where

$$D(\mathbf{x}) = (r_I - \rho_I x_I)^2 + (r_Q - \rho_Q x_Q)^2.$$

Here $\mathbf{r} = \begin{bmatrix} r_I \\ r_Q \end{bmatrix}$ denotes the coordinate of the received OFDM cell in the two-dimensional I-Q plane, ρ_I and ρ_Q denote the amplitude fading factors of the channel, and σ^2 is noise variance. C_i^0 and C_i^1 denote the sets of rotated constellation points for which the i:th bit equals 0 and 1, respectively. Also, note that $\mathbf{x} = \begin{bmatrix} x_I \\ x_Q \end{bmatrix}$ in equation (4.1).

The maximum likelihood algorithm is complex in that it contains transcendental functions, and requires calculating a two-dimensional distance from the received cell to each of the 2^m constellation points for each output bit. This algorithm can be simplified, at the cost of some decoding performance, by applying the max-log approximation:

$$\ln\left(\sum_{i\in[1,n]}(e^{a_i})\right) \approx \max_{i\in[1,n]}(a_i), \tag{4.2}$$

We may also calculate distances to only a subset of the constellation points depending on the received signal. The complexity remains quite high however.

Kim, Bae, and Yang [62] propose performing a minimum mean square error (MMSE) decorrelation followed by interference cancellation (IC) to decorrelate the I and Q components. This is similar to methods commonly used in MIMO (multiple input - multiple output) detectors. Based on the derotated and decorrelated I and Q components, we may perform reduced complexity demapping separately on the two components, similarly to traditional QAM demapping.

In Paper IV [46] — an extension of a conference publication [44] — we implemented and measured the full maximum likelihood algorithm, the max-log approximation, as well as the MMSE-based algorithm [62] with and without interference cancellation. The paper also describes how the demapper and LDPC decoder from Paper II can run efficiently on the same GPU.

4.4.1 GPU implementations

As seen from the architectural overview in figure 4.1, constellation mapping and LDPC coding are separated only by a bit interleaver. It turns out that by interleaving the columns of the LDPC parity check matrix, the bit deinterleaving can be postponed in the demodulator until after the LDPC decoder. This allowed us to implement the GPU demapper in such a way that the LLR values output by the demapper can be directly operated on by the GPU LDPC decoder, avoiding the overhead of transferring samples back to host memory in between. The bit deinterleaver was implemented using lookup tables on the CPU.

Since we implemented the demapper to interface directly to our LDPC decoder, we decoded the OFDM cells corresponding to 128 LDPC codewords (16200 or 64800 bits) in parallel. Each OFDM cell (I/Q sample) decodes to m bits when using 2^m-QAM modulation.

The demapper algorithms fit the GPU architecture well, as each thread on the GPU can operate on one OFDM cell independently. The execution times achieved using the various demapping algorithms can be seen in figure 4.5. We can see that the MMSE-based derotation algorithms provided a massive advantage in terms of execution time compared to the maximum likelihood algorithm, even with the max-log approximation applied. The advantage is especially pronounced with higher order constellations.

The throughput of the combined demapper and LDPC decoder was measured in Paper IV, and it was found that the GPU blocks are able to meet the throughput requirements of the DVB-T2 standard for all modulation schemes when using the MMSE demappers. For 64-QAM and below, even the max-

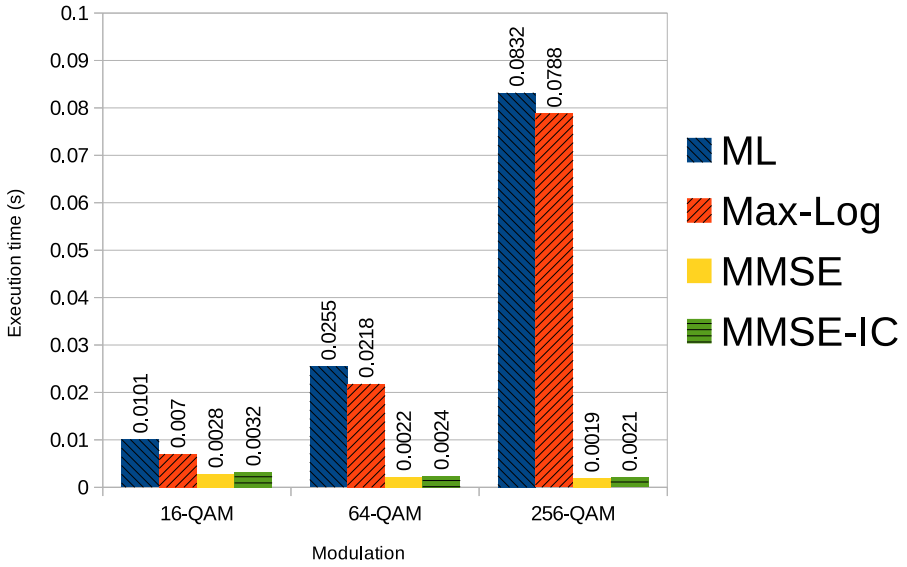

Figure 4.5: Execution times (in seconds) of the various demapper algorithms on the GPU, given 16-QAM, 64-QAM, and 256-QAM modulation schemes.

imum likelihood algorithm exceeded real-time requirements. In terms of demapping accuracy, i.e. bit error rate (BER) performance, there is a penalty for using the MMSE-based approaches. Considering the large gap in execution time, the penalty remained arguably quite small, especially with IC enabled.

4.4.2 Related works

DVB-T2 demapper implementations have been presented to some extent in the literature. To the author's best knowledge, GPU or CPU implementations have not been published, however. In Paper IV, we used the MMSE demappers proposed by Kim, Bae, and Yang [62]. The same group also implemented the MMSE demapper on an in-house coarse-grained reconfigurable array (CGRA) DSP architecture, where the reduced complexity demapper allows running the demapper and several other DVB-T2 blocks on a 600 MHz CGRA [61].

An FPGA implementation of a DVB-T2 demapper is described by Li et al. [70]. The proposed demapper is designed to receive feedback from the LDPC decoder — also implemented on the FPGA — in order to improve BER performance. This kind of feedback mechanism could also perhaps be

implemented in the GPU implementation presented in Paper IV, since both the LDPC decoder and demapper are run back-to-back on the GPU.

4.4.3 Author's contributions to Paper IV

As in Papers I-III, the GPU demapper implementations for Paper IV were implemented within the DVB-T2 simulator written mostly by my co-authors. The ML demapper and Max-Log approximation implementations were based on the existing CPU implementations within the simulator. The GPU implementations were implemented and analyzed by me. The majority of the paper was written by me.

4.5 Case I conclusion and discussion

The main objective of the studies shown in Papers I-IV was to show the feasibility of implementing a recent state of the art telecommunications standard on general purpose computers. At the time of publishing Paper I, the DVB-T2 standard [25] had recently been approved by the European Telecommunications Standards Institute (ETSI). This fresh standard was chosen for the case study due to prior involvement in the standardization efforts.

The preliminary complexity study of a DVB-T2 simulator in Paper I indicated that — at least — the channel decoding and QAM constellation demappers of the simulator were highly computationally complex and would require efficient algorithms and/or implementations to reach real-time throughput on regular consumer PC hardware.

Paper II describes an LDPC decoder implemented on a GPU that satisifies the real-time throughput requirements of the DVB-T2 standard. Paper II also describes a fast LDPC decoder running on a regular Intel CPU, which exceeds the throughput of the original simulator implementation by roughly an order of magnitude. Paper III analyzes the performance of a similar decoder architecture on a low-power ARM processor.

Paper IV goes on to demonstrate a real-time GPU implementation of a DVB-T2 QAM demapper. Paper IV also describes how the LDPC decoder and demapper can run in real-time on the same GPU, and thus free CPU resources for performing the less computationally complex operations in a DVB-T2 demodulator.

The work in Papers I-IV indicates that implementation of a DVB-T2 demodulator appears feasible on modern consumer-grade PC hardware, since Paper I indicates that most of the signal processing in a DVB-T2 demodulator could be performed on one or two CPU cores, while the combined LDPC decoder and demapper can be run on a GPU as shown in Paper IV.

The study has some limitations, however. The main limitations of this series of studies is that some signal processing blocks were not implemented

in the simulator due to being non-essential for simulation purposes. Thus, blocks related to signal acquisition and synchronization were ignored, and the BCH channel decoder was also not present in the simulator.

4.5.1 Related works

This work contributes to evidence in earlier and successive works regarding the feasibility of implementing the physical layers of advanced communications standards on COTS hardware. In the literature, there is further work on DVB standards implemented in software [89, 63, 52]. Pellegrini, Bacci, and Luise [89] described a DVB-T modulator written using the GNU Radio framework. The article was published in 2008, and the implementation was found to perform in real-time even on an entry-level laptop at the time.

Kocks et al. [63] describe a DVB-T2 receiver implementation realized using a mix of a DSP and FPGAs. The signal first passes through a mixed signal processing board consisting of a tuner, a DSP, and an FPGA for filtering and decimation. Next, the samples are passed to a TI TMS32OC6455 DSP running at 1.2 GHz for synchronization and demodulation, after which a Xilinx Virtex-5 FPGA performs FEC decoding. The article does not mention whether rotated QAM constellations are supported, which would likely affect the load on the demodulating DSP.

The software implementation of DVB-C2 described by Hasse and Robert [52] demonstrates a real-time DVB-C2 demodulator implemented in software running on a general purpose Intel processor. Hasse and Robert did simplify the demodulator algorithms to meet real-time requirements. The demapper does not calculate LLR values at all, but instead performs a hard decision on the bit values sent to the LDPC decoder. Also, DVB-C2 [24] does not support rotated constellations, making distance calculations less complex than in the DVB-T2 case. The DVB-C2 standard does however support up to 4096-QAM constellations. The LDPC decoder was implemented using a simple bit-flipping decoder [52].

Another widely used modern communications standard that has received attention from the SDR community is Long-Term Evolution (LTE). Like the commercial software-based GSM base station sold by Vanu Inc., [65], Amarisoft has commercialized an LTE base station running on general purpose computers [2]. Open source efforts, such as the OpenLTE [86] and OpenAirInterface [83, 82] projects also exist.

It would seem quite natural that software-defined cellular base stations have seen the most adoption commercially, since base stations benefit from the flexibility of software and low cost of COTS hardware. The ability to use general purpose server hardware is especially interesting in so-called cloud radio access network (C-RAN) systems [54, 83, 60], where some of the baseband processing of a cellular base station is moved onto virtual machines

in the cloud. This enables the provisioning of computational capacity on demand. Base stations also do not have to meet the same strict power budgets that end-user devices such as handsets and set-top boxes might have to meet.

Modern communications standards, such as DVB-T2, are indeed apparently possible to implement on general purpose PCs today, and perhaps on mobile hardware tomorrow. Software implementations bring many advantages due to their reconfigurability and flexibility. Power consumption of general purpose processors is still a significant issue in many use cases, however.

Chapter 5

Case II: Spectrum Monitoring on Low-Cost Hardware

As mentioned in section 2.3.4, we have recently seen the emergence of simple, very low-cost SDR capable devices such as USB DVB-T receivers capable of fairly high-speed output of raw I/Q-samples. This inspired us to build a low-cost network of receivers for use in spectrum monitoring tasks. The studies included in this thesis, i.e. Papers V and VI, were initial feasibility studies in the utilization of low-cost equipment for RF spectrum monitoring tasks.

For both studies, we used a small network of low-cost receiver nodes. The nodes consist of a small Raspberry Pi computer, which is equipped with a single-core 700 MHz ARM11 CPU and 512 MB of RAM. The SDR receiver is a USB dongle intended for the reception of DVB-T, DAB, and FM radio broadcasts, with the capability of outputting raw 8-bit I/Q samples at approximately 2-2.5 MS/s reliably. The particular tuner chip in the dongle used for our experiments covers a 24 - 1766 MHz frequency range [87]. The family of similar USB dongles, when used for SDR is commonly called RTL-SDR, due to them being based on the RTL2832U DVB-T demodulator chip from Realtek. These Raspberry Pi nodes equipped with RTL-SDR receivers are hereafter called RPi nodes for short.

Paper V, the first study, investigates the feasibility of using this network of nodes to perform distributed spectrum monitoring at a small cost. In Paper VI, we demonstrate how this same network can also be used to locate unknown transmissions geographically, simply by altering the software algorithms run on the nodes.

5.1 Paper V: Distributed Spectrum Sensing

Paper V [49] is an extension of [48]. The objective of both articles was to show how well a low-cost distributed spectrum sensing system could detect signals on air when compared to dedicated spectrum sensing equipment.

Despite the fact that the RFEye Node reference equipment is far superior to the individual RPi nodes in performance, the idea was that a more dense network of low-cost nodes can be built, which in turn can pick up weaker signals close to the receiver. Multiple spectrum sensors in an area also enables the use of data fusion algorithms, i.e. combining the measured spectrum from several nodes in order to, ideally, make better decisions.

5.1.1 Spectrum sensing and experimental setup

Spectrum sensing can be seen as a binary hypothesis testing problem as follows:

$$y(t) = \begin{cases} n(t), & H_0 \\ s(t) + n(t), & H_1, \end{cases}$$

where $y(t)$ is the measured signal, $s(t)$ is an actual signal on air, and $n(t)$ is noise. One simple approach to detecting signals on air is through the use of energy detection. With this method, if the measured signal strength in a frequency bin exceeds a set threshold, we accept H_1 (there is a signal present), and reject it otherwise (there is only noise). In the article, we applied energy detection, using a threshold above an estimated noise floor to reject or accept H_1.

In the article, we analyzed the frequency spectrum between 110 MHz and 1200 MHz. Spectrum sensing on each RPi node in the network was performed by tuning to a 1.25 MHz wide slice of the spectrum at a time, and performing a 32-point FFT on the data to get signal strength values with a 39 kHz resolution bandwidth (RBW). However, 12.5% of data on each end of each 1.25 MHz slice was discarded, due to the attenuation of frequencies furthest from the center frequency.

The CRFS RFEye Node[18] spectrum sensor was used as a reference in the experiments. This sensor is clearly superior to the RTL-SDR-based setup, as the RFEye is capable of scanning a spectrum between 10 MHz and 6 GHz with an instantaneous bandwidth of 20 MHz and a maximum spectrum sweeping speed of 45 GHz/s (however at quite coarse resolution). The RTL-SDR dongle, meanwhile, is restricted to less than 2 GHz of spectrum and an instantaneous bandwidth of roughly 2 MHz. The RFEye Node was set to sweep the same spectrum as the RTL-SDR nodes with a similar RBW.

A central MongoDB[1] document database was used to collect the spectrum sensing data from all RPi nodes, as well as the RFEye Node. A separate computer was then used to perform offline analysis on the collected data.

The nodes of our test network were placed at the authors' homes and at the ICT building in Turku, Finland. An overview map of the network is provided in figure 5.1. The nodes RPi 1-4 were used in the spectrum sensing experiments in Paper V. The reference RFEye device was used for comparison, and connected to the same antenna as RPi node 4.

Figure 5.1: Placement of the Raspberry Pi-based sensor nodes (RPi 1 to 4 in figure), as well as the reference RFeye Node in Turku, Finland [49]. Map data ©OpenStreetMap contributors under ODbL license `http://www.openstreetmap.org/copyright`

5.1.2 Experiments and results

Data fusion strategies when using multiple receivers can be divided into hard decision and soft decision algorithms. With hard decision, we decide between H_0 and H_1 for each frequency bin separately for each participating node.

[1] `http://www.mongodb.org`

The resulting binary decisions are then fused according to voting rules such as AND, OR and fractional rules. For example, with the AND rule, the final decision is H_1 for a frequency bin if and only if H_1 is accepted for each participating node. This rule minimizes false positives since all nodes must agree on the presence of a signal. Conversely, the OR rule only requires one of the participating nodes to exceed the threshold for an H_1 decision. This reduces the number of missed signals at the expense of false alarms. In between the OR and AND rules are fractional rules, where a certain number of nodes have to agree on a decision before we accept H_1.

With soft decisions, we do not make binary decisions for each node, but instead combine the signal strength measurements from each node before making a decision. In the article, we averaged the signal strengths from all participating nodes and set a threshold on this average, above which H_1 is accepted.

Figure 5.2 shows the entire measured spectrum for the RPi nodes, where the peak signal strength of any node for each bin has been captured. Below is the same spectrum as measured by the RFEye Node. From this figure, we see that in general the RFEye has a roughly 10dB advantage in terms of signal-to-noise (see the peaks from DVB broadcasts around 700 MHz for example).

We either averaged the captured signal strengths over a 10 minute time span, or used the maximum peak over the time span when analyzing the results. Averaging will help in discovering always-on signals, while we are more likely to detect bursty signals by retaining the maximum peak, at the cost of also retaining a higher noise floor.

During analysis, the binary hypothesis decisions made for the RPi nodes and for the RFEye Node tended to match for roughly 90-95% of frequencies in many cases. Decisions matched the least when peaks were retained, likely due to bursty transmissions being detected by one node, but not the other. Conversely, the best agreement could be seen when signals were averaged, to remove noise and bursts. RPi nodes 3 and 4 were also the best matching nodes to the RFEye Node, which is to be expected as these nodes were placed on the same building as the RFEye Node. RPi 4 was even connected to the same high-quality antenna as the reference equipment.

The article was mostly written to give insight into the possibility of sensing spectrum using very low cost sensors. We resorted to quite extensive manual tuning of parameters such as decision thresholds, receiver gains and so on. Parameter selection should ideally be automated, however. Automatic tuning of parameters is very important in the envisioned case, where many nodes are used, and placed in varying conditions connected to antennas of varying quality. In such a case, suitable data fusion rules need to be employed in order to minimize the impact of malfunctioning or even rogue nodes in the network, while still enabling the detection of local signals.

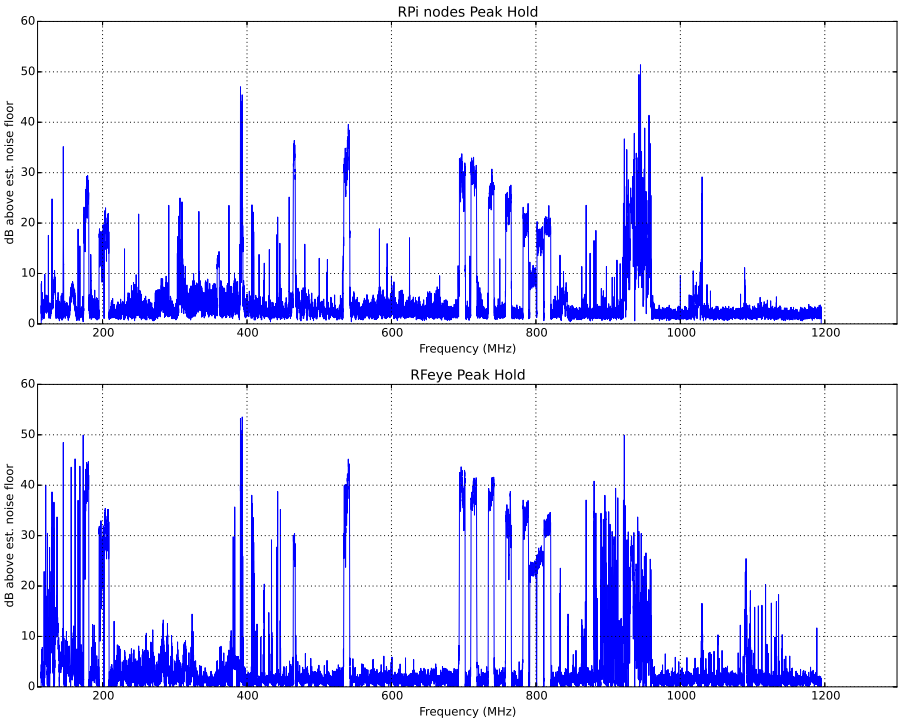

Figure 5.2: An overview of the captured spectrum from the RPi nodes, as well as from the RFeye reference node. The data from the RPi nodes has been combined by retaining the maximum value over time and over all nodes (peak hold) [49].

5.1.3 Related works

Nika et al. [81] considered the use of a mobile phone equipped with the same type of DVB-T dongle for spectrum sensing. The performance of the setup in terms of e.g. sensitivity and signal detection accuracy is compared to a PC equipped with the same dongle as well as an Ettus Research USRP device for comparison.

The idea is to equip ordinary mobile phones with spectrum sensing capabilities and use crowd-sourcing to collect large amounts of measurements. That goal is similar to the idea in Paper V, however the approaches differ. Measurement methods differ in that we used field-measurements, while Nika et al. [81] conducted experiments using controlled transmissions in an empty spectrum band. Nika et al. also did not perform co-operative sensing in the

article [81], but concentrated efforts on characterizing the performance of single devices.

Similar work to ours was made by Pfammatter, Giustiniano, and Lenders [91], where spectrum sensing using the same dongle type is considered to enable collaborative spectrum sensing. The work analyzes various strategies for "hopping" through the spectrum, and suggests methods for compensating for frequency offset due to the oscillator used in the dongle. Pfammatter et al. also propose data compression for the data sent from the sensing nodes to reduce network bandwidth use. Performance comparisons are made to a USRP device.

The experiments in Paper V are focused on building a demonstration network of nodes and performing field measurements, while Pfammatter et al. perform a more in-depth analysis of the various characteristics of the dongle, as well as analysis of the impact of hopping strategies on detection of different kinds of signals. Pfammatter et al. draw similar conclusions to ours: "While our sensor architecture runs on limited hardware with less accurate measurement capabilities than higher-end spectrum analyzers, we believe such a setup to be well suited to commoditize and crowdsource the task of wideband spectrum monitoring in the future." [91]. They also similarly envision that a dense network of sensors may compensate for the less accurate sensing of a single node.

Another low-cost spectrum sensing system is described by Arcia-Moret, Pietrosemoli, and Zennaro [5], where TV spectrum is monitored using a Raspberry Pi computer connected to an RF Explorer (120 USD) handheld RF spectrum analyzer.

To the author's knowledge there are quite few field-measurement studies in cooperative spectrum sensing. One such study is [107], where co-operative spectrum sensing techniques were evaluated during field measurements in Germany using professional RF spectrum analyzers. The study aimed to confirm theoretical findings related to co-operative spectrum sensing, while we focus on the feasibility of very low-cost sensing networks.

5.1.4 Author's contributions to Paper V

For Paper V, I was responsible for the implementation of the Raspberry Pi-based sensing network, and the software for collecting and analyzing results. I also wrote the parts of the paper concerning experimental setup and results analysis. My co-authors Kristian Nybom and Jerker Björkqvist performed much of the literature review for the paper. My co-authors at Turku University of Applied Sciences participated in planning for the paper, and were responsible for setting up and extracting measurement data from the RFEye reference equipment.

5.2 Paper VI: Distributed Low-Cost Geolocation

In Paper V, we explored the use of a network of connected Raspberry Pi nodes with SDR-capable USB dongles (RPi nodes for short) to perform distributed spectrum sensing. We also wished to examine the possibility of using the same network to locate unknown RF emitters geographically. This feasibility study was performed in Paper VI [47].

5.2.1 Multilateration

An unknown transmitter can be located, through the process of multilateration, by measuring the differences in arrival time of a signal to a set of synchronized receivers. The process of multilateration, where a plurality of receivers locate a single transmitter, is for example used in wide area multilateration (WAM) systems. These systems locate aircraft using the aircraft's own reported altitude together with the time differences of arrival (TDOAs) of the transponder signal to at least three ground stations.

Ideally, if the receivers are synchronized, the measured TDOA is only a function of the difference in distance between the emitter and a pair of receivers (with the signal traveling at the speed of light). Each measured TDOA yields an infinite number of possible locations along a hyperbola in 2D space, or a hyperboloid in 3D space. Thus we typically need a minimum of three (2D space) or four (3D space) TDOAs to find the unknown emitter in the ideal, noiseless, case.

5.2.2 Receiver synchronization

For Paper VI, we built an initial prototype of a system consisting of the RPi nodes 1-3 in figure 5.1 for measuring the signals of interest, and a separate computer that receives sampled signals, and calculates TDOAs. In TDOA-based multilateration, it is important that all nodes are synchronized to some global reference. Without such a reference, we will be unable to align the sample streams from each node, and will thus not be able to obtain delays resulting from TDOAs.

Geographically distributed receivers are commonly synchronized using the time signal from GPS satellites. We considered this option, however a few issues made us seek alternative solutions.

The first issue is how to accurately time-stamp the sample stream from the RTL-SDR dongle. Synchronizing the clock of the Raspberry Pi using GPS would not be sufficient, since the data buses have unknown and varying delays, making it very difficult to know the time of acquisition of a sample. One possible solution to this could have been to inject a GPS disciplined periodic signal — for example the pulse per second (PPS) output of many GPS receivers — directly into the antenna input of the dongle.

This leads us to another issue with using GPS synchronization: cost and complexity. A GPS receiver adds to the cost of the system. Extracting the timing information would likely require hardware modifications to the otherwise COTS hardware, increasing complexity and cost. The GPS receiver also needs to be placed appropriately to "see" enough satellites for good performance.

An alternative to GPS synchronization is to synchronize to signals of opportunity with a known origin, such as public TV and radio broadcasts. Especially television broadcasts tend to be strong, wide-band, and ever present signals that would be suitable for use as references. Experiments revealed that tuning the RTL-SDR dongle to a new frequency while sampling does not seem to interrupt the sample stream from the ADC. This makes it possible to first sample a number of samples from the known reference frequency, after which we can tune to the unknown signal of interest and obtain another set of samples.

5.2.3 The implementation

Let $\mathbf{v}_i$ be the samples from the reference transmitter obtained at node i. Similarly $\mathbf{u}_i$ are the samples collected when tuned to the frequency of interest. The initial delay δ_i for node i is the initial delay between the time at which we wish to start sampling and when the node actually starts sampling. This delay is caused by unsynchronized receivers and delays in processing, buses, and network links.

If we cross-correlate the samples from the reference frequency between two nodes, we (ideally) obtain τ_{v_i,v_j}, the delay in samples between the streams. This delay includes the delay $\delta_i - \delta_j$ caused by initiating the sampling process at different times (typically on the order of 10-100 milliseconds), as well as the delay resulting from the TDOA from the reference transmitter (typically on the order of microseconds). By cross-correlating the samples from the transmitter of interest, we obtain a delay τ_{u_i,u_j} that is a function of the same $\delta_i - \delta_j$, as well as the TDOA from the unknown transmitter. In the ideal case, the following should hold:

$$\begin{aligned}\tau_{v_i,v_j} &\equiv (\frac{F_s}{c}(\|\mathbf{p}_i - \mathbf{q}_c\|) + \delta_i) - (\frac{F_s}{c}(\|\mathbf{p}_j - \mathbf{q}_c\|) + \delta_j) \\ &= \frac{F_s}{c}(\|\mathbf{p}_i - \mathbf{q}_c\| - \|\mathbf{p}_j - \mathbf{q}_c\|) + (\delta_i - \delta_j),\end{aligned}$$

with F_s being the sampling frequency, and c the speed of light. Node i is located at coordinates $\mathbf{p}_i$, and the known reference transmitter is at $\mathbf{q}_c$. Similarly when tuned to the transmitter of interest, we have:

$$\tau_{u_i,u_j} \equiv \frac{F_s}{c}(\|\mathbf{p}_i - \mathbf{q}_t\| - \|\mathbf{p}_j - \mathbf{q}_t\|) + (\delta_i - \delta_j),$$

where $\mathbf{q}_t$ is the unknown location of the transmitter of interest.

Thus, again in the ideal case, we can eliminate the unknown $(\delta_i - \delta_j)$ term, and get an estimation on the difference in TDOA between a pair of nodes and the two transmitters as follows:

$$\tau_{v_i,v_j} - \tau_{u_i,u_j} = \frac{F_s}{c}\Big((\|\mathbf{p}_i - \mathbf{q}_c\| - \|\mathbf{p}_j - \mathbf{q}_c\|) - (\|\mathbf{p}_i - \mathbf{q}_t\| - \|\mathbf{p}_j - \mathbf{q}_t\|)\Big),$$

or equivalently

$$\|\mathbf{p}_i - \mathbf{q}_t\| - \|\mathbf{p}_j - \mathbf{q}_t\| = \|\mathbf{p}_i - \mathbf{q}_c\| - \|\mathbf{p}_j - \mathbf{q}_c\| - \frac{c}{F_s}(\tau_{v_i,v_j} - \tau_{u_i,u_j}),$$

where the TDOA of interest is the left hand side of the equation. The right hand side can be calculated, as we know the distances between our measurement nodes and the reference transmitter.

The RTL-SDR dongle does however not have a very stable clock, and this causes various issues in practice. For example, we compensate for frequency offset between receivers by applying a set of frequency shifts to the samples from one receiver to find the shift that yields the highest correlation peak. A more detailed description of the algorithm and techniques used to mitigate the inaccuracies of the receivers can be found in Paper VI.

5.2.4 Results and Discussion

Depending on which type of transmitter was used as a reference, and as a target, we observed quite different performance in terms of distance errors. We tested the system using FM radio and DVB-T television broadcasts from two transmitter sites in the Turku region.

The worst accuracy was achieved when attempting to locate an FM radio transmitter using an FM transmitter as a reference. Conversely, the smallest errors were encountered when using DVB-T signals as both reference and target. This is to be expected, since broadband signals correlate better than narrowband signals. FM radio signals are also more affected by which content is on-air at the moment, than digital TV signals.

Generally an accuracy of roughly 100 m was achieved with DVB-T signals, while up to 1 km errors were encountered in some configurations. The system may still be useful for rough location of the origin of an unknown transmission.

Challenges include mitigating skewed distance estimations due to non-line-of-sight (NLOS) conditions, i.e. where we correlate against a reflected signal. The current implementation also requires a constant transmission that can be heard by all participating nodes. For bursty signals, a mechanism to trigger the sampling when a burst occurs would be necessary.

5.2.5 Related works

As mentioned, the use of TDOA-based methods for positioning using RF signals is not new, and is used in commercially deployed systems. As an illustrative example, the use of signals of opportunity — CDMA cellular communications more precisely — for vehicle positioning was discussed by Caffery and Stuber [12] in 1994. The purpose of the study in Paper VI was thus not to show that this kind of positioning is possible, but rather to demonstrate how it is now feasible to implement using low-cost COTS hardware where the functionality is entirely defined by software executing on a low-end general purpose applications processor.

The use of low-cost hardware for locationing purposes has been explored in a number of publications. El Gemayel et al. [22] set up a network of low-cost sensors using Ettus Research USRP hardware in each node. TDOA measurements are used to calculate the location of unknown transmitters. The USRP hardware is however significantly more expensive than the hardware setup used in this work. El Gemayel et al. also used GPS synchronization, while in this work, we synchronize to arbitrary known signals.

Navigation using non-GPS signals of opportunity such as TV and radio broadcasts as well as cellular network signals has also been discussed in the literature [109, 110, 92, 23, 90]. As with studies in cooperative spectrum sensing, most works appear to focus on simulation results and lab measurements, while Paper VI explores location using signals of opportunity in a deployed network of nodes and with real-world broadcast signals.

The system presented by Rabinowitz and Spilker [92] uses timing information obtained from ATSC (Advanced Television Systems Committee) digital television signals for geolocation both indoors and outdoors. Field measurements showed promising results also in locations with poor GPS coverage [92].

Peral-Rosado et al. [90] describe a TDOA-based locationing system which locates the receiver using LTE signals. The system is tested on both an RTL-SDR dongle as well as a USRP connected to stable reference clock. The system exploits the LTE signal structure for tracking frequency shift and time delay, enabling positioning when signals from several LTE base stations are tracked. An accuracy of 5-10m is reported for a simulated LTE network. The simulations do not take multipath propagation into account, i.e. line of sight is assumed [90]. While we aimed for a more waveform-independent approach in Paper VI, exploiting the characteristics of specific waveforms can certainly aid significantly in increasing the accuracy of positioning. One could also take a hybrid approach, where identified waveforms are exploited for more accurate positioning, while unknown waveforms are simply correlated as in Paper VI.

Yan and Fan [109] propose a system where receivers do not know their position beforehand, and cooperatively locate themselves using signals of opportunity such as television and AM radio signals. The differential TDOA approach described is designed to eliminate the need for synchronous receiver clocks, and takes into account clock drift between receivers. The article only presents simulations, and not field measurements, however.

The idea of a large network of cooperating nodes for navigation is also discussed by Enright and Kurby [23]. Through simulations, the suitability of WCDMA (Wideband Code Division Multiple Access), digital television and GSM (Global System for Mobile Communications) signals for navigation were investigated [23]. Time of arrival (TOA) and TDOA-based approaches are also among the many approaches used in indoor positioning, where GPS signals can not be relied on [50].

5.2.6 Author's contributions to Paper VI

The idea for the paper was conceived during meetings with my co-authors. I was responsible for implementing the software, as well as analyzing the results. My co-authors wrote some of the introductory and concluding sections of the paper, while I wrote sections regarding experimental setup and results analysis.

5.3 Case II conclusion and discussion

Papers V and VI explore the use of a network of very low cost nodes for spectrum monitoring. The nodes are low cost due to them consisting entirely of mass-produced COTS hardware in the form of a Raspberry Pi ARM-based computer, and a DVB-T receiver with the capability of outputting raw complex I/Q samples.

Paper V explores the use of this low cost network for spectrum sensing. Paper VI extends the functionality of the sensing network with the necessary synchronization of receivers to perform TDOA measurements for geolocation of unknown signals.

The initial study on spectrum sensing in Paper V compared the distributed low-cost network to professional equipment in the form of an RFEye Node spectrum sensing node. The low-cost setup is capable of giving useful data on spectrum utilization in that the same signals are detected with high probability compared to the reference equipment.

As can be expected however, the low-cost nodes do have significant drawbacks, such as orders of magnitude slower scanning speed, lower sensitivity and a narrower frequency range. The low-cost design also means that internal noise from for example the oscillator shows up in the sampled signal. Also, the antenna quality and placement has a profound impact on the quality

of measurements. The price of a good antenna alone may thus significantly exceed the price of the rest of the node.

The entirely software-based approach to spectrum monitoring in Papers V-VI could enable crowd-sourcing of data however, as all that is required is access to raw digitized samples from a radio frontend and a relatively low-end processor to perform the signal processing required. Perhaps the best way to achieve a dense network of spectrum sensors would be to build this kind of monitoring technology into devices such as various wireless access points, base stations and perhaps even mobile handsets. In these cases the required wide-band antenna is still an issue, however.

Spectrum sensing is often one component in systems for dynamic spectrum access (DSA), where licensed but unused spectrum can be used by secondary users on demand to increase radio spectrum utilization. The low-cost hardware described in Paper V might not be very useful for accurately measuring whether a certain spectrum band is in use at the moment or not, due to the low scanning speed. The general direction in these kinds of systems appears to be towards maintaining databases containing spectrum allocations rather than depending on spectrum sensing [111]. A distributed spectrum sensing system can still provide useful information for example in the form of radio environmental maps [8].

While not clearly demonstrated in the initial study of Paper V, Paper VI shows how we overcome some limitations of the low-cost hardware by compensating for them in software. This approach should also be applied to the spectrum sensing part of the system, i.e. one could perhaps compensate for the flaws in the hardware by using improved algorithms to detect signals buried under noise, and for example by scanning the spectrum more intelligently to compensate for the limited scanning speed.

Chapter 6

Conclusions and Future Work

During my doctoral studies, I have been involved in quite a wide variety of projects, ranging from wireless communications and video coding to ubiquitous computing and e-health. Although the subject of this thesis has been narrowed down to only cover SDR-related topics — which have been the core of my research — the topic of this thesis is nevertheless also quite broad.

The first series of papers, Papers I-IV [43, 45, 40, 46], aimed at analyzing a modern communications standard to assess the feasibility of executing the processing required for the physical layer completely in software on general purpose, COTS computing platforms. We analyzed the complexity of the components of the DVB-T2 physical layer and implemented the computationally complex LDPC decoder and rotated QAM demapper on a GPU to reach throughputs required to meet the DVB-T2 requirements for real-time operation.

The LDPC decoder was further implemented on alternative platforms: an Intel multi-core high-end consumer CPU and two models of CPUs commonly found in mobile phones. The CPU implementations of the LDPC decoder did not reach the throughput required to broadcast a full 8 MHz DVB-T2 channel multiplex. These studies did provide insight into the differences in performance between a typical GPU, desktop CPU, and mobile CPUs for the memory intensive, but parallel workload that is LDPC decoding. Furthermore, some interest in the LDPC decoder implementations from industry led to the release of the SSE-optimized CPU decoder as an open-source software library[1].

While the optimization of the DVB-T2 physical layer exploited rather high-end, high-power consumer platforms, the second set of studies explored low-end, low-power SDR for spectrum monitoring tasks. The change of focus reflects the growing importance of monitoring usage of the RF spectrum in times when dynamic spectrum access — i.e. sharing and allocation of unused

[1] `https://github.com/stippeng/ldpc`

spectrum based on demand — is considered in order to combat spectrum underutilization. When RF bands are allocated dynamically by area and time, spectrum monitoring can aid the allocation of spectrum such that interference to other spectrum users is minimized.

Our vision for the series of studies starting with Paper V [49] is to be able the use many low-cost sensors to complement, or even replace, more expensive dedicated spectrum sensing equipment. The rationale is that a more dense network can provide more localized measurements. Very low-cost spectrum monitoring capabilities could perhaps be built into consumer electronics such as mobile phones and broadband routers, or lead to crowd-sourced spectrum monitoring through other means.

Papers V and VI [49, 47] are only initial feasibility studies into the very low-cost spectrum monitoring field. Perhaps the most prominent contribution of Paper V is to show that we can obtain useful spectrum utilization data from a hardware setup that costs around USD 50 per node, and that the measured data matches the data collected using reference spectrum sensing equipment to quite a high degree. By replacing the software run on the spectrum sensing nodes, we showed in Paper VI that rough geolocation of unknown radio transmitters can be performed with the same low-cost hardware setup.

Papers V and VI highlight a key strength of SDR on general purpose hardware. We were able to utilize mass-produced, and thus low-cost hardware that was never intended for use in spectrum monitoring to build our low-cost spectrum sensors.

6.1 Future work

The low-cost spectrum monitoring track provides many opportunities for further work, both concerning spectrum sensing — i.e. detecting activity in the RF spectrum — and concerning the geolocation of RF emitters. In spectrum sensing, more sophisticated detection algorithms than pure energy detection could be utilized to detect weak signals that are easily mistaken for noise. Methods for detecting bursty signals using the low-cost equipment should also be explored further. The precise synchronization of the nodes as described in Paper VI could perhaps be used to provide more advanced collaborative spectrum sensing capabilities outside geolocation as well.

Bibliography

[1] K. Abburi. "A Scalable LDPC Decoder on GPU". In: *VLSI Design (VLSI Design), 2011 24th International Conference on*. Jan. 2011, pp. 183–188. DOI: 10.1109/VLSID.2011.44.

[2] Amarisoft. *Amarisoft website*. http://amarisoft.com [Online; accessed 13.06.16]. 2016.

[3] AnandTech. *Qualcomm Details Hexagon 680 DSP in Snapdragon 820: Accelerated Imaging*. News website, http://www.anandtech.com/show/9552/qualcomm-details-hexagon-680-dsp-in-snapdragon-820-accelerated-imaging. Accessed 20.6.2016. 2010.

[4] O. Anjum et al. "State of the art baseband DSP platforms for Software Defined Radio: A survey". In: *EURASIP Journal on Wireless Communications and Networking* 2011.1 (2011), pp. 1–19. ISSN: 1687-1499. DOI: 10.1186/1687-1499-2011-5. URL: http://dx.doi.org/10.1186/1687-1499-2011-5.

[5] A. Arcia-Moret, E. Pietrosemoli, and M. Zennaro. "WhispPi: White space monitoring with Raspberry Pi". In: *Global Information Infrastructure Symposium, 2013*. Oct. 2013, pp. 1–6. DOI: 10.1109/GIIS.2013.6684374.

[6] ARM. *ARM Architecture Reference Manual: ARMv7-A and ARMv7-R edition (Issue C.c)*. Manual, http://infocenter.arm.com. 2014.

[7] ASGARD. *The ASGARD software radio*. http://blog.asgard.lab.es.aau.dk [Online; accessed 19.06.16]. 2016.

[8] V. Atanasovski et al. "Constructing radio environment maps with heterogeneous spectrum sensors". In: *New Frontiers in Dynamic Spectrum Access Networks (DySPAN), 2011 IEEE Symposium on*. May 2011, pp. 660–661. DOI: 10.1109/DYSPAN.2011.5936266.

[9] E. Blossom. "GNU radio: tools for exploring the radio frequency spectrum". In: *Linux Journal* (122 June 2004). ISSN: 1075-3583.

[10] E. Blossom. *Re: wikipedia entry*. Discuss-gnuradio mailing list correspondence; https://lists.gnu.org/archive/html/discuss-gnuradio/2008-09/msg00130.html [Online; accessed 4.8.2014]. Sept. 2008.

[11] V. Bose et al. "Virtual radios". In: *Selected Areas in Communications, IEEE Journal on* 17.4 (Apr. 1999), pp. 591–602. ISSN: 0733-8716. DOI: 10.1109/49.761038.

[12] J. J. Caffery and G. L. Stuber. "Vehicle location and tracking for IVHS in CDMA microcells". In: *Personal, Indoor and Mobile Radio Communications, 1994. Wireless Networks - Catching the Mobile Future., 5th IEEE International Symposium on*. Vol. 4. Sept. 1994, 1227–1231 vol.4. DOI: 10.1109/WNCMF.1994.529449.

[13] C.-C. Chang et al. "Accelerating Regular LDPC Code Decoders on GPUs". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 4.3 (Sept. 2011), pp. 653–659. ISSN: 1939-1404, 2151-1535. DOI: 10.1109/JSTARS.2011.2142295.

[14] Y.-L. Chang et al. "High-throughput GPU-based LDPC decoding". In: ed. by B. Huang et al. Aug. 2010, p. 781008. DOI: 10.1117/12.862716.

[15] J. Chen et al. "Reduced-Complexity Decoding of LDPC Codes". In: *Communications, IEEE Transactions on* 53.8 (Aug. 2005), pp. 1288–1299. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2005.852852.

[16] L. Codrescu et al. "Hexagon DSP: An Architecture Optimized for Mobile Multimedia and Communications". In: *IEEE Micro* 34.2 (Mar. 2014), pp. 34–43. ISSN: 0272-1732. DOI: 10.1109/MM.2014.12.

[17] P. Cook and W. Bonser. "Architectural overview of the SPEAKeasy system". In: *Selected Areas in Communications, IEEE Journal on* 17.4 (Apr. 1999), pp. 650–661. ISSN: 0733-8716. DOI: 10.1109/49.761042.

[18] CRFS Ltd. *RFeye Node*. Data Sheet NOD-EYE0002, http://media.crfs.com/uploads/files/1/crfs-md00011-c07-rfeye-node.pdf (Accessed 15.06.2014). 2014.

[19] B. Davies and T. Davies. "Microprocessor implementation of tactical modems for data transmissions over v.h.f. radios". In: *Radio and Electronic Engineer* 49.4 (Apr. 1979), pp. 204–210. ISSN: 0033-7722. DOI: 10.1049/ree.1979.0039.

[20] E. Demers. *Evolution of AMD Graphics*. Presentation, developer.amd.com/wordpress/media/2013/06/6-Demers-FINAL.pdf [Online; accessed 25.02.2016], AMD. 2013.

[21] Draft ETSI TR 102 831 V0.10.4. *Implementation guidelines for a second generation digital terrestrial television broadcasting system (DVB-T2)*. ETSI Technical Report. 2010.

[22] N. El Gemayel et al. “A low cost TDOA localization system: Setup, challenges and results”. In: *Positioning Navigation and Communication (WPNC), 2013 10th Workshop on*. Mar. 2013, pp. 1–4. DOI: 10.1109/WPNC.2013.6533293.

[23] M. Enright and C. Kurby. “A signals of opportunity based cooperative navigation network”. In: *Aerospace Electronics Conference (NAECON), Proceedings of the IEEE 2009 National*. July 2009, pp. 213–218. DOI: 10.1109/NAECON.2009.5426626.

[24] ETSI EN 302 769 V1.2.1. *Frame structure channel coding and modulation for a second generation digital transmission system for cable systems (DVB-C2)*. ETSI Technical Report. 2011.

[25] ETSI EN 302 755 v1.1.1. *Digital Video Broadcasting (DVB); Frame Structure Channel Coding and Modulation for a Second Generation Digital Terrestrial Television Broadcasting System (DVB-T2)*. ETSI Technical Report. 2009.

[26] M. Ettus. *Ettus Research LLC*. Website, Wayback Machine archived copy. https://web.archive.org/web/20050203003152/http://www.ettus.com/ [Online; accessed 4.8.2014]. Feb. 2005.

[27] M. Ettus. *RE: radio front end ?* Discuss-gnuradio mailing list corresepondence; https://lists.gnu.org/archive/html/discuss-gnuradio/2002-03/msg00019.html [Online; accessed 6.8.2014]. Mar. 2002.

[28] M. Ettus. *USRP User's and Developer's Guide*. Manual, Archived copy: https://web.archive.org/web/20051218022036/http://www.ettus.com/downloads/usrp_guide.pdf [Online; accessed 4.8.2014]. 2005.

[29] Ettus Research. *USRP Software Defined Radio (SDR) online catalog*. Website. https://www.ettus.com/product [Online; accessed 22.6.2016]. 2016.

[30] G. Falcão, L. Sousa, and V. Silva. “Massive parallel LDPC decoding on GPU”. In: *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*. PPoPP '08. Salt Lake City, UT, USA: ACM, 2008, pp. 83–90. ISBN: 978-1-59593-795-7. DOI: 10.1145/1345206.1345221.

[31] G. Falcão et al. “GPU-based DVB-S2 LDPC decoder with high throughput and fast error floor detection”. In: *Electronics Letters* 47.9 (Apr. 2011), pp. 542–543. ISSN: 0013-5194. DOI: 10.1049/el.2011.0201.

[32] G. Falcao et al. “Portable LDPC Decoding on Multicores Using OpenCL [Applications Corner]”. In: *IEEE Signal Processing Magazine* 29.4 (July 2012), pp. 81–109. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2192212.

[33] G. Falcao, L. Sousa, and V. Silva. “Massively LDPC Decoding on Multicore Architectures”. In: *IEEE Transactions on Parallel and Distributed Systems* 22.2 (Feb. 2011), pp. 309–322. ISSN: 1045-9219. DOI: 10.1109/TPDS.2010.66.

[34] G. Falcao et al. “High coded data rate and multicodeword WiMAX LDPC decoding on Cell/BE”. In: *Electronics Letters* 44.24 (2008), pp. 1415–1416.

[35] G. Falcão et al. “Parallel LDPC decoding on GPUs using a stream-based computing approach”. In: *Journal of Computer Science and Technology* 24.5 (2009), pp. 913–924.

[36] J. Gilmore. *Draft letter to Cahners re GNU Radio*. Discuss-gnuradio mailing list corresepondence; https://lists.gnu.org/archive/html/discuss-gnuradio/2002-01/msg00006.html [Online; accessed 4.8.2014]. Jan. 2002.

[37] GNU Radio. *GNU Radio - The Free & Open Software Radio Ecosystem*. http://gnuradio.org/ [Online; accessed 6.8.2014]. 2014.

[38] C. R. A. Gonzalez et al. “Open-source SCA-based core framework and rapid development tools enable software-defined radio education and research”. In: *IEEE Communications Magazine* 47.10 (Oct. 2009), pp. 48–55. ISSN: 0163-6804. DOI: 10.1109/MCOM.2009.5273808.

[39] Great Scott Gadgets. *HackRF*. https://greatscottgadgets.com/hackrf/ [Online; accessed 6.8.2014]. 2014.

[40] S. Grönroos and J. Björkqvist. “Performance evaluation of LDPC decoding on a general purpose mobile CPU”. In: *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. Dec. 2013, pp. 1278–1281. DOI: 10.1109/GlobalSIP.2013.6737142. URL: https://dx.doi.org/10.1109/GlobalSIP.2013.6737142.

[41] S. Grönroos, K. Nybom, and J. Björkqvist. “An Efficient GPU-Based LDPC Decoder for Long Codewords”. In: *Proceedings of the SDR 11 Technical Conference and Product Exposition*. Wireless Innovation Forum, Inc., 2011, pp. 272 –279.

[42] S. Grönroos, K. Nybom, and J. Björkqvist. “Complexity Analysis of Software Defined DVB-T2 Physical Layer”. In: *Proceedings of the SDR ’10 Technical Conference and Product Exposition*. Wireless Innovation Forum, Inc., 2010, pp. 634 –639.

[43] S. Grönroos, K. Nybom, and J. Björkqvist. "Complexity analysis of software defined DVB-T2 physical layer". In: *Analog Integrated Circuits and Signal Processing* 69.2 (2011), pp. 131–142. ISSN: 1573-1979. DOI: 10.1007/s10470-011-9724-4. URL: http://dx.doi.org/10.1007/s10470-011-9724-4.

[44] S. Grönroos, K. Nybom, and J. Björkqvist. "DVB-T2 Rotated Constellation Demapping on a GPU". In: *Proceedings of SDR-WinnComm 2013: Wireless Innovation Conference and Product Exposition*. The Wireless Innovation Forum, 2013, pp. 233 –238.

[45] S. Grönroos, K. Nybom, and J. Björkqvist. "Efficient GPU and CPU-based LDPC decoders for long codewords". In: *Analog Integrated Circuits and Signal Processing* 73.2 (2012), pp. 583–595. ISSN: 1573-1979. DOI: 10.1007/s10470-012-9895-7. URL: http://dx.doi.org/10.1007/s10470-012-9895-7.

[46] S. Grönroos, K. Nybom, and J. Björkqvist. "Implementation and performance analysis of DVB-T2 rotated constellation demappers on a GPU". In: *Analog Integrated Circuits and Signal Processing* 78.3 (2014), pp. 589–598. ISSN: 1573-1979. DOI: 10.1007/s10470-013-0101-3. URL: http://dx.doi.org/10.1007/s10470-013-0101-3.

[47] S. Grönroos, K. Nybom, and J. Björkqvist. "Synchronization of Low-Cost Distributed Spectrum Sensing Nodes for Multilateration-Based Geolocation". In: *Proceedings of SDR-WinnComm 2015: Wireless Innovation Conference on Wireless Communications Technologies and Software Defined Radio*. Wireless Innovation Forum Conference on Wireless Communications Technologies and Software Defined Radio. The Wireless Innovation Forum, 2015, pp. 223 –229.

[48] S. Grönroos et al. "Distributed Spectrum Sensing Using Low Cost Hardware". In: *Proceedings of WInnComm-Europe 2014 Wireless Innovation European Conference on Wireless Communications Technologies and Software Defined Radio*. Wireless Innovation Forum, 2014, pp. 1 –8.

[49] S. Grönroos et al. "Distributed Spectrum Sensing Using Low Cost Hardware". In: *Journal of Signal Processing Systems* (2015), pp. 1–13. ISSN: 1939-8018. DOI: 10.1007/s11265-015-1033-1. URL: http://dx.doi.org/10.1007/s11265-015-1033-1.

[50] Y. Gu, A. Lo, and I. Niemegeers. "A survey of indoor positioning systems for wireless personal networks". In: *IEEE Communications Surveys Tutorials* 11.1 (Mar. 2009), pp. 13–32. ISSN: 1553-877X. DOI: 10.1109/SURV.2009.090103.

[51] X. Han, K. Niu, and Z. He. "Implementation of IEEE 802.11 n LDPC codes based on general purpose processors". In: *Communication Technology (ICCT), 2013 15th IEEE International Conference on*. IEEE, 2013, pp. 218–222.

[52] P. Hasse and J. Robert. "A Software-Based Real-Time DVB-C2 Receiver". In: *Broadband Multimedia Systems and Broadcasting (BMSB), 2011. IEEE International Symposium on*. June 2011.

[53] J. Hyunwoo, C. Junho, and S. Wonyong. "Massively parallel implementation of cyclic LDPC codes on a general purpose graphics processing unit". In: *Signal Processing Systems, 2009. SiPS 2009. IEEE Workshop on*. Oct. 2009, pp. 285–290. DOI: 10.1109/SIPS.2009.5336268.

[54] C. L. I et al. "Toward green and soft: a 5G perspective". In: *IEEE Communications Magazine* 52.2 (Feb. 2014), pp. 66–73. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6736745.

[55] IEEE. "IEEE Standard Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management". In: *IEEE Std 1900.1-2008* (Sept. 2008), pp. c1–48. DOI: 10.1109/IEEESTD.2008.4633734.

[56] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual*. Manual, http://www.intel.com. 2015.

[57] M. Ismert. "Making Commodity PCs Fit for Signal Processing". In: *Proceedings of the Annual Conference on USENIX Annual Technical Conference*. ATEC '98. New Orleans, Louisiana: USENIX Association, 1998, pp. 19–19. URL: http://dl.acm.org/citation.cfm?id=1268256.1268275.

[58] H. Ji, J. Cho, and W. Sung. "Memory Access Optimized Implementation of Cyclic and Quasi-Cyclic LDPC Codes on a GPGPU". en. In: *Journal of Signal Processing Systems* 64.1 (July 2011), pp. 149–159. ISSN: 1939-8018, 1939-8115. DOI: 10.1007/s11265-010-0547-9.

[59] S. Kang and J. Moon. "Parallel LDPC decoder implementation on GPU based on unbalanced memory coalescing". In: *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3692–3697.

[60] J. Kerttula et al. "Implementing TD-LTE As Software Defined Radio in General Purpose Processor". In: *Proceedings of the 2014 ACM Workshop on Software Radio Implementation Forum*. SRIF '14. Chicago, Illinois, USA: ACM, 2014, pp. 61–68. ISBN: 978-1-4503-2995-8. DOI: 10.1145/2627788.2627793. URL: http://doi.acm.org/10.1145/2627788.2627793.

[61] K. Kim et al. “One-Dimensional Soft-Demapping Algorithms for Rotated QAM and Software Implementation on DSP”. In: *IEEE Transactions on Signal Processing* 61.15 (Aug. 2013), pp. 3918–3930. ISSN: 1053-587X. DOI: 10.1109/TSP.2013.2262681.

[62] K. Kim, K. Bae, and H. Yang. “One-dimensional soft-demapping using decorrelation with interference cancellation for rotated QAM constellations”. In: *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*. Jan. 2012, pp. 787 –791. DOI: 10.1109/CCNC.2012.6181165.

[63] C. Kocks et al. “A DVB-T2 receiver realization based on a software-defined radio concept”. In: *Communications, Control and Signal Processing (ISCCSP), 2010 4th International Symposium on*. Mar. 2010, pp. 1–4. DOI: 10.1109/ISCCSP.2010.5463488.

[64] F. Kschischang and B. Frey. “Iterative decoding of compound codes by probability propagation in graphical models”. In: *Selected Areas in Communications, IEEE Journal on* 16.2 (Feb. 1998), pp. 219–230. ISSN: 0733-8716. DOI: 10.1109/49.661110.

[65] J. Kumagai. “Radio Revolutionaries”. In: *Spectrum, IEEE* 44.1 (Jan. 2007), pp. 28–32. ISSN: 0018-9235. DOI: 10.1109/MSPEC.2007.273037.

[66] R. Lackey and D. Upmal. “Speakeasy: the military software radio”. In: *Communications Magazine, IEEE* 33.5 (May 1995), pp. 56–61. ISSN: 0163-6804. DOI: 10.1109/35.392998.

[67] B. Le Gal and C. Jego. “High-Throughput LDPC Decoder on Low-Power Embedded Processors”. In: *IEEE Communications Letters* 19.11 (Nov. 2015), pp. 1861–1864. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2015.2477081.

[68] B. Le Gal and C. Jego. “High-Throughput Multi-Core LDPC Decoders Based on x86 Processor”. In: *IEEE Transactions on Parallel and Distributed Systems* 27.5 (May 2016), pp. 1373–1386. ISSN: 1045-9219. DOI: 10.1109/TPDS.2015.2435787.

[69] B. Le Gal, C. Jego, and J. Crenne. “A High Throughput Efficient Approach for Decoding LDPC Codes onto GPU Devices”. In: *IEEE Embedded Systems Letters* 6.2 (June 2014), pp. 29–32. ISSN: 1943-0663, 1943-0671. DOI: 10.1109/LES.2014.2311317.

[70] M. Li et al. “A shuffled iterative bit-interleaved coded modulation receiver for the DVB-T2 standard: Design, implementation and FPGA prototyping”. In: *2011 IEEE Workshop on Signal Processing Systems (SiPS)*. Oct. 2011, pp. 55–60. DOI: 10.1109/SiPS.2011.6088949.

[71] R. Li et al. "A multi-standard efficient column-layered LDPC decoder for software defined radio on GPUs". In: *Signal Processing Advances in Wireless Communications (SPAWC), 2013 IEEE 14th Workshop on*. IEEE, 2013, pp. 724–728.

[72] Y. Lin and W. Niu. "High throughput LDPC decoder on GPU". In: *Communications Letters, IEEE* 18.2 (2014), pp. 344–347.

[73] D. MacKay. "Good error-correcting codes based on very sparse matrices". In: *Information Theory, IEEE Transactions on* 45.2 (Mar. 1999), pp. 399–431. ISSN: 0018-9448. DOI: 10.1109/18.748992.

[74] M. Mansour and N. Shanbhag. "Memory-efficient turbo decoder architectures for LDPC codes". In: *Signal Processing Systems, 2002. (SIPS '02). IEEE Workshop on*. Oct. 2002, pp. 159–164. DOI: 10.1109/SIPS.2002.1049702.

[75] F. J. Martínez-Zaldívar et al. "Tridimensional block multiword LDPC decoding on GPUs". en. In: *The Journal of Supercomputing* 58.3 (Dec. 2011), pp. 314–322. ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-011-0587-3.

[76] J. Mitola and J. Maguire G.Q. "Cognitive radio: making software radios more personal". In: *Personal Communications, IEEE* 6.4 (Aug. 1999), pp. 13–18. ISSN: 1070-9916. DOI: 10.1109/98.788210.

[77] J. Mitola. "The software radio architecture". In: *Communications Magazine, IEEE* 33.5 (1995), pp. 26 –38. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=393001 (visited on 07/30/2013).

[78] J. Mitola III. "Software radios: Survey, critical evaluation and future directions". In: *Aerospace and Electronic Systems Magazine, IEEE* 8.4 (1993), pp. 25 –36. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=210638 (visited on 07/30/2013).

[79] Mobile Dev & Design. *New SDR release builds on existing free software toolkit*. http://mobiledevdesign.com/news/new-sdr-release-builds-existing-free-software-toolkit [Online; accessed 4.8.2014]. Nov. 2006.

[80] "New Research Lab Leads to Unique Radio Receiver". In: *E-Systems Team* 5 (4 May 1985).

[81] A. Nika et al. "Towards Commoditized Real-time Spectrum Monitoring". In: *Proceedings of the 1st ACM Workshop on Hot Topics in Wireless*. HotWireless '14. Maui, Hawaii, USA: ACM, 2014, pp. 25–30. ISBN: 978-1-4503-3076-3. DOI: 10.1145/2643614.2643615. URL: http://doi.acm.org/10.1145/2643614.2643615.

[82] N. Nikaein et al. “Demo: OpenAirInterface: An Open LTE Network in a PC”. In: *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*. MobiCom ’14. Maui, Hawaii, USA: ACM, 2014, pp. 305–308. ISBN: 978-1-4503-2783-1. DOI: 10.1145/2639108.2641745. URL: http://doi.acm.org/10.1145/2639108.2641745.

[83] N. Nikaein et al. “OpenAirInterface: A Flexible Platform for 5G Research”. In: *SIGCOMM Comput. Commun. Rev.* 44.5 (Oct. 2014), pp. 33–38. ISSN: 0146-4833. DOI: 10.1145/2677046.2677053. URL: http://doi.acm.org/10.1145/2677046.2677053.

[84] NVIDIA. *NVIDIA’s Next Generation CUDA Compute Architecture: Fermi*. Whitepaper, http://www.nvidia.com. 2009.

[85] K. Nybom et al. *A Software Based Implementation of the DVB-T2 Bit Interleaved Coding and Modulation Transmission Chain*. Tech. rep. 948. Turku Centre for Computer Science, 2009.

[86] OpenLTE. *OpenLTE website*. http://openlte.sourceforge.net [Online; accessed 13.06.16]. 2016.

[87] Osmocom. *OsmocomSDR (Wiki)*. http://sdr.osmocom.org/trac/wiki/rtl-sdr [Online; accessed 13.06.14]. 2014.

[88] X. Pan et al. “A high throughput LDPC decoder in CMMB based on virtual radio”. In: *Wireless Communications and Networking Conference Workshops (WCNCW), 2013 IEEE*. IEEE, 2013, pp. 95–99.

[89] V. Pellegrini, G. Bacci, and M. Luise. “Soft-DVB, a Fully Software, GNURadio Based ETSI DVB-T Modulator”. In: *5th Karlsruhe Workshop on Software Radios* (2008).

[90] J. A. del Peral-Rosado et al. “Comparative results analysis on positioning with real LTE signals and low-cost hardware platforms”. In: *2014 7th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC)*. Dec. 2014, pp. 1–8. DOI: 10.1109/NAVITEC.2014.7045148.

[91] D. Pfammatter, D. Giustiniano, and V. Lenders. “A Software-defined Sensor Architecture for Large-scale Wideband Spectrum Monitoring”. In: *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*. IPSN ’15. Seattle, Washington: ACM, 2015, pp. 71–82. ISBN: 978-1-4503-3475-4. DOI: 10.1145/2737095.2737119. URL: http://doi.acm.org/10.1145/2737095.2737119.

[92] M. Rabinowitz and J. Spilker J.J. “A new positioning system using television synchronization signals”. In: *Broadcasting, IEEE Transactions on* 51.1 (Mar. 2005), pp. 51–61. ISSN: 0018-9316. DOI: 10.1109/TBC.2004.837876.

[93] A. Rege. *An Introduction to Modern GPU Architecture*. Presentation, ftp://download.nvidia.com/developer/cuda/seminar/TDCI_Arch.pdf [Online; accessed 25.02.2016], NVIDIA.

[94] K. Rupp. *40 Years of Microprocessor Trend Data*. https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/ [Online; accessed 24.2.2016]. June 2015.

[95] *SDRF Cognitive Radio Definitions*. SDRF-06-R-0011-V1.0.0. Nov. 2006. URL: http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf.

[96] E. Sharon, S. Litsyn, and J. Goldberger. "An efficient message-passing schedule for LDPC decoding". In: *Electrical and Electronics Engineers in Israel, 2004. Proceedings. 2004 23rd IEEE Convention of*. Sept. 2004, pp. 223–226.

[97] A. Shokrollahi. "Coding, Cryptography and Combinatorics". In: ed. by K. Feng, H. Niederreiter, and C. Xing. Basel: Birkhäuser Basel, 2004. Chap. LDPC Codes: An Introduction, pp. 85–110. ISBN: 978-3-0348-7865-4. DOI: 10.1007/978-3-0348-7865-4_5. URL: http://dx.doi.org/10.1007/978-3-0348-7865-4_5.

[98] W. Shuang, S. Cheng, and W. Qiang. "A parallel decoding algorithm of LDPC codes using CUDA". In: *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*. Oct. 2008, pp. 171–175. DOI: 10.1109/ACSSC.2008.5074385.

[99] P. D. Sutton et al. "Iris: an architecture for cognitive radio networking testbeds". In: *IEEE Communications Magazine* 48.9 (Sept. 2010), pp. 114–122. ISSN: 0163-6804. DOI: 10.1109/MCOM.2010.5560595.

[100] Texas Instruments. *TMS320C66x DSP - CPU and Instruction Set*. Reference Guide (SPRUGH7), http://www.ti.com. 2010.

[101] The Amp Hour Electronics Podcast. *An Interview with Matt Ettus – Quality Quadrature Quidam*. Audio interview. http://www.theamphour.com/the-amp-hour-101-quality-quadrature-quidam/ [Online; accessed 4.8.2014]. June 2012.

[102] P. Urard et al. "A 135Mbps DVB-S2 compliant codec based on 64800-bit LDPC and BCH codes (ISSCC paper 24.3)". In: *Proceedings of the 42nd annual Design Automation Conference*. ACM, 2005, pp. 547–548.

[103] L. Vangelista et al. "Key technologies for next-generation terrestrial digital television standard DVB-T2". In: *Communications Magazine, IEEE* 47.10 (Oct. 2009), pp. 146–153. ISSN: 0163-6804. DOI: 10.1109/MCOM.2009.5273822.

[104] G. Wang et al. “A massively parallel implementation of QC-LDPC decoder on GPU”. In: *Application Specific Processors (SASP), 2011 IEEE 9th Symposium on*. IEEE, 2011, pp. 82–85.

[105] G. Wang et al. “GPU accelerated scalable parallel decoding of LDPC codes”. In: *Signals, Systems and Computers (ASILOMAR), 2011 Conference Record of the Forty Fifth Asilomar Conference on*. IEEE, 2011, pp. 2053–2057.

[106] G. Wang et al. “High throughput low latency LDPC decoding on GPU for SDR systems”. In: *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*. IEEE, 2013, pp. 1258–1261.

[107] M. Wellens et al. “Evaluation of Cooperative Spectrum Sensing Based on Large Scale Measurements”. In: *New Frontiers in Dynamic Spectrum Access Networks, 2008. DySPAN 2008. 3rd IEEE Symposium on*. Oct. 2008, pp. 1–12. DOI: 10.1109/DYSPAN.2008.27.

[108] N. Wiberg. “Codes and Decoding on General Graphs”. PhD thesis. Linköping University, 1996.

[109] C. Yan and H. Fan. “Asynchronous differential TDOA for non-GPS navigation using signals of opportunity”. In: *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. Mar. 2008, pp. 5312–5315. DOI: 10.1109/ICASSP.2008.4518859.

[110] C. Yang et al. “Cooperative position location with signals of opportunity”. In: *Proceedings of the IEEE 2009 National Aerospace Electronics Conference (NAECON)*. July 2009, pp. 18–25. DOI: 10.1109/NAECON.2009.5426658.

[111] J. Zander et al. “On the scalability of cognitive radio: assessing the commercial viability of secondary spectrum access”. In: *IEEE Wireless Communications* 20.2 (Apr. 2013), pp. 28–36. ISSN: 1536-1284. DOI: 10.1109/MWC.2013.6507391.

[112] J. Zhao et al. “High performance LDPC decoder on CELL BE for WiMAX system”. In: *Communications and Mobile Computing (CMC), 2011 Third International Conference on*. IEEE, 2011, pp. 278–281.

Part II

Original Publications

Paper I

Complexity analysis of software defined DVB-T2 physical layer

Stefan Grönroos, Kristian Nybom and Jerker Björkqvist

Originally published in *Analog Integrated Circuits and Signal Processing*, Springer US, 2011.

©2011 Springer. Reprinted with permission.

Extended from:
S. Grönroos, K. Nybom, and J. Björkqvist. "Complexity Analysis of Software Defined DVB-T2 Physical Layer". In: *Proceedings of the SDR '10 Technical Conference and Product Exposition.* Wireless Innovation Forum, Inc., 2010, pp. 634 –639

Paper II

Efficient GPU and CPU-based LDPC decoders for long codewords

Stefan Grönroos, Kristian Nybom and Jerker Björkqvist

Originally published in *Analog Integrated Circuits and Signal Processing*, Springer US, 2012.

©2012 Springer. Reprinted with permission.

Extended from:
S. Grönroos, K. Nybom, and J. Björkqvist. "An Efficient GPU-Based LDPC Decoder for Long Codewords". In: *Proceedings of the SDR 11 Technical Conference and Product Exposition*. Wireless Innovation Forum, Inc., 2011, pp. 272 –279

Paper III

Performance Evaluation of LDPC Decoding on a General Purpose Mobile CPU

Stefan Grönroos and Jerker Björkqvist

Originally published in Proceedings of the 1st *IEEE Global Conference on Signal and Information Processing (GlobalSIP 2013)*, IEEE, 2013.

©2013 IEEE. Reprinted with permission.

Paper IV

Implementation and performance analysis of DVB-T2 rotated constellation demappers on a GPU

Stefan Grönroos, Kristian Nybom and Jerker Björkqvist

Originally published in *Analog Integrated Circuits and Signal Processing*, Springer US, 2014.

©2014 Springer. Reprinted with permission.

Extended from:
S. Grönroos, K. Nybom, and J. Björkqvist. "DVB-T2 Rotated Constellation Demapping on a GPU". in: *Proceedings of SDR-WinnComm 2013: Wireless Innovation Conference and Product Exposition.* The Wireless Innovation Forum, 2013, pp. 233 –238

Paper V

Distributed Spectrum Sensing Using Low Cost Hardware

Stefan Grönroos, Kristian Nybom, Jerker Björkqvist, Juhani Hallio, Jani Auranen and Reijo Ekman

Originally published in *Journal of Signal Processing Systems, Springer US, 2015*

©2015 Springer. Reprinted with permission.

Extended from:
S. Grönroos et al. "Distributed Spectrum Sensing Using Low Cost Hardware". In: *Proceedings of WInnComm-Europe 2014 Wireless Innovation European Conference on Wireless Communications Technologies and Software Defined Radio.* Wireless Innovation Forum, 2014, pp. 1 –8

Paper VI

Synchronization of Low-Cost Distributed Spectrum Sensing Nodes for Multilateration-based Geolocation

Stefan Grönroos, Kristian Nybom and Jerker Björkqvist

Originally published in *Proceedings of WInnComm '15 - Wireless Innovation Forum Conference on Wireless Communications Technologies and Software Defined Radio*, Wireless Innovation Forum, 2015.

©2015 Wireless Innovation Forum. Reprinted with permission.

Turku Centre for Computer Science
TUCS Dissertations

1. **Marjo Lipponen**, On Primitive Solutions of the Post Correspondence Problem
2. **Timo Käkölä**, Dual Information Systems in Hyperknowledge Organizations
3. **Ville Leppänen**, Studies on the Realization of PRAM
4. **Cunsheng Ding**, Cryptographic Counter Generators
5. **Sami Viitanen**, Some New Global Optimization Algorithms
6. **Tapio Salakoski**, Representative Classification of Protein Structures
7. **Thomas Långbacka**, An Interactive Environment Supporting the Development of Formally Correct Programs
8. **Thomas Finne**, A Decision Support System for Improving Information Security
9. **Valeria Mihalache**, Cooperation, Communication, Control. Investigations on Grammar Systems.
10. **Marina Waldén**, Formal Reasoning About Distributed Algorithms
11. **Tero Laihonen**, Estimates on the Covering Radius When the Dual Distance is Known
12. **Lucian Ilie**, Decision Problems on Orders of Words
13. **Jukkapekka Hekanaho**, An Evolutionary Approach to Concept Learning
14. **Jouni Järvinen**, Knowledge Representation and Rough Sets
15. **Tomi Pasanen**, In-Place Algorithms for Sorting Problems
16. **Mika Johnsson**, Operational and Tactical Level Optimization in Printed Circuit Board Assembly
17. **Mats Aspnäs**, Multiprocessor Architecture and Programming: The Hathi-2 System
18. **Anna Mikhajlova**, Ensuring Correctness of Object and Component Systems
19. **Vesa Torvinen**, Construction and Evaluation of the Labour Game Method
20. **Jorma Boberg**, Cluster Analysis. A Mathematical Approach with Applications to Protein Structures
21. **Leonid Mikhajlov**, Software Reuse Mechanisms and Techniques: Safety Versus Flexibility
22. **Timo Kaukoranta**, Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization
23. **Gábor Magyar**, On Solution Approaches for Some Industrially Motivated Combinatorial Optimization Problems
24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming – C++ is more Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-Disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Market Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Frank Tétard**, Managers, Fragmentation of Working Time, and Information Systems

41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations Between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity – A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-Type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images
63. **Tommi Meskanen,** On the NTRU Cryptosystem
64. **Saeed Salehi**, Varieties of Tree Languages
65. **Jukka Arvo**, Efficient Algorithms for Hardware-Accelerated Shadow Computation
66. **Mika Hirvikorpi**, On the Tactical Level Production Planning in Flexible Manufacturing Systems
67. **Adrian Costea**, Computational Intelligence Methods for Quantitative Data Mining
68. **Cristina Seceleanu**, A Methodology for Constructing Correct Reactive Systems
69. **Luigia Petre**, Modeling with Action Systems
70. **Lu Yan**, Systematic Design of Ubiquitous Systems
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory

86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming

128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations
132. **Riikka Vuokko**, A Practice Perspective on Organizational Implementation of Information Technology
133. **Jeanette Heidenberg**, Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches
134. **Yong Liu**, Solving the Puzzle of Mobile Learning Adoption
135. **Stina Ojala**, Towards an Integrative Information Society: Studies on Individuality in Speech and Sign
136. **Matteo Brunelli**, Some Advances in Mathematical Models for Preference Relations
137. **Ville Junnila**, On Identifying and Locating-Dominating Codes
138. **Andrzej Mizera**, Methods for Construction and Analysis of Computational Models in Systems Biology. Applications to the Modelling of the Heat Shock Response and the Self-Assembly of Intermediate Filaments.
139. **Csaba Ráduly-Baka**, Algorithmic Solutions for Combinatorial Problems in Resource Management of Manufacturing Environments
140. **Jari Kyngäs**, Solving Challenging Real-World Scheduling Problems
141. **Arho Suominen**, Notes on Emerging Technologies
142. **József Mezei**, A Quantitative View on Fuzzy Numbers
143. **Marta Olszewska**, On the Impact of Rigorous Approaches on the Quality of Development
144. **Antti Airola**, Kernel-Based Ranking: Methods for Learning and Performace Estimation
145. **Aleksi Saarela**, Word Equations and Related Topics: Independence, Decidability and Characterizations
146. **Lasse Bergroth**, Kahden merkkijonon pisimmän yhteisen alijonon ongelma ja sen ratkaiseminen
147. **Thomas Canhao Xu**, Hardware/Software Co-Design for Multicore Architectures
148. **Tuomas Mäkilä**, Software Development Process Modeling – Developers Perspective to Contemporary Modeling Techniques
149. **Shahrokh Nikou**, Opening the Black-Box of IT Artifacts: Looking into Mobile Service Characteristics and Individual Perception
150. **Alessandro Buoni**, Fraud Detection in the Banking Sector: A Multi-Agent Approach
151. **Mats Neovius**, Trustworthy Context Dependency in Ubiquitous Systems
152. **Fredrik Degerlund**, Scheduling of Guarded Command Based Models
153. **Amir-Mohammad Rahmani-Sane**, Exploration and Design of Power-Efficient Networked Many-Core Systems
154. **Ville Rantala**, On Dynamic Monitoring Methods for Networks-on-Chip
155. **Mikko Pelto**, On Identifying and Locating-Dominating Codes in the Infinite King Grid
156. **Anton Tarasyuk**, Formal Development and Quantitative Verification of Dependable Systems
157. **Muhammad Mohsin Saleemi**, Towards Combining Interactive Mobile TV and Smart Spaces: Architectures, Tools and Application Development
158. **Tommi J. M. Lehtinen**, Numbers and Languages
159. **Peter Sarlin**, Mapping Financial Stability
160. **Alexander Wei Yin**, On Energy Efficient Computing Platforms
161. **Mikołaj Olszewski**, Scaling Up Stepwise Feature Introduction to Construction of Large Software Systems
162. **Maryam Kamali**, Reusable Formal Architectures for Networked Systems
163. **Zhiyuan Yao**, Visual Customer Segmentation and Behavior Analysis – A SOM-Based Approach
164. **Timo Jolivet**, Combinatorics of Pisot Substitutions
165. **Rajeev Kumar Kanth**, Analysis and Life Cycle Assessment of Printed Antennas for Sustainable Wireless Systems
166. **Khalid Latif**, Design Space Exploration for MPSoC Architectures

167. **Bo Yang**, Towards Optimal Application Mapping for Energy-Efficient Many-Core Platforms
168. **Ali Hanzala Khan**, Consistency of UML Based Designs Using Ontology Reasoners
169. **Sonja Leskinen**, m-Equine: IS Support for the Horse Industry
170. **Fareed Ahmed Jokhio**, Video Transcoding in a Distributed Cloud Computing Environment
171. **Moazzam Fareed Niazi**, A Model-Based Development and Verification Framework for Distributed System-on-Chip Architecture
172. **Mari Huova**, Combinatorics on Words: New Aspects on Avoidability, Defect Effect, Equations and Palindromes
173. **Ville Timonen**, Scalable Algorithms for Height Field Illumination
174. **Henri Korvela**, Virtual Communities – A Virtual Treasure Trove for End-User Developers
175. **Kameswar Rao Vaddina**, Thermal-Aware Networked Many-Core Systems
176. **Janne Lahtiranta**, New and Emerging Challenges of the ICT-Mediated Health and Well-Being Services
177. **Irum Rauf**, Design and Validation of Stateful Composite RESTful Web Services
178. **Jari Björne**, Biomedical Event Extraction with Machine Learning
179. **Katri Haverinen**, Natural Language Processing Resources for Finnish: Corpus Development in the General and Clinical Domains
180. **Ville Salo**, Subshifts with Simple Cellular Automata
181. **Johan Ersfolk**, Scheduling Dynamic Dataflow Graphs
182. **Hongyan Liu**, On Advancing Business Intelligence in the Electricity Retail Market
183. **Adnan Ashraf**, Cost-Efficient Virtual Machine Management: Provisioning, Admission Control, and Consolidation
184. **Muhammad Nazrul Islam**, Design and Evaluation of Web Interface Signs to Improve Web Usability: A Semiotic Framework
185. **Johannes Tuikkala**, Algorithmic Techniques in Gene Expression Processing: From Imputation to Visualization
186. **Natalia Díaz Rodríguez**, Semantic and Fuzzy Modelling for Human Behaviour Recognition in Smart Spaces. A Case Study on Ambient Assisted Living
187. **Mikko Pänkäälä**, Potential and Challenges of Analog Reconfigurable Computation in Modern and Future CMOS
188. **Sami Hyrynsalmi**, Letters from the War of Ecosystems – An Analysis of Independent Software Vendors in Mobile Application Marketplaces
189. **Seppo Pulkkinen**, Efficient Optimization Algorithms for Nonlinear Data Analysis
190. **Sami Pyöttiälä**, Optimization and Measuring Techniques for Collect-and-Place Machines in Printed Circuit Board Industry
191. **Syed Mohammad Asad Hassan Jafri**, Virtual Runtime Application Partitions for Resource Management in Massively Parallel Architectures
192. **Toni Ernvall**, On Distributed Storage Codes
193. **Yuliya Prokhorova**, Rigorous Development of Safety-Critical Systems
194. **Olli Lahdenoja**, Local Binary Patterns in Focal-Plane Processing – Analysis and Applications
195. **Annika H. Holmbom**, Visual Analytics for Behavioral and Niche Market Segmentation
196. **Sergey Ostroumov**, Agent-Based Management System for Many-Core Platforms: Rigorous Design and Efficient Implementation
197. **Espen Suenson**, How Computer Programmers Work – Understanding Software Development in Practise
198. **Tuomas Poikela**, Readout Architectures for Hybrid Pixel Detector Readout Chips
199. **Bogdan Iancu**, Quantitative Refinement of Reaction-Based Biomodels
200. **Ilkka Törmä**, Structural and Computational Existence Results for Multidimensional Subshifts
201. **Sebastian Okser**, Scalable Feature Selection Applications for Genome-Wide Association Studies of Complex Diseases
202. **Fredrik Abbors**, Model-Based Testing of Software Systems: Functionality and Performance
203. **Inna Pereverzeva**, Formal Development of Resilient Distributed Systems
204. **Mikhail Barash**, Defining Contexts in Context-Free Grammars
205. **Sepinoud Azimi**, Computational Models for and from Biology: Simple Gene Assembly and Reaction Systems
206. **Petter Sandvik**, Formal Modelling for Digital Media Distribution

207. **Jongyun Moon**, Hydrogen Sensor Application of Anodic Titanium Oxide Nanostructures
208. **Simon Holmbacka**, Energy Aware Software for Many-Core Systems
209. **Charalampos Zinoviadis**, Hierarchy and Expansiveness in Two-Dimensional Subshifts of Finite Type
210. **Mika Murtojärvi**, Efficient Algorithms for Coastal Geographic Problems
211. **Sami Mäkelä**, Cohesion Metrics for Improving Software Quality
212. **Eyal Eshet**, Examining Human-Centered Design Practice in the Mobile Apps Era
213. **Jetro Vesti**, Rich Words and Balanced Words
214. **Jarkko Peltomäki**, Privileged Words and Sturmian Words
215. **Fahimeh Farahnakian**, Energy and Performance Management of Virtual Machines: Provisioning, Placement and Consolidation
216. **Diana-Elena Gratie**, Refinement of Biomodels Using Petri Nets
217. **Harri Merisaari**, Algorithmic Analysis Techniques for Molecular Imaging
218. **Stefan Grönroos**, Efficient and Low-Cost Software Defined Radio on Commodity Hardware

Turku
Centre *for*
Computer
Science

http://www. tucs.fi

tucs@abo.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Faculty of Science and Engineering
- Computer Engineering
- Computer Science

Faculty of Social Sciences, Business and Economics
- Information Systems

ISBN 978-952-12-3456-9
ISSN 1239-1883

Stefan Grönroos

Efficient and Low-Cost Software Defined Radio on Commodity Hardware